Submodular Data Selection and Augmentation for Resource-Efficient Learning

Lilly Kumari

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2025

Reading Committee:

Jeffrey Bilmes, Chair

Hannaneh Hajishirzi

Tianyi Zhou

Program Authorized to Offer Degree:
Electrical and Computer Engineering

University of Washington

Abstract

Submodular Data Selection and Augmentation
for Resource-Efficient Learning

Lilly Kumari

Chair of the Supervisory Committee:
Jeffrey Bilmes
Department of Electrical and Computer Engineering

The increasing complexity of modern machine learning (ML) systems, particularly large-scale transformer-based models, presents significant challenges with respect to computational, data, and memory efficiency. Models with billions of parameters, such as Large Language Models (LLMs), Vision Transformers (ViTs), and Multimodal Large Language Models (MLLMs), require vast datasets, substantial memory, and extensive compute resources, making them difficult to scale and deploy—especially in resource-constrained and data-scarce environments. These challenges are further compounded by the redundancy present in both real and synthetic data, which leads to inefficiencies in data annotation, training, and inference stages by increasing resource overhead and making it difficult to extract meaningful insights from the data. Additionally, input context selection in LLMs and MLLMs is critical for improving their test-time efficiency, as these models often process long sequences of text or multimodal information. Processing redundant and irrelevant context strains compute and memory resources and can degrade output quality, highlighting the importance of effective context selection in optimizing resource usage. To address these above-mentioned challenges and improve the efficiency and accessibility of ML systems, we require strategies that optimize resource utilization through high-quality data selection, augmentation, and efficient input context selection—particularly in LLMs and MLLMs.

We explore two complementary approaches—submodular data selection and data augmentation—to enhance the efficiency of ML systems without compromising model performance. The first approach leverages submodular optimization to model diversity, representativeness, and relevance in selecting data subsets and input contexts for training and inference. The second approach focuses on data augmentation to enhance data utility, improve model robustness, and mitigate catastrophic forgetting in resource-constrained settings such as continual learning.

To enable efficient query-focused data selection, we propose *Submodular Span Summarization (S3)*, a framework that selects diverse and query-relevant data subsets. We demonstrate its effectiveness across multiple data modalities for query-focused summarization tasks. The S3 framework provides an effective solution for optimizing annotation, training, and inference costs by selecting query-relevant and representative subsets, facilitating tasks such as active learning, targeted data selection, and efficient input context selection for LLMs and MLLMs. Extending this, we propose *Div-S3*, an end-to-end submodular optimization approach for in-context learning (ICL) using LLMs to enable efficient exemplar selection and retrieval while improving downstream task performance under data annotation constraints.

Building on the principles of submodularity, we further optimize LLMs inference efficiency with *BumbleBee*, a novel key-value (KV) cache summarization algorithm. As LLMs scale, maintaining large KV caches for autoregressive inference becomes increasingly resource-intensive. BumbleBee reduces computational overhead and memory footprint, allowing LLMs to maintain an effectively infinite context without any architectural modifications or additional fine-tuning. We extend BumbleBee to multimodal tasks and propose *VisionBee*, a submodular optimization framework that reduces the number of visual tokens processed by MLLMs significantly while maintaining performance on several image and video understanding tasks.

Beyond data selection, we explore data augmentation for continual learning, where an

ML model learns from a stream of data coming from different tasks without revisiting previous tasks' data. We introduce *Retrospective Adversarial Replay (RAR)*, which generates informative replay instances that capture the forgetting frontier, using adversarial augmentations and MixUp to enhance data diversity. Overall, our experiments across diverse tasks and data modalities show that our proposed data selection and augmentation approaches significantly improve resource efficiency during the annotation, training, and inference stages while maintaining model performance.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to first express my deepest gratitude to my advisor, Prof. Jeff Bilmes. He introduced me to the rich and elegant field of submodularity, which forms the foundation of this dissertation. His teachings and insights have profoundly influenced how I approach research—particularly in thinking critically, designing systematic experiments, and expressing complex ideas with clarity, coherence, and precision in academic writing. I am especially thankful for his constant encouragement to explore ideas independently. I remain deeply appreciative of his guidance and unwavering support throughout my Ph.D. journey, as well as the life lessons that have shaped not only my research but also my broader perspective. Working with him has been a defining part of this journey, and his teachings will continue to guide my thinking, curiosity, and problem-solving approach as I embark on the next chapters of my professional life.

I am also profoundly thankful to my dissertation committee members: Prof. Hannaneh Hajishirzi, Prof. Tianyi Zhou, and Prof. Ashis Banerjee for their invaluable feedback and support during my Ph.D. journey.

I feel incredibly fortunate to be a part of the MELODI lab, surrounded by supportive and hardworking labmates who have made this journey both intellectually stimulating and personally fulfilling. I am deeply grateful to my collaborators—Prof. Tianyi Zhou, Prof. Shengjie Wang, Arnav Das, and Gantavya Bhatt—for their insightful discussions and collaborations, from brainstorming ideas to paper writing. Their support, especially during moments of uncertainty and challenge, has been invaluable. I am also thankful to my labmates—Dr. Chandrashekhar Lavania, Armagan Er, and Sahil Verma—for the many enriching discussions. Thank you all for your support and for making my Ph.D. experience

both rewarding and memorable.

I would like to sincerely thank Jennifer Huberman, Stephanie Swanson, and Christie Peralta for their invaluable support in helping me navigate various administrative processes and for their assistance with my teaching assistant responsibilities.

I am also thankful to my mentors during my internships at Google Research and Adobe Research—Dr. Srikumar Ramalingam, Dr. Ayan Chakrabarti, Dr. Ritwik Sinha, and Dr. Niyati Chhaya—for their invaluable guidance and mentorship. Working with them taught me not only core research skills but also how to navigate open-ended problems in real-world industrial settings with a strong focus on scalability and practical impact.

I would like to thank my friends from Roorkee and Seattle: Mehak, Vaishali, Ishu, Prateek, Neela, Srinath, Amrit, Suryansh, Anupriya, and Himanshu. Their unwavering emotional support and motivation during my Ph.D. endeavor have been invaluable.

Finally, I am profoundly grateful to my grandparents for instilling in me a deep sense of curiosity and a lifelong quest for knowledge. I am thankful to my parents for their unwavering support, encouragement, and belief in my decisions throughout this journey. I am also deeply appreciative of my brother for standing by me and taking on family responsibilities during my years of study. I extend heartfelt thanks to my sister, brother-in-law, bua, fufa, and my cousins for their constant encouragement and affection. A very special thank you goes to my two-year-old nephew, whose cheerful presence, endearing baby talk, and joyful babbling over video calls brought light and laughter to some of my most stressful moments—especially during my job search and key Ph.D. milestones. His infectious smile often reminded me to pause, breathe, and find joy in the little things. I am deeply thankful to my partner for his unwavering love, patience, and encouragement through every high and low of this journey. His presence has been my anchor, and his belief in me has been my strength.

DEDICATION

To my Grandparents and Parents

Chapter 1

INTRODUCTION

Modern machine learning (ML) systems have achieved remarkable performance, driven by advancements in model architectures such as transformers [258, 36] and the availability of large-scale datasets [95, 257, 287, 248]. These advancements have transformed fields including, but not limited to, computer vision [386], natural language processing [36], and robotics [294]. However, the increasing complexity of models and the growing size of datasets present significant challenges related to resource utilization, particularly across three critical resource dimensions: data, compute, and memory. For example, state-of-the-art Large Language Models (LLMs) [321, 314], Vision Transformers (ViTs) [81], and Multimodal Large Language Models (MLLMs) [371], with billions of parameters, rely on massive amounts of pre-training data, require vast computational resources for training and inference, and demand substantial memory to store model components such as weights, activations, and intermediate computations. Despite their impressive performance, their high resource requirements often limit their deployment in practical settings, especially in resource-constrained environments like real-time systems, edge devices, or domains where acquiring large labeled datasets is prohibitively expensive or infeasible.

At the same time, the exponential growth of data—spanning text, images, videos, and other modalities—has further added to these challenges. While these datasets are essential for training high-performance models, they often contain redundant information, such as multiple news articles covering the same event or images belonging to certain categories with static background information. This redundancy increases computational inefficiencies and inflates resource requirements during training and inference. Furthermore, the landscape of data generation is evolving rapidly, with a substantial amount of synthetic data [214, 87]

being generated using advanced ML models like LLMs [269, 12], Generative Adversarial Networks (GANs) [60], diffusion models [61], etc. While synthetic data provides opportunities for scalable data generation, it often mirrors the redundancy issues [204] found in real data, making it increasingly difficult to extract meaningful insights as well as inflating resource requirements during training and inference [4]. For example, in many inference scenarios involving LLMs and MLLMs [30, 308, 48], the input context frequently contains redundant and non-relevant information. This not only fails to contribute meaningfully to the task at hand but further increases the strain on computational and memory resources, as the associated costs scale with input sequence length in attention-based models [328]. This underscores the need for efficient input context selection strategies for LLMs and MLLMs.

As the size of these datasets grows, it also becomes increasingly challenging to process them for different steps that are an essential part of any modern ML framework [120, 360]. These steps include but are not limited to data annotation [293, 292], mini-batch training involving both forward and backward propagation [256, 22], and data inference via forward propagation using a massive neural network [284, 18]. Prioritizing the optimization of resource utilization—through careful curation of high-quality datasets [379, 243, 286, 189, 221] and selection of relevant input contexts [78, 97, 77]—can significantly enhance efficiency across the annotation, training, and inference stages. Addressing these resource efficiency challenges necessitates principled methods that tackle both data and input context redundancy and relevance to manage resources effectively while maintaining model performance.

1.1 Key Resource Dimensions in Modern ML Systems

As the complexity of ML models and the size of data sets grow, effectively managing resource utilization becomes a critical challenge to improving the scalability and efficiency of modern ML systems. To better understand the bottlenecks in modern ML systems, we categorize them into three key dimensions: data, compute, and memory, as shown in Fig. 1.1.

1. **Data**: Large-scale datasets are indispensable for training modern ML models, particu-

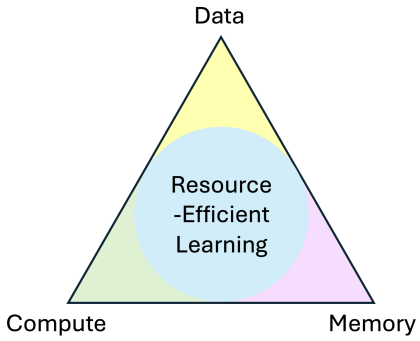


Figure 1.1: Three Key Dimensions for Resource-Efficient Learning

larly during self-supervised pre-training stages [258, 257]. However, the costs associated with collecting and annotating such datasets are often prohibitively high, making it challenging to scale data acquisition. Moreover, the amount of redundancy present in such extensive datasets introduces inefficiencies during training [106, 346, 235, 317] and inference [174, 142, 109], leading to diminishing returns in model performance [4].

2. **Compute**: As models grow in size and complexity in terms of parameter count, their computational demands increase significantly. For example, transformer-based models, particularly the self-attention mechanism, scale quadratically in complexity, making training and inference highly compute-intensive [328, 285]. These demands make deploying such large models infeasible on resource-constrained devices or in real-time systems that require low-latency responses.

3. **Memory**: Modern ML systems require substantial memory resources to store model parameters, intermediate computations, and auxiliary structures, such as key-value (KV) caches [393, 207] in transformers. In autoregressive models, these memory requirements grow with sequence length and batch size, increasing memory (or storage) costs and creating significant challenges for applications that rely on retaining long-term context.

Effectively addressing these bottlenecks requires principled approaches that optimize both

data and input context selection and model efficiency. This thesis explores two distinct but complementary approaches—submodular data selection and data augmentation—that systematically reduce redundancy, optimize resource allocation, and improve efficiency across the ML lifecycle. By intelligently selecting the most relevant and diverse subsets of data, submodular selection minimizes inefficiencies during annotation, training, and inference. Meanwhile, data augmentation enhances model generalization, reducing reliance on extensively labeled datasets while mitigating catastrophic forgetting in resource-constrained paradigms like continual learning.

1.2 Thesis Contributions

1.2.1 Submodular Optimization for Data and Input Context Selection

Data selection, which aims to extract a representative and informative subset from a large dataset, is a fundamental task in ML [322, 291, 348, 346, 203, 235]. The objective is to select a subset that captures the essential characteristics of the original data while minimizing redundancy and maximizing coverage. By selecting the most useful and diverse data points, data selection techniques can significantly reduce annotation and computational costs, improve model interpretability, and facilitate efficient learning from large-scale datasets.

Similarly, input context selection is crucial in LLMs and MLLMs, where retrieving and utilizing relevant information directly impacts both efficiency and performance [126, 97, 264]. Techniques such as retrieval-augmented generation (RAG) [88], in-context learning (ICL) [78], and KV cache summarization (or reduction) [100, 207] rely on selecting the most relevant and diverse contexts from external knowledge sources, past interactions, or provided input sequences to optimize model response generation. By filtering out irrelevant and redundant context, input context selection helps reduce memory overhead and computational costs, particularly in long-context scenarios.

Submodular functions [26] are a powerful tool for data and input context selection and summarization due to their ability to model the diminishing returns property. They natu-

rally capture notions of diversity, representativeness, and coverage. Formally, a function is submodular if the marginal gain (or incremental value) of adding a new item decreases as the size of the selected input set grows. This property aligns well with the objectives of data selection and summarization, as the importance of adding a new data point to the summary diminishes as the summary grows larger.

Mathematically, a submodular function $f : 2^V \rightarrow \mathbb{R}$ defined over the ground set V satisfies the following diminishing returns property:

$$f(v|A) \geq f(v|B) \text{ for any } v \notin B \text{ and } A \subseteq B \subseteq V \tag{1.1}$$

where $f(v|A) = f(A \cup \{v\}) - f(A)$ is the marginal gain of adding v to A. Submodular functions have been successfully applied to various machine learning tasks of finding diverse subsets, including text, image and video summarization [195, 196, 110, 166, 155], feature selection [206, 397], curriculum learning [402, 403], active learning [107, 346], training data selection [347], etc. Given a submodular function f that is non-negative ($\forall A \subseteq V, f(A) \geq 0$) and monotone non-decreasing ($\forall A \subseteq B \subseteq V, f(A) \leq f(B)$), the ground set V can be summarized via submodular maximization under a cardinality constraint, i.e., $\max_{A \subseteq V, |A| \leq k} f(A)$. This can be approximated with a $(1 - \frac{1}{e})$ constant factor guarantee using the greedy algorithm [239, 231]. We assume all submodular functions discussed in this thesis are non-negative and monotone.

This thesis explores how submodular optimization can be applied to data selection and input context selection to improve the resource efficiency of modern ML systems without compromising model performance.

Submodular Span Problem & Conditional Data Summarization

In this thesis, we introduce the concept of submodular span and its application in conditional, i.e., query-focused data summarization. The submodular span problem aims to find a subset of data points relevant to a given query set and diverse with respect to each other. We propose a novel two-stage framework called *Submodular Span Summarization (S3)* that leverages a

single submodular function to model the two important aspects of a query-focused summary: diversity and query relevance. The S3 framework [155] consists of a query-focused data selection stage, followed by a diversification stage that ensures the selected subset, while being query-relevant, is also non-redundant.

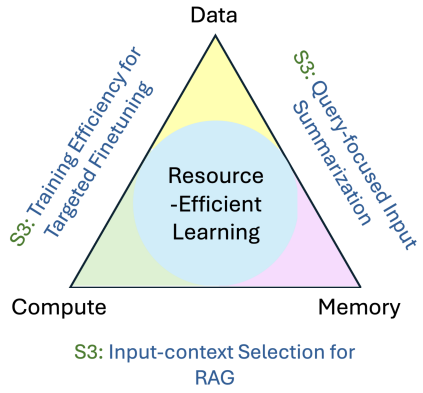


Figure 1.2: Submodular Span Summarization (S3) for Resource-Efficient Learning

We provide theoretical analysis and approximation algorithms for the submodular span problem, extending the notion of the submodular span from matroids to the more general class of polymatroid functions. We demonstrate the effectiveness of the S3 framework on various query-focused summarization tasks, including multi-document summarization, video summarization, and image corpus summarization. Our results show that the S3 framework outperforms existing methods in generating informative and diverse summaries while being computationally efficient.

The S3 framework provides a robust solution for optimizing data annotation, training, and inference costs by enabling the selection of query-relevant and non-redundant subsets of data. It can help facilitate tasks such as active learning [291], targeted data selection [353, 2], and efficient input-context selection in LLMs [156] and MLLMs. Additionally, it can improve downstream applications, including abstractive summarization and dynamic retrieval in models using retrieval-augmented generation (RAG) [46, 180], by prioritizing the most informative and task-relevant content.

Submodularity for Data-Efficient In-Context Learning using LLMs

In-context learning [78] has recently gained significant attention as a promising approach for adapting LLMs to new tasks using only a few task-specific examples, often called demonstrations or exemplars. By providing a limited number of input-output demonstrations belonging to a downstream task, LLMs can learn to perform new tasks without requiring extensive fine-tuning on task-specific training data. However, the effectiveness of in-context learning heavily relies on the selection of informative and diverse examples that can guide the model towards the desired output [201, 311, 220].

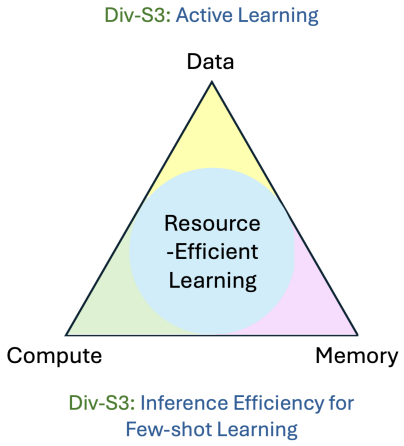


Figure 1.3: Div-S3 for Resource-Efficient In-context Learning using LLMs

Building upon the S3 framework, we propose an end-to-end submodular optimization approach, *Div-S3*, for data-efficient in-context learning using LLMs. Div-S3 [157] addresses the challenge of selecting informative examples by employing submodular optimization in exemplar annotation and retrieval stages. By ensuring diverse subset selection within a fixed annotation budget and leveraging the S3 framework for retrieving relevant and diverse exemplars given a test query, Div-S3 enhances in-context learning performance across multiple NLP tasks compared to baseline methods relying only on similarity-based selection.

Submodularity for KV Cache Summarization in LLMs

The multi-headed self-attention mechanism introduced in [328] is a core component of various state-of-the-art Transformer-based models employed in tasks belonging to different data modalities [262, 81, 350]. Despite its effectiveness in capturing long-range dependencies within sequences, deploying LLMs that utilize this mechanism faces significant challenges. Firstly, the computational and memory costs scale quadratically with the sequence length due to the attention mechanism. Secondly, the autoregressive decoding process, where each token is generated sequentially based on previous tokens, demands extensive memory to store computed key and value representations in KV cache [253] for all preceding tokens. Existing strategies to manage these challenges include truncating input sequences [72, 175], using sliding windows [177, 103], and employing retrieval-based methods [137, 351, 32, 400]. However, these often result in issues like context fragmentation or require large-scale external memory banks.

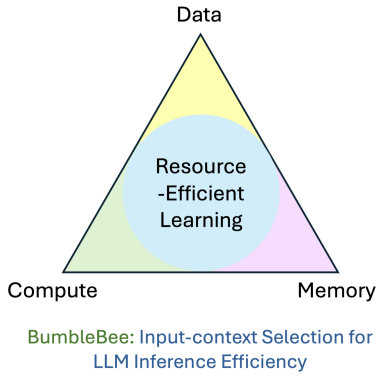


Figure 1.4: BumbleBee for Resource-Efficient Inference using LLMs

We propose *BumbleBee*, a novel approach for KV cache summarization that addresses the limitations of the existing methods. Unlike heuristic-based techniques, BumbleBee [159] reframes KV cache selection as a subset selection problem, evaluating the utility of key-value pairs collectively rather than independently. By considering the importance of each KV attention state in relation to others within the KV cache, BumbleBee enables the utilization of existing LLMs for longer contexts without additional fine-tuning, CPU offloading [298],

or context truncation [24]. Notably, BumbleBee extends the temporal span of the context without unbounded growth of the KV cache, thereby enabling the LLMs to have an infinite context.

Submodularity for Visual Tokens Selection in MLLMs

Like LLMs, MLLMs rely on the information extracted from the multimodal input sequence in the form of visual and textual tokens [386]. The visual tokens are first encoded using a ViT-based vision encoder before being fed through an alignment module to align the vision and language modalities in a shared latent space. Each image or frame sampled from a video contributes hundreds or thousands of visual tokens. For example, LLaVA [198] encodes 336×336 images into 576 tokens, while LLaVA-NeXT [199] expands this to 2880 tokens for 672×672 images—a dramatic increase in sequence length compared to text inputs. This issue is further exacerbated in video-based models, such as Video-LLaVA [192] and VideoPoet [148], where multiple frames contribute hundreds of thousands to millions of visual tokens, leading to prohibitive memory consumption and inference latency. Given the quadratic complexity of self-attention, an efficient mechanism for visual token selection and summarization is crucial to enable scalable inference in resource-constrained environments.

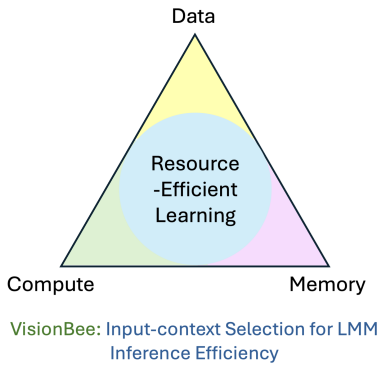


Figure 1.5: VisionBee for Resource-Efficient Inference using MLLMs

In this thesis, we propose *VisionBee*, a submodular optimization framework for visual token selection that ensures optimal trade-offs between efficiency and model performance.

VisionBee formulates visual tokens selection as a submodular optimization problem, ensuring that selected tokens maximize informativeness while minimizing redundancy. Inspired by BumbleBee, which summarizes the LLMs' KV cache tokens using a mixture of submodular functions, VisionBee extends this concept to MLLMs by leveraging the [CLS] token's attention scores and the vision encoder's feature diversity at the patch (or token) level to dynamically filter out redundant visual tokens. Using the submodular function evaluation, VisionBee retains the most informative and diverse tokens, which are then fed to the LLM decoder, significantly reducing the overall sequence length while preserving multimodal reasoning capabilities. By performing visual token selection at the vision encoder stage, our approach removes redundant tokens early, preventing unnecessary computation at the LLM layers and substantially reducing memory footprint and inference latency.

1.2.2 Diverse Adversarial Data Augmentation & Continual Learning

Data augmentation [383] is a simple and effective approach where the training dataset is artificially expanded by applying transformations such as rotations, flips, noise injection, or domain-specific modifications. By increasing data diversity, these transformations improve model robustness and help mitigate challenges such as overfitting [241] and catastrophic forgetting [276]. While data augmentation has long been valuable in traditional ML settings, it becomes even more critical in continual learning (CL), where models must learn under data and resource constraints. The final part of this thesis explores advanced augmentation techniques specifically designed for such resource-constrained environments.

Continual learning (CL)—also known as lifelong learning or incremental learning—refers to a model's ability to learn from a continuous stream of tasks while retaining knowledge acquired from previous tasks [154, 250, 382]. In real-world scenarios, data often arrives sequentially, requiring models to adapt to new tasks while retaining their performance on previously learned tasks. On top of this, continual learning is inherently resource-constrained, with limited access to past data, small memory buffers, and computational limitations. These constraints exacerbate the challenge of catastrophic forgetting [93, 223, 276], where the

model's performance on previous tasks significantly degrades as it learns new tasks.

Data augmentation has proven to be an effective strategy for addressing catastrophic forgetting in continual learning [130]. By generating new training examples that capture the essential characteristics of previous tasks, data augmentation can help the model maintain its performance on old tasks while learning new ones. However, traditional augmentation methods often rely on simple transformations or random perturbations, which may not effectively capture the forgetting frontier—the decision boundary between old and new tasks.

To address this limitation, we propose a novel data augmentation technique called *Retrospective Adversarial Replay (RAR)*, specifically designed for continual learning. RAR [158] generates informative replay instances by focusing on memory (or buffer) samples that are most susceptible to forgetting. It applies targeted adversarial perturbations to these samples, creating challenging examples that force the model to reinforce its understanding of previous tasks. The key idea behind RAR is to produce augmented examples that closely resemble old task data while being difficult for the model to classify correctly, thereby strengthening the model's retention of prior knowledge. This thesis explores the utility of RAR in improving continual learning under resource-constrained settings, demonstrating its ability to mitigate catastrophic forgetting more effectively compared to traditional data augmentation methods.

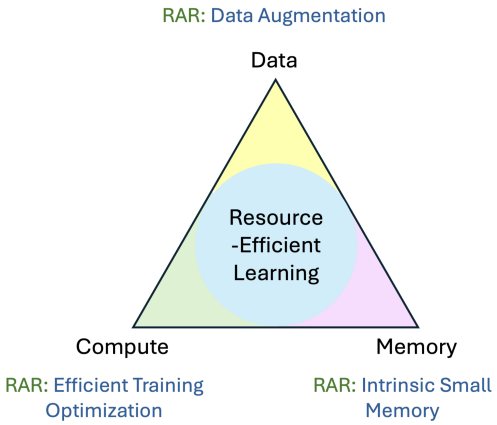


Figure 1.6: RAR for Resource-Efficient Continual Learning of ML models

RAR consists of three main steps: (1) identifying the most likely to be forgotten buffer

samples based on the model's uncertainty or the samples' proximity to the decision boundary; (2) pairing the selected buffer samples with their nearest neighbors from the new task data; and (3) applying targeted adversarial perturbations to the buffer samples, moving them towards the paired new task samples in the model's latent space. By generating adversarial examples that are close to the forgetting frontier, RAR effectively captures the regions where the model is most susceptible to forgetting and provides targeted training signals to mitigate catastrophic forgetting. Extensive experiments on benchmark datasets for continual learning demonstrate the effectiveness of RAR in alleviating catastrophic forgetting and improving the overall accuracy of continual learning models.

1.3 Thesis Outline

The remainder of this thesis is organized as follows:

In Chapter 2, we provide an introduction to submodular functions, submodular optimization algorithms subject to different constraints, and submodular information-theoretic functions. Chapter 3 provides a brief overview of existing literature on data selection and augmentation techniques, specifically addressing the resource efficiency of ML systems. Chapter 4 presents the Submodular Span Summarization (S3) framework for conditional data summarization. We introduce the submodular span problem, provide theoretical analysis and approximation algorithms, and evaluate the S3 framework on various query-focused (or conditional) summarization tasks. Chapter 5 describes the Div-S3 framework for data-efficient in-context learning using LLMs. We detail the submodular optimization approach for exemplar annotation and retrieval and present experimental results on multiple natural language processing tasks.

Chapter 6 describes offline and streaming KV cache summarization algorithms to reduce LLMs' KV cache size. We demonstrate that even with a 5× context length reduction, LLMs can demonstrate comparable performance on various long-range downstream tasks, like multi-document question answering, summarization, etc. Chapter 7 describes the visual tokens selection algorithm, VisionBee, to reduce the length of the multimodal sequence in

MLLMs. Even with a 9× visual tokens length reduction, we demonstrate that VisionBee achieves competitive performance compared to when using all visual tokens across visual question-answering tasks.

Chapter 8 introduces the Retrospective Adversarial Replay (RAR) method for mitigating catastrophic forgetting in continual learning. We describe the RAR algorithm, investigate the role of MixUp in improving the diversity of perturbed samples, and present experimental results on benchmark datasets. Chapter 9 concludes the thesis by discussing future research directions for leveraging submodular optimization to enhance the resource efficiency of modern ML systems. It outlines potential extensions and applications of the proposed frameworks and techniques.

In summary, this thesis contributes to the field of resource-efficient machine learning by developing novel frameworks and techniques for data selection and augmentation, with applications in diverse areas such as in-context learning, context management in LLMs and MLLMs, and continual learning. The proposed methods build upon the principles of submodularity and data augmentation to optimize ML models while maintaining their performance across various tasks. Through comprehensive theoretical analysis, algorithmic development, and empirical validation, this thesis provides critical insights into the design and implementation of data-centric methods to improve the end-to-end efficiency of modern ML systems.

Chapter 2

SUBMODULARITY IN DEEP LEARNING

In this chapter, we begin by defining submodular functions in Sec. 2.1.1, followed by an overview of various submodular functions and their applications in machine learning in Sec. 2.1.2. We then explore existing submodular optimization algorithms subject to cardinality and knapsack constraints in Sec. 2.2. Additionally, we cover the mixture of submodular functions in Sec. 2.1.2 and introduce submodular information-theoretic functions in Sec. 2.3.

2.1 Submodular Functions

Submodularity is a fundamental property of discrete set functions and plays a pivotal role in diverse domains such as mathematics, economics, operations research, and machine learning. This property naturally arises in numerous combinatorial problems, establishing its relevance in fields ranging from machine learning and computer vision to game theory and energy systems [26]. The theoretical foundations of submodularity, combined with its practical applicability across different machine learning tasks such as active learning, data summarization, feature selection, and curriculum learning, position it as an indispensable tool in the current deep learning landscape.

2.1.1 Submodular Function Definition

Given a ground set V of n items, a set function f is defined over this ground set. Formally, $f : 2^V \rightarrow \mathbb{R}$ evaluates the real-valued f-valuation of any subset $A \subseteq V$ such that $f(A) \in \mathbb{R}$. Submodular function is a special class of such set functions that satisfy the diminishing returns property. Mathematically, for any two subsets A and B such that $A \subseteq B \subset V$, and an item a such that $a \in V, a \notin B$, the diminishing returns property is defined as follows:

$$f(A \cup \{a\}) - f(A) \geq f(B \cup \{a\}) - f(B) \tag{2.1}$$

The marginal gain of adding an item a to set $A \subset V$ is referred to as the conditional gain and denoted by $f(a|A)$. Specifically, we have $f(a|A) \triangleq f(A \cup \{a\}) - f(A)$. Eq. (2.1) implies that the marginal utility (or gain) of an incoming element diminishes as the conditioning set grows. This diminishing-returns property naturally arises across varied machine learning and data science tasks, making submodular functions highly effective for modeling different utility functions in machine learning, depending on the task at hand.

Another widely used definition of submodular functions is as follows:

$$f(A) + f(B) \geq f(A \cup B) + f(A \cap B) \tag{2.2}$$

for any two subsets $A, B \subseteq V$. Note that the two definitions in Eq. (2.1) and (2.2) are equivalent. Below, we present Theorem 2.1.1 showing different equivalent definitions of submodular functions.

Theorem 2.1.1. *Given a submodular function $f : 2^V \to \mathbb{R}$, the following definitions are equivalent [239, 26].*

- $f(A) + f(B) \geq f(A \cup B) + f(A \cap B)$, $\forall\, A, B \subseteq V$
- $f(a|A) \geq f(a|B)$, $\forall\, A \subseteq B \subset V$ *and* $\forall a \in V \setminus B$
- $f(x|A) \geq f(a|A \cup y)$, $\forall\, A \subset V$, $\forall\, x \in V \setminus (A \cup \{y\})$
- $f(C|A) \geq f(C|B)$, $\forall\, A \subseteq B \subset V$, $\forall\, C \subseteq V \setminus B$
- $f(B) \leq f(A) + \sum_{a \in B \setminus A} f(a|A)$, $\forall\, A \subseteq B \subseteq V$
- $f(B) \leq f(A) - \sum_{a \in A \setminus B} f(a|A \setminus \{a\})$, $\forall\, B \subseteq A \subseteq V$

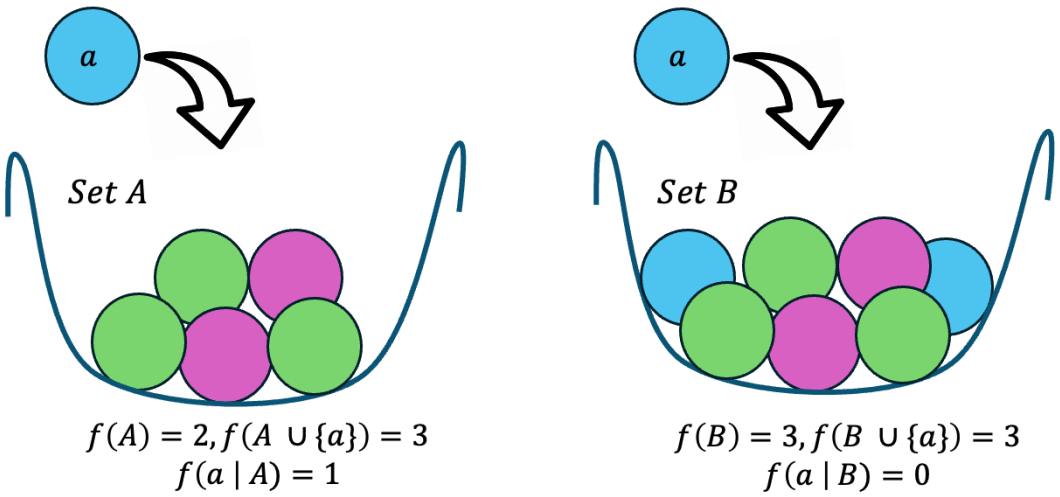


Figure 2.1: Example of a Submodular function

A submodular function is said to be normalized if $f(\emptyset) = 0$, and it is monotone non-decreasing if $f(A) \leq f(B)$, $\forall A \subseteq B \subseteq V$. Alternatively, this can be written as $f(a|A) \geq 0$, $\forall a \in V \setminus A, A \subseteq V$, meaning the conditional gain is always non-negative. A submodular function that is normalized, monotone, and non-decreasing is referred to as a polymatroid function [64].

In Fig. 2.1, we demonstrate an example of a submodular function. The set function $f(A)$ is defined as the number of unique colors in the set (or bowl) $A \subseteq V$. Bowl A on the left has only two unique colors present, so $f(A) = 2$, while bowl B on the right contains three unique colors, making $f(B) = 3$. Adding a new blue ball a to bowl A achieves a marginal gain of 1, i.e., $f(a|A) = 1$, while in the context of the larger set B, the marginal gain is zero, showing the diminishing returns property of the submodular function f.

Supermodular Functions: If a given function $f(A)$ is submodular, then $g(A) = -f(A)$ is *supermodular*, and the inequality in Eq. (2.1) is reversed. Specifically, for all $A \subseteq B \subseteq V$ and $a \in V \setminus B$, we have the following inequality (increasing returns):

$$f(A \cup \{a\}) - f(A) \leq f(B \cup \{a\}) - f(B) \tag{2.3}$$

Modular Functions: A set function $m(A)$ that is both submodular and supermodular is known as a *modular* function and the inequality in Equations (2.1) and (2.3) is changed to an equality. This implies that the marginal (or conditional) gain of an item does not depend on the conditioning set. Mathematically, modular function m is defined as follows:

$$m(A) = c + \sum_{a \in A} m(a) \;\; \forall \; A \subseteq V \tag{2.4}$$

If the modular function is normalized, i.e., $m(\emptyset) = 0$, then $c = 0$. The modular function valuation of a set can be seen as simply adding the contributions of each singleton item in the set, and hence $m(a|A) = m(a|B)$, $\forall a \in V \setminus B$ and $\forall \; A \subseteq B \subset V$.

2.1.2 Examples of Submodular Functions

Submodular functions [94] are widely recognized to model notions of diversity, representativeness, and coverage in many applications [26]. These functions satisfy the diminishing returns property; that is, the incremental benefit of adding a new element decreases as the context size increases. In this section, we cover some examples of submodular functions used in the machine learning literature.

Facility Location Function

The facility location function identifies a subset of locations A from a ground set V such that the similarity between any location in V and its closest facility in A is maximized. Mathematically, it is defined as follows:

$$f(A) = \sum_{v \in V} \max_{a \in A} \; \text{sim}(a, v) \tag{2.5}$$

where $\text{sim}(a, v)$ measures the similarity between two locations a and v. This function captures the diversity of the set A relative to V—a high $f(A)$ implies that every location in V is well-represented by at least one facility in A. Consequently, maximizing $f(A)$ under constraints

(e.g., cardinality) can yield a representative subset of V, as often required in summarization and data selection tasks.

The utility of this function for practical ML applications often relies on the choice of feature representations used to represent each item in the ground set V and the choice of the similarity kernel. Representations learned via unsupervised or self-supervised neural networks are frequently used with similarity metrics like RBF kernel and non-negative cosine similarity (via ReLU truncation or add-one transformation). Finally, tuning these hyperparameters is extremely important for optimal performance and is often dataset and task-dependent—there is no one-size-fits-all solution.

Set Cover Function

The set cover function is a classic submodular function widely used to evaluate coverage in optimization tasks. It is defined as:

$$f(A) = |\bigcup_{a \in A} S_a| \tag{2.6}$$

where S_a represents a subset of elements (or concepts $\mathcal{U}$) associated with each a in the ground set V and $A \subseteq V$. The function $f(A)$ measures the total number of unique elements covered by the subsets S_a corresponding to the selected items in A.

This function is particularly useful for selecting a subset A that provides maximum coverage of the universe of elements $S = \bigcup_{a \in V} S_a$. It is often applied in tasks such as resource allocation and document summarization, where ensuring comprehensive representation is critical.

Feature-Based Function

The feature-based submodular function is yet another submodular function for modeling diversity based on features. Given a ground set V and a set of features denoted by $\mathcal{U}$, each element $v \in V$ is associated with a non-negative feature vector $m(v) \in \mathbb{R}^{|\mathcal{U}|}_{\geq 0}$. The

feature-based function is defined as follows:

$$f(A) = \sum_{u \in \mathcal{U}} \phi_u \left(\sum_{a \in A} m_u(a) \right) \tag{2.7}$$

where $m_u(a)$ denotes the u-ness of item a or the contribution of element a to the feature $u \in \mathcal{U}$, and ϕ_u is a monotone concave function (e.g., $\log(1+x)$, $\sqrt{x}$). The composition of a monotone concave function with any non-negative modular function results in a submodular function.

Each element in the subset $A \subseteq V$ contributes to the u-ness of the overall set A in an additive manner, meaning $m_u(\cdot)$ is a modular function such that $m_u(A) = \sum_{a \in A} m_u(a)$. The concavity of ϕ_u ensures the diminishing returns property: as the subset A grows, the marginal contribution of additional elements diminishes. This property promotes diversity by rewarding subsets A that distribute feature values across $\mathcal{U}$. For instance, achieving a high function value requires leveraging features that are under-represented in A, encouraging the selection of diverse elements in terms of feature coverage and representation.

Practical implementations of this function often rely on feature representations tailored to the application. For categorical features, one-hot encoding can be used to capture diversity by ensuring a balanced representation across categories. For numerical features, clustering values into discrete categories followed by one-hot encoding provides a similar effect. Designing appropriate feature representations for this function is highly task-dependent for best performance on various ML downstream tasks.

Mixture of Submodular Functions

A mixture of submodular functions is constructed by combining multiple submodular functions into a single function using non-negative weights. Formally, given a set of submodular functions $f_1, f_2, \ldots, f_k$, the mixture function is defined as $f(A) = \sum_{i=1}^{k} \lambda_i f_i(A)$ where $\lambda_i \geq 0$ are mixture weights. This composition preserves submodularity because submodular functions are closed under non-negative linear combinations.

Additionally, combining a submodular function with a modular function yields another

submodular function. If $f_s(A)$ is a submodular function and $m(A) = \sum_{a \in A} m(a)$ is a modular function, their mixture $f(A) = f_s(A) + m(A)$ remains submodular. This combination facilitates tasks that require balancing diversity and representation (modeled by the submodular component) with individual element importance (modeled by the modular component). These formulations are particularly useful in real-world applications like query-focused summarization, where both coverage and weighted relevance of items to a given query set must be optimized.

Mixtures of submodular functions are particularly useful for modeling complex objectives that require balancing multiple criteria, such as diversity, coverage, and representation. By adjusting the weights λ_i, one can tailor the mixture function to emphasize specific submodular properties that align with different task requirements.

2.2 Submodular Optimization

2.2.1 Submodular Maximization

Submodular maximization is a fundamental problem in combinatorial optimization and is widely applicable in scenarios requiring one to maximize diversity, coverage, or representativeness. Formally, given a submodular function $f(\cdot)$, the goal is to find a subset $A \subseteq V$ that maximizes $f(A)$ under certain constraints on A, such as cardinality, knapsack, etc., as shown below:

Problem 1 (Constrained Submodular Maximization)**.**

$$\max_{A \subseteq V} f(A) \textit{ subject to constraints on } A \tag{2.8}$$

While finding the exact solution is often computationally intractable due to the combinatorial nature of the problem, efficient approximation algorithms exist.

Cardinality Constrained Submodular Maximization

Maximizing a monotone submodular function ($f(A) \geq f(B)\ \forall\ B \subseteq A \subseteq V$) without any constraints is often trivial, as the optimal solution is simply the entire ground set V. However,

real-world problems typically impose constraints that make the problem both meaningful and computationally challenging. One common and practically relevant constraint is the cardinality constraint, where the goal is to select a subset $A \subseteq V$ of limited size $|A| \leq k$ that maximizes the submodular function $f(A)$ as shown in Problem 2

Problem 2 (Cardinality Constrained Submodular Maximization)**.**

$$\max_{A \subseteq V, |A| \leq k} f(A) \tag{2.9}$$

Algorithm 1 Greedy Algorithms [239] for Problem 2

1: **Input:** Polymatroid function $f : 2^V \to \mathbb{R}_+$, cardinality constraint k
2: **Output:** Set $A \subseteq V$ maximizing $f(A)$ under cardinality constraint k
3: Initialize an empty set $A \leftarrow \emptyset$
4: **for** $j = 1$ to k **do**
5: $\quad e \leftarrow \text{argmax}_{v \in V \setminus A}(f(v|A)$
6: $\quad A \leftarrow A \cup \{e\}$
7: **end for**
8: return A

The greedy algorithm outlined in Algorithm 1 is a widely used approach for solving this problem. Starting with an empty set, the algorithm iteratively adds the element that provides the largest marginal gain to the function value, continuing until the subset reaches the cardinality limit k. For monotone submodular functions, the greedy algorithm achieves a $(1-1/e)$-approximation, where e is the base of the natural logarithm, guaranteeing solutions that are provably close to the optimal.

Theorem 2.2.1. *[239] For a given monotone non-decreasing submodular function f, Algorithm 1 produces a solution S that satisfies the following approximation guarantee:*

$$f(S) \geq (1 - \frac{1}{e}) f(S^*) \approx 0.63 f(S^*) \tag{2.10}$$

where $S^* \in \text{argmax}_{|A| \leq k, A \subseteq V} f(A)$ is the optimal solution to Problem 2.

Cardinality-constrained submodular maximization has numerous practical applications, including summarization, data subset selection, and resource allocation. For instance, in summarization tasks, the goal is to select a limited number of sentences or data points that maximize information coverage. Similarly, resource allocation involves selecting a subset of locations or items that best represent or serve a target population within budget constraints. The combination of submodularity and cardinality constraints makes this formulation both versatile and effective in addressing diverse optimization challenges.

The greedy algorithm (Algorithm 1) finishes in k iterations, selecting at each step the element with the highest marginal gain conditioned on the previously selected subset. This leads to a time complexity of $O(nk)$, which scales to $O(n^2)$ when k is proportional to n.

To reduce the computational cost, the *lazy greedy algorithm* [231] leverages the diminishing returns property of submodular functions to avoid redundant evaluations. Instead of evaluating all elements at each iteration, the algorithm maintains a priority queue of marginal gains. At each step, it selects the top element from the queue and evaluates its true gain conditioned on the current subset. If the true gain remains the highest, the element is selected; otherwise, it is returned to the queue, and the process continues. Although the worst-case complexity remains the same as the vanilla greedy algorithm, this approach using a priority queue data structure significantly reduces computation in practice.

Further optimization strategies include approximate greedy algorithms. The *threshold-based greedy algorithm* [17] achieves an approximation ratio of $(1 - 1/e - \epsilon)$ with $O\left(\frac{n}{\epsilon} \log \frac{n}{\epsilon}\right)$ function evaluations. Here, ϵ is a parameter controlling the trade-off between approximation quality and computational efficiency. The algorithm iteratively reduces a threshold exponentially and adds all elements with marginal gains exceeding the threshold.

Similarly, the *lazier-than-lazy greedy algorithm* [234] uses random sampling to limit the search space, achieving the same approximation ratio with $O(n \log \frac{1}{\epsilon})$ function evaluations. Specifically, it uses uniform sampling to select a subset of elements at each iteration and evaluates the marginal gains only for this subset, reducing computation. The size of the sampled subset is controlled by the parameter ϵ, allowing further tuning of the trade-off

between computational efficiency and approximation quality.

Knapsack Constrained Submodular Maximization

Knapsack-constrained submodular maximization generalizes the cardinality constrained problem by assigning a weight $w(v)$ to each element $v \in V$ and imposing a budget b such that the total weight of the selected subset does not exceed b. Here, $w(\cdot)$ is modular function meaning $w(A) = \sum_{v \in A} w(v)$. Formally, the goal is to find a subset $A \subseteq V$ that maximizes a monotone submodular function $f(A)$ under the constraint $\sum_{v \in A} w(v) \leq b$ as stated below:

Problem 3 (Knapsack Constrained Submodular Maximization)**.**

$$\max_{A \subseteq V, w(A) \leq b} f(A) \tag{2.11}$$

Algorithm 2 Greedy Algorithm [151, 138] for Problem 3

1: **Input:** Ground set V, budget b, cost function $w : V \to \mathbb{R}_+$, polymatroid function $f : 2^V \to \mathbb{R}_+$
2: **Output:** Set $A^{'} \subseteq V$ maximizing $f(A^{'})$ under budget constraint $w(A^{'}) \leq b$
3: Initialize an empty set $A \leftarrow \emptyset$
4: **while** $w(A) < b$ **do**
5: $\quad e \leftarrow \text{argmax}_{v \in V \setminus A, w(A \cup \{v\}) \leq b} \frac{f(A \cup \{v\}) - f(A)}{w(v)}$
6: $\quad A \leftarrow A \cup \{e\}$
7: **end while**
8: Let $e^* \leftarrow \text{argmax}_{v \in V, w(v) \leq b} f(\{v\})$
9: **return** $A^{'} = \text{argmax}_{S \in \{A, \{e^*\}\}} f(S)$

When each element has an identical cost, the budget constraint simplifies to a cardinality constraint, under which the greedy algorithm achieves a $(1 - \frac{1}{e})$-approximation guarantee.

The greedy algorithm can be adapted to the knapsack setting by iteratively selecting the element with the highest marginal gain per unit weight, $\frac{f(a|A)}{w(a)}$ until the budget is exhausted.

This approach as shown in Algorithm 2 achieves a $(\frac{1-e^{-1}}{2})$-approximation for monotone submodular functions as stated below:

Theorem 2.2.2. *[151] For a given monotone non-decreasing submodular function f, Algorithm 2 produces a solution S that satisfies the following approximation guarantee:*

$$f(S) \geq \frac{1-e^{-1}}{2} f(S^*) \tag{2.12}$$

where $S^* \in \text{argmax}_{A \subseteq V, w(A) \leq b} f(A)$ is the optimal solution to Problem 3.

Algorithm 3 Modified Greedy Algorithm [194] for Problem 3

Input: Ground set V, budget b, cost function $w : V \to \mathbb{R}_+$, polymatroid function $f : 2^V \to \mathbb{R}_+$, scaling factor $r > 0$
Output: Set $A^{'} \subseteq V$ maximizing $f(A^{'})$ under budget constraint $w(A^{'}) \leq b$
Initialize an empty set $A \leftarrow \emptyset$
Initialize remaining set $R \leftarrow V$
while $R \neq \emptyset$ **do**
  $e \leftarrow \text{argmax}_{v \in R} \frac{f(A \cup \{v\}) - f(A)}{(w(v))^r}$
  if $\sum_{a \in A} w(a) + w(e) \leq b$ **and** $f(A \cup \{k\}) - f(A) \geq 0$ **then**
    $A \leftarrow A \cup \{e\}$
  end if
  $R \leftarrow R \setminus \{e\}$
end while
$e^* \leftarrow \text{argmax}_{v \in V, w(v) \leq b} f(\{v\})$
return $A^{'} = \text{argmax}_{S \in \{A, \{e^*\}\}} f(S)$

Lin and Bilmes [194] proposed a modified version of the greedy algorithm as described in Algorithm 3 by introducing a scaling factor $r > 0$. By introducing r, the algorithm provides a flexible mechanism to balance the trade-off between the function marginal gain and item cost. When $r > 1$, the algorithm imposes a stronger penalty on high-cost items, leading

to a more conservative selection strategy that favors lower-cost items. On the other hand, when $r < 1$, the impact of cost-based penalization is diminished, allowing the algorithm to prioritize elements with higher marginal gains, even when their associated costs are relatively high. r can be tailored based on domain knowledge. For example, if costs vary widely and high-cost elements are disproportionately impactful, reducing r could be advantageous.

Theorem 2.2.3. *[194] For a normalized monotone submodular function f, Algorithm 3 with $r = 1$ provides the following approximation guarantee:*

$$f(S) \geq \left(1 - e^{-1/2}\right) f(S^*) \tag{2.13}$$

where S^* is the optimal solution to Problem 3.

Algorithm 2 is a special case of Algorithm 3 for $r = 1$, with the latter achieving a better bound. Introducing the scaling factor r provides a mechanism to control the trade-off between cost efficiency and marginal gain, making the greedy algorithm more adaptable to problem-specific requirements compared to the original greedy approach in Algorithm 2.

Knapsack constraints are highly relevant in applications where resources are limited. For instance, in data subset selection, $w(v)$ may represent the cost of processing or labeling an item, and the budget b corresponds to the total available resources. Similarly, in sensor placement, $w(v)$ could denote the cost of deploying a sensor, while b represents the deployment budget. These practical scenarios highlight the importance of efficient and scalable solutions for performing knapsack-constrained submodular maximization.

2.3 Submodular Information Theoretic Functions

Information-theoretic measures [59], such as mutual information (MI) and conditional entropy, have long been central to machine learning [91, 249, 280, 85, 359] and data-driven decision-making [318, 310, 118], offering a principled framework for quantifying relevance, redundancy, and diversity. These measures have been applied in continuous domains, where they quantify dependencies between random variables, enabling optimization in tasks such

as feature selection, active learning, clustering, etc. However, many practical machine learning scenarios involve finite, discrete datasets where the relationships between data subsets must also capture diversity and representativeness. Submodular functions, with their diminishing returns property, naturally extend these continuous information-theoretic concepts to the discrete domain. Submodular information-theoretic measures [125] such as Submodular Mutual Information (SMI) and Conditional Gain enable efficient selection of diverse and relevant data subsets, making them invaluable in addressing challenges of large-scale machine learning, including data summarization, active learning, and resource-efficient model training.

2.3.1 Submodular Mutual Information (SMI)

Submodular Mutual Information (SMI) extends the notion of MI to discrete settings, enabling the evaluation of the relevance of a subset $A \subseteq V$ with respect to a query set $Q \subseteq V$. Formally, it is defined as:

$$I_f(A; Q) = f(A) + f(Q) - f(A \cup Q) \tag{2.14}$$

where $f : 2^V \rightarrow \mathbb{R}$ is a submodular function. The SMI function quantifies the shared information between A and Q, effectively measuring their similarity (or relevance). Here, Q is referred to as the query or conditioning set, representing the anchor set for selection. Maximizing the SMI function helps identify a subset A that not only aligns closely with Q (or is redundant with respect to Q) but also maintains desirable properties such as representation, coverage, and diversity, depending on the underlying submodular function. This formulation is particularly useful in data selection tasks where the goal is to choose subsets that are both diverse and highly relevant to a specific query set or task.

For example, in query-focused, that is, targeted data selection/summarization, A might represent a candidate subset of data points, and Q is a set of labeled examples or key features. The SMI ensures that the selected subset A captures the most important aspects

of Q, balancing coverage, diversity, and relevance.

2.3.2 Submodular Conditional Gain

The conditional gain extends the notion of conditional entropy to submodular functions, measuring the incremental value of adding a subset A to another subset Q. It is defined as:

$$f(A \mid Q) = f(A \cup Q) - f(Q) \tag{2.15}$$

where $f : 2^V \rightarrow \mathbb{R}$ is a submodular function and Q is referred to as the conditioning set. Conditional gain captures the marginal utility or contribution of including A in the context of Q. Intuitively, it models the additional information A brings when combined with the already existing set Q.

Maximizing the conditional gain is particularly useful in scenarios where the goal is to identify a subset A that not only complements Q but is also diverse and representative. For example, in data selection tasks, Q might represent a set of data points already selected or labeled, and A represents new candidates being evaluated for inclusion or annotation. By maximizing the conditional gain, the selected subset A ensures minimal redundancy with respect to Q while maintaining representativeness.

Conversely, minimizing the conditional gain can also be useful in specific contexts. By minimizing $f(A \mid Q)$, the objective is to identify subsets A that closely align with Q, capturing elements that exhibit high redundancy or similarity to Q. This is particularly valuable in query-relevance tasks, where the aim is to find subsets A that provide the most overlap with or relevance to the conditioning set Q. For example, in query-focused summarization, minimizing the conditional gain helps select elements that fully cover the aspects represented by the query set Q but might contain redundant elements, as we highlight in Chapter 4.

Therefore, conditional gain is a versatile framework that supports both complementarity and redundancy-based selection, depending on the specific optimization objective. By leveraging the diminishing returns property of submodular functions, conditional gain provides

a principled approach to quantifying the incremental contributions of subsets A in the presence of Q. This makes it applicable to a wide range of machine learning tasks, such as data selection, summarization, active learning, and diversity-aware optimization.

Chapter 3

DATA SELECTION AND AUGMENTATION

In this chapter, we provide a broad overview of existing literature on data selection (Sec. 3.1) and augmentation (Sec. 3.2) techniques, with a particular focus on approaches that enhance the resource efficiency of machine learning models. We categorize data selection algorithms across three key ML paradigms: (1) active learning (Sec. 3.1.1), (2) targeted fine-tuning (Sec. 3.1.2), and (3) test-time input context selection for efficient inference (Sec. 3.1.3). While this chapter provides a high-level survey of related work, each subsequent chapter contains a dedicated related work section, where we delve deeper into the specific challenges and methodologies relevant to the specific task and application area covered in that chapter.

3.1 Data Selection in Deep Learning

Data selection is a fundamental component in the training of deep learning models [383, 203], where, given a collection of data points, the goal is to identify a subset that is optimal in terms of improving model performance, resource efficiency, and generalization ability. The primary objective is to ensure that the data utilized for training is both relevant and of high quality [57]—this becomes even more critical as datasets grow in size, complexity, and heterogeneity. Data selection approaches range from heuristic techniques that apply predefined filtering rules [266, 168] to more advanced algorithmic methods that evaluate and prioritize data points based on their relevance and informativeness [362, 235, 140, 139].

Heuristic approaches typically rely on straightforward filtering rules, such as excluding noisy, incomplete, or low-quality samples, aiming to improve the overall quality of the dataset [260, 263, 165]. These methods are often easy to implement and are computation-

ally efficient, making them well-suited for initial dataset refinement. However, their reliance on simple criteria can cause them to overlook intricate data relationships and fail to identify complex redundancies, ultimately limiting their effectiveness in improving the resource efficiency of ML models, particularly for large-scale datasets.

In contrast, algorithmic approaches offer a more structured framework for data selection by leveraging optimization techniques to assess and prioritize samples based on their relevance and informativeness. Methods such as GLISTER [140] use bi-level optimization to identify data subsets that maximize validation performance while maintaining robustness to noise and imbalances in the dataset. Similarly, GRAD-MATCH [139] uses gradient matching to select subsets that replicate the learning signal of the entire dataset, achieving an effective balance between computational efficiency and predictive accuracy. These methods have been shown to significantly reduce training times and computational costs while maintaining overall model performance.

Another set of techniques involves coreset selection, such as CRAIG [235] and CREST [363]. These approaches identify representative weighted subsets of data that closely approximate the gradient of the full dataset, ensuring theoretical convergence guarantees. CRAIG, for instance, maximizes a submodular function to select subsets, providing efficient training without compromising accuracy. CREST extends this concept to non-convex models by iteratively updating subsets during training to adapt to evolving data dynamics, thereby improving scalability and sustainability [363].

3.1.1 Data Selection for Active Learning

Another data selection paradigm, active learning [291], focuses on iteratively querying the most informative samples for annotation. This enables the efficient use of labeled data in scenarios where labeling costs are prohibitively high. Traditional methods, such as uncertainty sampling and query-by-committee, select samples based on model uncertainty or disagreement among predictors [291]. Although effective in certain scenarios, these approaches often neglect diversity within the selected batches [346], leading to redundant samples and thus

limiting overall improvements in terms of resource efficiency.

To address these limitations, diversified active learning methods have been proposed to balance informativeness and diversity in the selection process. For instance, Filtered Active Submodular Selection (FASS) [346] combines uncertainty sampling with submodular optimization to select subsets that maximize both diversity and informativeness, leading to improved generalization. Similarly, Sener and Savarese [289] reformulate active learning as a coreset selection problem, leveraging the k-center algorithm to identify data points that are most representative of the dataset. The BADGE algorithm [15] extends these ideas by incorporating hypothesized gradients and selecting batches with diverse and high-magnitude gradient directions. By capturing both predictive uncertainty and sample diversity, BADGE provides a robust approach to active learning, particularly in resource-constrained scenarios where labeled data is limited. An alternative to traditional active learning methods is Selection via Proxy (SVP) [57], which employs smaller and computationally efficient proxy models to guide the selection process. These proxy models approximate the learning signals of larger, more complex models, thereby significantly reducing computational overhead without compromising model accuracy.

3.1.2 Data Selection for Targeted Fine-tuning

Data selection for targeted fine-tuning involves selecting a subset of data most relevant to a specific task or domain from a larger dataset, aiming to reduce computational costs while maintaining or improving model performance. This approach has gained significant attention with the increased usage of pre-trained large language models (LLMs), often pre-trained on vast and heterogeneous datasets. Fine-tuning these large models for diverse downstream tasks or applications is computationally expensive and can lead to negative task interference [122]. By carefully selecting task-relevant subsets, targeted fine-tuning reduces resource costs and leads to better task-specific generalization, providing an efficient framework for adapting pre-trained large models to specialized domains.

The most common set of methods employs similarity-based retrieval to identify task-

relevant data from large datasets encompassing a diverse range of tasks. The DEFT framework [122] leverages cross-task nearest neighbors to identify subsets of multi-task data closely aligned with the target task data points. DEFT operates by analyzing a small set of unlabeled examples from the target task to identify similar instances within the large pool of multi-task data, creating a curated subset for fine-tuning. By fine-tuning on these targeted subsets, DEFT achieves competitive or superior performance compared to models trained on the entire dataset, demonstrating the efficacy of targeted data selection. Similarly, LESS [353] selects targeted fine-tuning data by leveraging low-dimensional gradient feature similarity to estimate data influence effectively. These works demonstrate the importance of fine-grained data selection in order to achieve task-specific performance gains and resource efficiency concerning data, computing, and memory.

Another group of methods employs submodular optimization to systematically select diverse and informative subsets of data, providing a principled approach for targeted data selection. For example, DELIFT [2] optimizes data selection across various stages of fine-tuning—such as instruction tuning, task-specific fine-tuning, and continual fine-tuning —using a pairwise utility metric and submodular optimization to identify diverse, high-value data subsets. This approach leads to substantial reductions in dataset size, often up to 70%, thereby reducing resource requirements while maintaining model performance. Similarly, DITTO [150] uses Submodular Mutual Information (SMI) [125] functions (described in Sec. 2.3.1) to identify the most informative subsets of data for adapting speech recognition systems to target accents. By maximizing SMI, DITTO ensures that the selected data is representative of the target domain and diverse enough to generalize across multiple target accents, making it particularly effective in low-resource settings with limited labeled data.

Another group of methods leverages target task-specific retrieval augmentation to enhance data selection for targeted fine-tuning while maintaining computational efficiency. COBRA [68] introduces an SMI-based (described in Sec. 2.3.1) retrieval strategy to improve data selection for targeted fine-tuning. Unlike traditional retrieval approaches that rely solely on similarity measures, COBRA optimizes for both relevance and diversity using

Facility Location Mutual Information (FLMI), an SMI function, ensuring the selected few-shot examples cover the target task space well. This improves the data efficiency of few-shot learning by reducing redundancy in the selected data, making it particularly effective in low-data regimes.

3.1.3 Test-Time Input Context Selection

LLMs [258, 321] and Multimodal Large Language Models (MLLMs) [371, 314, 198, 337] have demonstrated state-of-the-art performance across a wide range of natural language processing (NLP) and computer vision (CV) tasks, including machine translation, document summarization, conversational AI, image captioning, and video understanding [36, 81, 344, 192]. Their ability to adapt across various downstream applications [259, 257, 371] has made them a core component of modern ML systems. However, their substantial computational and memory demands—stemming from their large parameter sizes, quadratic scaling of the self-attention [328] mechanism, and the need to handle long input contexts—pose significant challenges for practical deployment. As LLMs and MLLMs are increasingly used in real-time and resource-constrained settings, improving scalability and efficiency during inference has become a critical area of research [31, 333].

LLMs and MLLMs perform downstream tasks primarily through prompting, where the task specification and relevant data are provided as part of the input context, allowing the model to generate appropriate responses without explicit fine-tuning [176, 304, 344, 315, 78, 97, 50, 217]. These input contexts can be thousands of tokens long, particularly when models are used to process long-form documents such as legal or scientific texts and conversation histories or when augmented with external knowledge retrieval from search engines or databases [358, 264]. Similarly, in MLLMs, input contexts can be composed of long video sequences or multiple image frames [308], further exacerbating the computational overhead. Since transformer-based architectures exhibit quadratic scaling in compute and memory costs with respect to input length [328], selecting the input context efficiently is a fundamental challenge for scalable inference.

Test-time input context selection involves identifying and prioritizing the most relevant information to be provided to the model during inference. Filtering out redundant and irrelevant data helps reduce computational and memory overhead while maintaining or even improving model performance. This is particularly crucial in real-time and resource-constrained environments where inference must meet strict latency and efficiency requirements [164, 5, 372].

Importance of Input Context Selection The quality and relevance of input context play a critical role in the inference performance of LLMs and MLLMs. Providing redundant and irrelevant information increases the computational workload significantly due to the quadratic scaling of self-attention mechanisms in transformer-based architectures [328]. As input length increases, the self-attention operation requires computing pairwise interactions between all tokens, leading to $O(n^2 d)$ complexity, where n is the sequence length and d is the model's hidden dimension. This results in disproportionately higher computational costs, longer inference times, and increased memory consumption, making the efficient selection of relevant input context essential for scalable deployment [164, 298, 253, 372]. More importantly, irrelevant context can introduce noise into the model's processing, potentially degrading the quality of predictions by misguiding attention allocation, thereby affecting the model's ability to focus on task-critical information [300, 368, 173].

Conversely, selecting a task-relevant and diverse input context enables the model to focus on the information most pertinent to the task, improving both efficiency and interpretability. For example, in question answering or document summarization, providing concise and focused context allows the model to generate more accurate and coherent responses [176, 254, 128, 373].

Input Context Selection Across ICL, RAG, and KV Cache Summarization: Three major areas where input context selection plays a crucial role in optimizing inference efficiency are In-Context Learning (ICL), Retrieval-Augmented Generation (RAG), and KV cache summarization. These are described below.

In-Context Learning

In-context learning is a powerful paradigm that enables pre-trained models to perform new tasks by conditioning on task demonstrations (also called exemplars) provided within the input context [36, 344]. Without any fine-tuning, the model derives task-specific behaviors from exemplars included in the input prompt, allowing for adaptation to diverse tasks.

Since the choice of exemplars directly impacts downstream task performance, effective input context selection is crucial in in-context learning [78]. To ensure that the model captures key task characteristics, one needs to prioritize both query-relevant and diverse demonstrations for selection in the prompt [174, 142, 366], leading to reliable outputs while incurring low latency. Methods such as submodular optimization [109, 157] and relevance-based filtering [264, 220] have been employed to identify query-relevant and representative exemplars, maximizing the utility of constrained context windows.

Retrieval Augmented Generation

Retrieval-Augmented Generation (RAG) integrates external knowledge retrieval into the inference process, enabling LLMs and MLLMs to access domain-specific or up-to-date information. In this framework, relevant documents or data points are retrieved from large knowledge bases and included in the input context, enriching the model's understanding and improving generation quality [176, 126, 179, 264].

Effective context selection is crucial in RAG systems to prevent the inclusion of irrelevant or redundant information, which can increase the sequence length and increase computational and memory overhead. Processing longer sequences not only raises memory and latency costs but can also degrade model performance by introducing distracting or conflicting information [342, 226]. To address this, techniques such as semantic similarity-based selection [264, 97, 88] and diversity-based selection [252, 46] are commonly employed to identify and retrieve the most informative and non-redundant documents. By prioritizing relevant and informative data and filtering out redundant content, RAG systems improve

response quality while minimizing computational costs.

KV Cache (memory) selection in LLMs & MLLMs

The decoding process in LLMs and MLLMs follows an autoregressive mechanism, where tokens are processed sequentially to generate output tokens. At each step, the model attends to previously generated tokens in addition to the input tokens using the self-attention mechanism, enabling contextual reasoning over long sequences. In LLMs, the sequence consists of text tokens, while in MLLMs, it includes both text tokens and visual tokens [198] with visual token embeddings extracted from vision encoders such as CLIP [257]. To avoid redundant re-computations of past tokens at each generation step, KV (Key-Value) caching is employed [253]. KV caching stores previously computed attention keys (K) and values (V) across all layers, allowing the model to efficiently reuse past token representations in subsequent self-attention computations [328].

While KV caching eliminates redundant re-computations, the quadratic complexity of the self-attention mechanism with respect to the sequence length remains a bottleneck. Specifically, the computational and memory overheads scale as $O(N^2d)$, where N is the sequence length and d is the embedding dimension. As the sequence length increases, inference latency and memory consumption grow non-linearly, posing challenges for real-time applications [67, 5, 372]. Furthermore, the KV cache memory footprint increases linearly with sequence length, as each token requires storing its K and V representations across all model layers. In LLMs processing long documents [197] and MLLMs handling long video sequences [308, 113], this quickly leads to GPU memory saturation, significantly limiting scalability. Thus, efficiently managing the KV cache, or more broadly, the memory footprint of LLMs and MLLMs, is crucial for ensuring the scalability of these models in real-world applications.

Several KV cache summarization techniques have been proposed to mitigate the growing memory overhead in LLMs by selectively evicting, compressing, or quantizing tokens. H2O [393] identifies and retains heavy-hitter tokens based on a modular scoring function,

while FastGen [100] leverages attention profiling to determine which tokens can be evicted without hurting downstream task performance. Prompt compression methods [236] generate gist tokens to reduce KV cache size while preserving essential context. Post-training quantization techniques further optimize memory usage, with WKVQuant [376] applying a past-only quantization mechanism, QAQ [79] selectively quantizing elements based on attention scores, and Sparq [274] pruning tokens while correcting errors using sparse matrices. GEAR [133] approximates quantization errors using low-rank matrices, whereas Loma [339] introduces a lossless compression approach, though it requires fine-tuning. Additionally, PagedAttention [163] optimizes memory fragmentation in order to enhance the throughput of LLM serving. These methods collectively aim to reduce LLMs' KV cache size, improving efficiency in long-sequence processing and inference.

In large multimodal models, the problem becomes even more pronounced, as each image frame typically consists of over 600 visual tokens when using Vision Transformers (ViTs) to extract patch (or visual token) embeddings [257]. In MLLMs such as Video-LLaVA [192], where multiple frames are processed sequentially, the KV cache grows rapidly with each new frame, adding thousands of visual tokens to the sequence. As a result, effective visual token summarization is essential to maintaining computational efficiency and reducing memory footprint. To address this, methods such as FasterVLM [388] leverage self-attention-based token filtering to remove redundant visual tokens before they are processed by the LLM, while VisionZip [361] employs token merging techniques to condense multiple similar tokens into a smaller set of representative embeddings. These strategies enable VLMs to process long multimodal sequences while balancing computational & memory efficiency and model performance, ensuring that memory and compute constraints in real-world applications do not hurt accessibility.

3.2 Data Augmentation in Deep Learning

Modern ML systems using deep models have achieved significant success across diverse tasks [255] by leveraging their ability to extract complex patterns from large datasets. How-

ever, their effectiveness depends on their generalizability—the ability to perform well on unseen data beyond the training set. Generalization remains a central challenge in deep learning, as increasing model complexity can sometimes lead to poor performance on unseen test data [23]. While highly expressive models have the capacity to fit training data closely, this does not always translate into improved test performance [237]. The precise mechanisms by which deep models generalize—despite their ability to fit even random labels—remain an active area of research [241, 325, 23, 384]. Nevertheless, empirical evidence suggests that under certain conditions, overfitting can limit a model's applicability in real-world scenarios [268].

Techniques ranging from regularization [210, 92, 395] to architectural modifications [309, 121, 114, 16] have been proposed to mitigate overfitting and improve model generalization. Among these, data augmentation [383] stands out as a simple yet effective approach where the training dataset is artificially expanded by applying transformations such as rotations, flips, noise injection, or domain-specific modifications. These augmentations increase data diversity [340]—by introducing novel variations of existing samples—and encouraging the model to learn more robust and generalizable features, thereby reducing its dependence on memorizing specific examples. In this subsection, we focus on data augmentation techniques and their role in improving generalization in deep models, particularly in the limited data regime.

3.2.1 Data Augmentation for Image Data

In low-data settings for different vision-based tasks, where the training dataset size is insufficient to fully capture the variability of the underlying distribution, image data augmentation is a key technique for improving model performance while maintaining the same data annotation costs. Basic augmentation methods [303], such as geometric transformations (e.g., flipping, cropping, rotation, translation) and color space adjustments (e.g., brightness and contrast modifications, color histogram alterations), are widely used due to their simplicity and effectiveness. Geometric transformations help mitigate positional biases in datasets,

while color space transformations address lighting variations. However, these techniques do not always preserve labels, and their application requires domain expertise to avoid adding noise to the training data.

Beyond these basic techniques, more advanced augmentation methods generate synthetic data by manipulating samples at both pixel and feature-space levels. For instance, MixUp [385] and Manifold MixUp [329] generate synthetic data by combining multiple images or interpolating feature representations within the model's latent space. Cutout [74] enhances robustness by removing random image regions, encouraging the model to focus on broader contextual features. CutMix [378] extends this approach by replacing the removed regions with patches from another image and mixing their corresponding labels. Deep learning-based augmentation strategies, such as GAN-based augmentation [104, 76, 35] and neural style transfer [98], further increase dataset diversity by synthesizing realistic training examples, albeit at a higher computational cost. These augmentation techniques are particularly beneficial in data-scarce scenarios, such as long-tailed learning [401] and continual learning [7, 21, 130, 390, 158] with limited memory buffer (used for experience replay), etc. Additionally, reinforcement learning-based approaches, such as AutoAugment [62] and RandAugment [63], aim to automate augmentation strategies by learning optimal transformation policies, further enhancing model generalization while reducing manual effort.

3.2.2 Applications in Continual Learning

Continual Learning aims to learn continually from a stream of non-stationary data, but it is inherently prone to catastrophic forgetting [93, 223]. Several approaches have been proposed to mitigate forgetting. These include regularization-based methods [144], which constrain parameter updates to preserve previously learned knowledge; parameter isolation techniques [341], which assign dedicated subsets of parameters to different tasks; and replay-based methods [45], which revisit data from past tasks to mitigate forgetting. Among these strategies, experience replay—utilizing a fixed-size memory buffer—is one of the most effective techniques. By replaying stored samples from past tasks alongside new data, experience

replay helps maintain a balance between acquiring new information and retaining old knowledge.

However, the utility of experience replay is limited by the size of the memory buffer, which often limits the diversity and representativeness of the stored samples. Augmentation techniques such as cropping, rotation, flipping, and noise injection can enhance the replay buffer by introducing additional variability [21, 390], thereby improving the model's ability to generalize across sequential tasks. More advanced methods, such as MixUp and CutMix, further enhance the diversity of replayed data by interpolating features or blending image patches to create virtual replay samples.

Chapter 4

SUBMODULAR SPAN PROBLEM & CONDITIONAL DATA SUMMARIZATION

4.1 Introduction

In this chapter, we propose a submodular framework for conditional i.e., query-focused data summarization. Conditional data summarization focuses on selecting a subset from a large dataset that is both relevant to a given query set and representative of the large dataset. Multiple applications in machine learning and information retrieval are related to this task. For example, in query-based extractive Multi-Document Summarization (MDS), the goal is to produce a short human-readable summary given a large collection of text documents. This summary should not only be relevant to the query but also encapsulate the general content of all documents. Similarly, query-based image summarization aims to retrieve a subset of diverse images similar to the query images, given a large image dataset. This concept also applies to general web searches, where the end result is a succinct summary of the web that is relevant to the user query(s).

Conditional data summarization is related to generic summarization (where there is no query to which relevance is preferred) and is formulated as selecting a representative subset of a large dataset, often based on maximizing a utility function. The utility function captures properties such as informativeness, diversity, and coverage and often satisfies a submodularity property. Submodular functions possess a natural diminishing returns property, i.e., the incremental value of a new element is less in a larger than in a smaller context. Given a submodular function f, generic summarization can be addressed via cardinality constrained submodular maximization $\max_{A \subseteq V : |A| \leq k} f(A)$, which is solvable with a constant factor i.e., $(1 - 1/e)$ guarantee using the simple greedy algorithm [239, 231] outlined in Algorithm 1.

In ML and fields such as natural language processing (NLP) and information retrieval (IR), various approaches have been used to solve this problem. Query-based MDS can be either supervised where labels are available and a training phase occurs [195, 196] or unsupervised, where there are no target labels to train on [115, 365, 90]. Recent methods in query-based extractive video summarization include snippet selection using sequential and hierarchical Determinantal Point Processes (DPP) [295, 296]. Although applicable, these methods are supervised and consider the query to be extraneous to the data/corpus. Given the tremendous growth in data and expensive task-based data annotation, there is a pressing need for a unifying conditional data summarization framework that (a) generalizes to different queries, i.e., query-independent function formulation, (b) supports multiple query summarization, i.e., does not limit the query size to one, (c) considers the query to be intrinsic to the data/corpus, and (d) is minimally supervised, i.e., uses pre-existing summarization labels only on a limited validation set for hyperparameter tuning.

On this last point, extractive summarization labeling tasks are much harder than standard machine learning labeling and/or annotation tasks—the reason is that a training "set" must be of the form $\mathcal{D} = \{(V_i, A_i)\}_{i=1}^{l}$ where $\forall i$, V_i is the i^{th} dataset and $A_i \subseteq V_i$ is a summary of that dataset. For a human annotator, creating this is extremely difficult; imagine, for example, the task of selecting a size-1000 representative subset from 100,000 images, i.e., where $|V_i| = 100,000$ and $|A_i| = 1000$. Hence, minimal supervision (if any at all) is not only desirable but necessary for the general task of training or tuning big data summarization processes.

In this chapter, we develop a new minimally supervised conditional summarization framework based on a method that we call the *submodular span problem*. This method produces a *conditional summary*, i.e., a summary that is relevant to a given query set $Q \subseteq V$. We formulate this as an optimization problem over a submodular function $f : 2^V \to \mathbb{R}$, where the ground set V involves both the query set Q and the data items being summarized $V \backslash Q$. The utility function f is expected to capture the same fundamental properties that a utility function would capture for a generic summary (i.e., diversity, representativeness, etc.). Also, the utility function does not need to be reformulated as the query set Q changes.

This chapter is based on the work published in: **Lilly Kumari** and Jeff Bilmes. *Submodular span, with applications to conditional data summarization. In Proceedings of the AAAI Conference on Artificial Intelligence, 2021.*

4.2 Related Work

4.2.1 Conditional Multi-Document Summarization

The majority of existing extractive MDS methods are based on two tasks: query-based relevance ranking and sentence saliency score-based selection. One of the earlier standard methods is maximum marginal relevance (MMR) [40], which uses a greedy approach to select the most relevant sentences while considering the relevance and redundancy trade-offs. Follow-up works [224, 101] propose an optimal reformulation to the MMR framework in the form of an integer linear programming problem.

The methods based on data reconstruction, for example, DSDR [115], reconstruct each sentence by a non-negative linear combination of summary sentences and then use sparse coding to select summary sentences that minimize the document reconstruction error. SpOpt [365] adds a sentence dissimilarity term to the objective to maximize diversity. DocRebuild [216] further builds upon the DSDR framework using a neural document model. CTSUM [332] utilizes several hand-crafted features to predict sentence uncertainty scores and then uses them in a graph-based ranking scheme. More recently, deep learning-based techniques such as DocEmb [146], and the vector space model [132] utilize the sum of trained word embeddings to represent sentences or documents and formalize the task as maximizing a submodular function defined on the similarity of embeddings. The state-of-the-art unsupervised method called Dual-CES [278] proposes a two-step dual-cascade optimization framework, where both steps utilize the cross-entropy method to handle trade-offs between sentence saliency and focus. Among the supervised methods, the state-of-the-art method SRSum [271] uses a deep neural network-based model that comprises five sub-models: PriorSum, CSRSum, TSRSum, QSRSum, and SFSum. The individual models encode surface

features and latent semantic sentence meaning and use attention to simulate the context-aware reading of a human.

4.2.2 Conditional Video Summarization

Existing methods for this task are supervised in terms of using the summarization labels. Recent works [295, 296] based on DPP propose a sequential and hierarchical DPP to model a shot's relevance to the given query and representativeness in the video. Specifically, they use a Hierarchical Variational Network (HVN) consisting of a query-focused attention module and a multi-level self-attention variational block that captures the multi-level visual content of the scenes and adds to the user-oriented diversity as well. Another work [356] trains a Query-biased Self-Attentive Network (QSAN), which learns the mapping between the visual content and textual captions. It is then augmented with a query-aware scoring MLP to generate a query-focused summary.

4.2.3 Conditional Image Corpus Summarization

This domain is relatively new, and the existing work does not meet all the requirements of a conditional image summarization system. For example, authors in [323] propose learning a mixture of submodular functions for generic image collection summarization. Another work [13] focuses on image retrieval given multiple queries of the same object, resulting in improved recall of the system compared to a single query.

Although the existing methods perform well in their respective domains, there is no simple, effective, and unifying framework for conditional data summarization that requires minimal learning and can be used irrespective of the data modality. We believe the submodular span approach we present in this work fits this bill.

4.3 Submodular Span

A given set function $f : 2^V \rightarrow \mathbb{R}_+$ is non-negative, monotone, non-decreasing, and submodular if $f(j|A) \geq f(j|B) \geq 0$ for all $A \subseteq B \subseteq V$ and $j \in V$. Such a function is often called

a *polymatroid* function [64]. Given a polymatroid function $f : 2^V \to \mathbb{R}_+$, and a query set $Q \subseteq V$ and defining $V_Q = V \setminus Q$, we define the *submodular span problem* as

$$\text{maximize } \{|A| \text{ s.t. } A \subseteq V_Q, f(A|Q) \leq \epsilon\}, \tag{4.1}$$

where $\epsilon \geq 0$ is small. W.l.o.g., we assume all polymatroid functions are normalized so that not only $f(\emptyset) = 0$ but $f(V) = 1$. Dual to the submodular span problem is Eq. (4.32). We see that these problems are related in that they generally ask for large sets A that have low f-valuation when conditioned on the query set Q. We also see that the dual form is cardinality-constrained submodular minimization, a problem that is known to have no constant factor approximation algorithm in general [312], although, in the limited curvature case, it is constant-factor approximable (see Theorem 4.3.9 analogous to Theorem 5.4 in [124]).

Submodular span is used as the first step in our conditional summarization strategy, i.e., given a domain V over which a submodular function f is defined, and given a query set $Q \subseteq V$, the objective is to produce a Q-related summary of the remainder V_Q. Submodular span produces a large set A that is related to Q, but to be a good summary, it should also be non-redundant. Hence, given a solution A^*_Q to either Eq. (4.1) or (4.32), one can apply standard submodular maximization (via the greedy algorithm outlined in Algorithm 1), approximately solving Eq. (4.33). The resulting solution is both related to Q and non-redundant. Conditional summarization uses only one submodular function f defined both on Q and everything else V_Q.

4.3.1 Matroids, Span, and Redundancy

The reason we call the above the *submodular span problem* is that for a matroid rank function, it is identical to the matroid span. A matroid $\mathcal{M} = (V, \mathcal{I})$ is an algebraic system consisting of a pair $(V, \mathcal{I})$, where V is a ground set, and $\mathcal{I} = \{I_1, I_2, \dots\}$ is a *non-empty* set of *independent* subsets $I_i \subseteq V$ satisfying the two properties: (1) down-closed, if $I \in \mathcal{I}$ then $A \in \mathcal{I}$ for any $A \subseteq I$, and (2) exchangeable, for all $I_1, I_2 \in \mathcal{I}$ with $|I_1| < |I_2|$, then $\exists j \in I_2 \setminus I_1$ such that $I_1 \cup \{j\} \in \mathcal{I}$. The rank function $r_{\mathcal{M}} : 2^V \to \mathbb{R}$ of a matroid is defined as

$r_{\mathcal{M}}(A) = \max_{I \in \mathcal{I}} |A \cap I|$, i.e., the maximum independent subset of A which is an integer-valued unit-increment polymatroid function. The rank function also defines the matroid, so we can refer to the matroid simply as $r_{\mathcal{M}}$. Given $r_{\mathcal{M}}$ and a query set Q, the span function [246] is defined as:

$$\text{span}_{r_{\mathcal{M}},0}(Q) = \{v \in V : r_{\mathcal{M}}(Q \cup \{v\}) = r_{\mathcal{M}}(Q)\} \tag{4.2}$$

The subscript "$r_{\mathcal{M}}, 0$" notation will become apparent below. The span is also called the "closure" of Q, and the span of Q produces a "flat" (or a "subspace") that contains Q. We also define a Q-specific "redundancy" function redn for a matroid as follows:

$$\text{redn}_{r_{\mathcal{M}},0}(Q) \in \text{argmax}\{|A| : A \subseteq V_Q, r_{\mathcal{M}}(A|Q) = 0\} \tag{4.3}$$

We see that Eq. (4.1) with $f = r_{\mathcal{M}}$ being a matroid rank function and $\epsilon = 0$ computes $\text{redn}_{r_{\mathcal{M}},0}(Q)$. By simple inspection, we see that computing $\text{span}_{r_{\mathcal{M}},0}(Q)$ is much more straightforward via a simple $O(n)$ process than computing $\text{redn}_{r_{\mathcal{M}},0}(Q)$ which appears to be a form of constrained submodular minimization. Therefore, in the next several sections, we study $\text{span}_{r_{\mathcal{M}},0}(Q)$ as a surrogate for $\text{redn}_{r_{\mathcal{M}},0}(Q)$, starting with the case of pure matroids, where there is good news.

Lemma 4.3.1. $\text{redn}_{r_{\mathcal{M}},0}(Q)$ *is unique when* $r_{\mathcal{M}}$ *is a matroid rank function.*

Proof. Let $A_1, A_2 \in \{A \subseteq V \setminus Q : r_{\mathcal{M}}(A|Q) = 0\}$. We prove that $A_1 \cap A_2 \in \{A \subseteq V \setminus Q : r_{\mathcal{M}}(A|Q) = 0\}$ and $A_1 \cup A_2 \in \{A \subseteq V \setminus Q : r_{\mathcal{M}}(A|Q) = 0\}$.

Take any subset $B \subseteq A$, then $0 = r_{\mathcal{M}}(A|Q) \geq r_{\mathcal{M}}(B|Q) \geq 0$. Hence $r_{\mathcal{M}}(B|Q) = 0$. Since $A_1 \cap A_2 \subseteq A$, then $r_{\mathcal{M}}(A_1 \cap A_2|Q) = 0$.

Next, by submodularity, we have:

$$0 = r_{\mathcal{M}}(A_1|Q) + r_{\mathcal{M}}(A_2|Q) \geq \tag{4.4}$$

$$r_{\mathcal{M}}(A_1 \cup A_2|Q) + r_{\mathcal{M}}(A_1 \cap A_2|Q) \geq 0 \tag{4.5}$$

Hence $r_{\mathcal{M}}(A_1 \cup A_2|Q) = 0$. The uniqueness of redn follows by considering the following equivalent definition:

$$\text{redn}_{r_{\mathcal{M}},0}(Q) = \bigcup_{A' \in \{A \subseteq V \setminus Q | r_{\mathcal{M}}(A|Q)=0\}} A' \tag{4.6}$$

□

Theorem 4.3.2. $\text{span}_{r_{\mathcal{M}},0}(Q) = \text{redn}_{r_{\mathcal{M}},0}(Q)$ *when* $r_{\mathcal{M}}$ *is a matroid rank function.*

Proof. We first show that $\text{span}_{r_{\mathcal{M}},0}(Q) \subseteq \text{redn}_{r_{\mathcal{M}},0}(Q)$.

We prove this by induction on $i \in [k]$. Order elements $\{v_1, v_2, \ldots, v_k\} = \text{span}_{r_{\mathcal{M}},0}(Q) \setminus Q$. Then $r_{\mathcal{M}}(v_i|Q) = 0$ for all $i \in [k]$. The base case is as follows: from submodularity, we have $r_{\mathcal{M}}(Q \cup \{v_1\}) + r_{\mathcal{M}}(Q \cup \{v_2\}) \geq r_{\mathcal{M}}(Q) + r_{\mathcal{M}}(Q \cup \{v_1, v_2\})$ and hence $r_{\mathcal{M}}(Q \cup \{v_1\}) \geq r_{\mathcal{M}}(Q \cup \{v_1, v_2\})$ but by monotonicity, $r_{\mathcal{M}}(Q \cup \{v_1\}) \leq r_{\mathcal{M}}(Q \cup \{v_1, v_2\})$. Hence, $r_{\mathcal{M}}(Q) = r_{\mathcal{M}}(Q \cup \{v_1\}) = r_{\mathcal{M}}(Q \cup \{v_2\}) = r_{\mathcal{M}}(Q \cup \{v_1, v_2\})$. Inductively, assume that $r_{\mathcal{M}}(Q) = r_{\mathcal{M}}(Q \cup \{v_1\}) = r_{\mathcal{M}}(Q \cup \{v_1, v_2, \ldots, v_{i-1}\})$.

Then, by submodularity,

$$r_{\mathcal{M}}(Q \cup \{v_1, v_2, \ldots, v_{i-1}\}) + r_{\mathcal{M}}(Q \cup \{v_i\}) \geq \tag{4.7}$$

$$r_{\mathcal{M}}(Q) + r_{\mathcal{M}}(Q \cup \{v_1, v_2, \ldots, v_i\}) \tag{4.8}$$

which implies $r_{\mathcal{M}}(Q \cup \{v_i\}) \geq r_{\mathcal{M}}(Q \cup \{v_1, v_2, \ldots, v_i\})$, and along with monotonicity we get $r_{\mathcal{M}}(Q) = r_{\mathcal{M}}(Q \cup \{v_i\}) = r_{\mathcal{M}}(Q \cup \{v_1, v_2, \ldots, v_i\})$. Taking $i = k$, this implies that $r_{\mathcal{M}}(A|Q) = 0$ where $A = \text{span}_{r_{\mathcal{M}},0}(Q) \setminus Q$. Since $\text{redn}_{r_{\mathcal{M}},0}(Q)$ is the largest set having this property and is unique, the result follows.

Next we show that $\text{redn}_{r_{\mathcal{M}},0}(Q) \subseteq \text{span}_{r_{\mathcal{M}},0}(Q)$. Let $A = \text{redn}_{r_{\mathcal{M}},0}(Q)$ and let $A' \subseteq A$. Then,

$$0 = r_{\mathcal{M}}(A|Q) \geq r_{\mathcal{M}}(A'|Q) \geq 0 \tag{4.9}$$

This implies $r_{\mathcal{M}}(A'|Q) = 0$. Take, in turn, $A' = \{a\}$ for all $a \in A$ giving $r_{\mathcal{M}}(a|Q) = 0$.

Thus, from these results, we conclude that $\text{span}_{r_{\mathcal{M}},0}(Q) = \text{redn}_{r_{\mathcal{M}},0}(Q)$. □

4.3.2 Polymatroids, Span, and Redundancy

We can easily generalize span and redn to polymatroids. Given a polymatroid function f, a set Q such that $Q \subseteq V$, and $\epsilon \geq 0$, the ϵ-span function $\text{span}_{f,\epsilon}(Q)$ is defined as:

$$\text{span}_{f,\epsilon}(Q) = \{v \in V_Q : f(v|Q) \leq \epsilon\}. \tag{4.10}$$

We also define a Q-specific ϵ-redundancy function $\text{redn}_{f,\epsilon}$ for a polymatroid function f as:

$$\text{redn}_{f,\epsilon}(Q) \in \text{argmax}\{|A| : A \subseteq V_Q, f(A|Q) \leq \epsilon\}. \tag{4.11}$$

We see that $\text{redn}_{f,\epsilon}(Q)$ computes the submodular span defined in Eq. (4.1), and hence involves constrained submodular minimization. The question we wish to address is the extent to which $\text{span}_{f,\epsilon}(Q)$ can be used as a surrogate function for $\text{redn}_{f,\epsilon}(Q)$. Analysis comparing the above for the cases when $\epsilon = 0$ and $\epsilon > 0$ follows.

Theorem 4.3.3. *For a polymatroid $f : 2^V \to \mathbb{R}_+$, $\text{span}_{f,0}(Q) = \text{redn}_{f,0}(Q)$.*

Proof. The proof is identical to the proof of Theorem 4.3.2 since the uniqueness of $\text{redn}_{f,0}$ still holds by the proof of Lemma 4.3.1, and nothing in that proof assumed anything more than polymatroidality (i.e., monotone non-decreasing, normalized, and submodular) and uniqueness of $\text{redn}_{f,0}$. □

For $\epsilon = 0$, we observe that computing submodular span and redundancy lead to the same result, analogous to the matroid case. When $\epsilon > 0$, however, this is not the case.

Lemma 4.3.4. $\text{redn}_{f,\epsilon}(Q)$ *is not always unique for* $\epsilon > 0$.

Proof. For the uniqueness of $\text{redn}_{f,\epsilon}(Q)$, we still have that $\epsilon \geq f(A|Q) \geq f(A'|Q) \geq 0$, so that if $A, A' \in \text{redn}_{f,\epsilon}(Q)$, then $A \cap A' \in \text{redn}_{\epsilon}(Q)$. However, the union case has

$$2\epsilon \geq f(A|Q) + f(A'|Q) \geq \tag{4.12}$$

$$f(A \cup A'|Q) + f(A \cap A'|Q) \geq \epsilon \tag{4.13}$$

We also, by monotonicity, have that:

$$f(A \cap A'|Q) \leq \min(f(A|Q), f(A'|Q)) \leq \max(f(A|Q), f(A'|Q)) \leq f(A \cup A'|Q) \tag{4.14}$$

From this, if it was the case that $\max(f(A|Q), f(A'|Q)) \leq f(A \cap A'|Q)$ then we could get that $f(A \cup A'|Q) \leq \epsilon$. But since this is not necessarily the case, the only bound we get is $f(A \cup A'|Q) \leq 2\epsilon$. Hence, $\text{redn}_\epsilon(Q)$ might not be unique.

As an example, take $f(A) = \sqrt{|A|}$. Then, take any two non-intersecting same-sized sets B, B' that don't overlap with Q. Then $f(B|Q) = f(B'|Q) = \sqrt{|B \cup Q|} - \sqrt{|Q|}$, and if we take $\epsilon = \sqrt{|B \cup Q|} - \sqrt{|Q|}$ we have two non-unique maximal sets that could serve as $\text{redn}_{f,\epsilon}(Q)$. □

Theorem 4.3.5. *For a polymatroid $f : 2^V \to \mathbb{R}_+$ such that $n = |V|$, $\text{redn}_{f,\epsilon}(Q) \subseteq \text{span}_{f,\epsilon}(Q) \subseteq \text{redn}_{f,n\epsilon}(Q)$ when $\epsilon \geq 0$.*

Proof. If $f(A|Q) \leq \epsilon$ then, for any ordering $\{a_1, a_2, \ldots, a_k\} = A$, we have

$$f(A|Q) = \sum_{i=1}^{k} \underbrace{f(a_i|a_1, \ldots, a_{i-1}, Q)}_{\geq 0 \text{ and thus } \leq \epsilon} \leq \epsilon \tag{4.15}$$

Since each term is non-negative, and they sum to ϵ, they all must be at most ϵ. Hence $f(a_1|Q) \leq \epsilon$. Since the order is arbitrary, each a_i takes its turn in the first position, giving $f(a|Q) \leq \epsilon$ for $a \in A$.

A simpler proof is to just note that if $f(A|Q) \leq \epsilon$, then by monotonicity, $f(a|Q) \leq f(A|Q) \leq \epsilon$ for any $a \in A$.

To get the upper set bound, i.e., that $\text{redn}_{n\epsilon}(Q) \supseteq \text{span}_\epsilon(Q)$ when $\epsilon \geq 0$, Assume for a given A that $f(a|Q) \leq \epsilon$. Then $f(A|Q) \leq \sum_{a \in A} f(a|Q) \leq |A|\epsilon \leq n\epsilon$.

We can find a strict example where $\text{redn}_{f,\epsilon}(Q) \subset \text{span}_{f,\epsilon}(Q)$ when $\epsilon > 0$. For $f(A) = \sqrt{A}$, we only need to find a value $m = |Q|$ and ϵ such that $\sqrt{m+2} - \sqrt{m} > \epsilon$ and $\sqrt{m+1} - \sqrt{m} < \epsilon$. Take $m = 1$ and $\epsilon = 0.5$. Then $\sqrt{3} - \sqrt{1} \approx 0.7321 > \epsilon = 0.5$ and $\sqrt{2} - \sqrt{1} \approx 0.4142 < \epsilon = 0.5$.

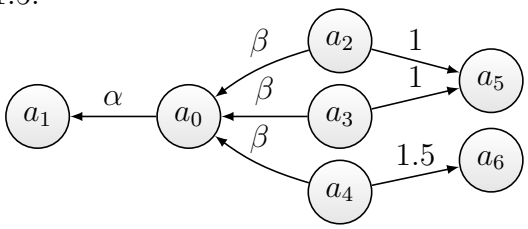
□

For a given ϵ, since $\text{span}_{f,\epsilon}(Q)$ covers all the elements of $\text{redn}_{f,\epsilon}(Q)$, we can compute the $\text{span}_{f,\epsilon}(Q)$ as a surrogate function for $\text{redn}_{f,\epsilon}(Q)$ and then summarize it. But first, we ask if, for some value of $\epsilon' \leq \epsilon$, their f-valuations are equal. Unfortunately, this is also not the case.

Lemma 4.3.6. *There does not, in general, exist an $\epsilon' \leq \epsilon$ such that $f(\text{span}_{f,\epsilon'}(Q)) = f(\text{redn}_{f,\epsilon}(Q))$ for all Q and $\epsilon > 0$.*

Proof. Assume that $\exists\ \epsilon' \leq \epsilon$ such that $f(\text{span}_{f,\epsilon'}(Q)) = f(A)$ where $A \in \text{redn}_{f,\epsilon}(Q)$. Since $\text{redn}_{f,\epsilon}(Q)$ is not always unique (from Lemma 4.3.4), there could exist a set $B \in \text{redn}_{f,\epsilon}(Q)$ such that $f(B) \neq f(\text{span}_{f,\epsilon'}(Q))$.

As a strict example, consider the directed graph below where $Q = \{a_0\}, \epsilon' = 1$ and $\epsilon = 2.5$. Let $f(A) = |\delta^+(A)|$ be the directed cut function, so $f(a_2|Q) = f(a_3|Q) = 1$ and $f(a_4|Q) = 1.5$.



Thus, $A = \text{span}_{f,\epsilon'}(Q) = \{a_2, a_3\}$. For the given ϵ, one can see that all three sets $A = \{a_2, a_3\}, B = \{a_3, a_4\}, C = \{a_2, a_4\}$ can serve as $\text{redn}_{f,\epsilon}(Q)$, but $f(B) = f(C) \neq f(A)$.

Similarly, for any value of $\epsilon' \leq \epsilon$, it can be shown for the given example, $f(\text{span}_{f,\epsilon'}(Q))$ $\neq f(\text{redn}_{f,\epsilon}(Q))$ for some set in $\text{redn}_{f,\epsilon}(Q)$.

□

Since there does not exist an $\epsilon' \leq \epsilon$ for which $f(\text{span}_{f,\epsilon'}(Q)) = f(\text{redn}_{f,\epsilon}(Q))$ when $\epsilon > 0$, we can form an upper bound on $f(\text{span}_{f,\epsilon}(Q))$ as follows.

Lemma 4.3.7. $f(\text{span}_{f,\epsilon}(Q)|Q) \leq (k_s - k_r + 1)\epsilon$ *where* $k_s = |\,\text{span}_{f,\epsilon}(Q)|$ *and* $k_r = |\,\text{redn}_{f,\epsilon}(Q)|$

Proof. Let $A_\text{s} = \text{span}_{f,\epsilon}(Q)$ and $|A_\text{s}| = k_\text{s}$. Similarly $A_\text{r} = \text{redn}_{f,\epsilon}(Q)$ and $|A_\text{r}| = k_\text{r}$.

From Theorem 4.3.5, we know that $\text{redn}_{f,\epsilon}(Q) \subseteq \text{span}_{f,\epsilon}(Q)$, so $A_r \subseteq A_s$. Let $A_s = A_r \cup A$ such that $A \cap A_r = \emptyset$.

For a submodular function f, we know that:

$$f(Y) \leq f(X) + \sum_{j \in Y \setminus X} f(j|X) - \sum_{j \in X \setminus Y} f(j|(X \cup Y) \setminus j) \tag{4.16}$$

$\forall \ X, Y \subseteq V$. Substituting X with A_r and Y with A_s in Eq. (4.16) and using the modified submodular function f_Q, we get:

$$f_{\mathrm{Q}}(A_{\mathrm{s}}) \leq f_{\mathrm{Q}}(A_{\mathrm{r}}) + \sum_{j \in A} f_{\mathrm{Q}}(j|A_{\mathrm{r}}) \tag{4.17}$$

Since, $f_{\mathrm{Q}}(j|A_{\mathrm{s}}) \leq f_{\mathrm{Q}}(j) \leq \epsilon \quad j \in A$, we modify the upper bound on $f_Q(A_s)$ as follows:

$$f_{\mathrm{Q}}(A_{\mathrm{s}}) \leq \epsilon + (k_{\mathrm{s}} - k_{\mathrm{r}})\epsilon = (k_{\mathrm{s}} - k_{\mathrm{r}} + 1)\epsilon \tag{4.18}$$

In a situation where k_s is much larger than k_r, which could happen when there are many elements in the ground set that are redundant to Q but are mostly non-redundant with respect to each other. So, the worst case bound in Eq. (4.18) could be $n\epsilon$ where $n = |V|$

A strict example where $f_Q(A_s) \approx n\epsilon$ for $\epsilon \geq 0$ is as follows: Let $f(A) = \min(|A|, n)$ where $n = |V|$ and $\epsilon = 1$. For all $j \in V \setminus Q$, $f_Q(j) = (|Q| + 1) - |Q| = 1 \leq \epsilon$. So, all such data points will belong to the spanning set of Q such that $f_Q(A_s) \approx n\epsilon$ for $|Q| \ll n$. □

Lemma 4.3.7 shows that for our surrogate function $\text{span}_{f,\epsilon}(Q)$, the worst case bound on its f-valuation with respect to Q could be $n\epsilon$ where $n = |V_Q|$. This is most likely when the ground set V contains many elements that are redundant to Q but that are mostly mutually non-redundant.

Lemma 4.3.8. *With the conditional submodular curvature with respect to Q defined as*

$$\kappa_{f_Q}(A) \triangleq 1 - \min_{a \in A} \frac{f((a|(A \setminus a)), Q)}{f(a|Q)}, \tag{4.19}$$

$f(\text{span}_{f,\frac{\epsilon}{n}}(Q)|Q) \leq \epsilon - \frac{\epsilon}{n}(k_r - k_s)(1 - \kappa_{f_Q}(\text{redn}_{f,\epsilon}(Q)))$ *where* $k_s = |\,\text{span}_{f,\frac{\epsilon}{n}}(Q)|$ *and* $k_r = |\,\text{redn}_{f,\epsilon}(Q)|$.

Proof. Let $A_s = \text{span}_{f,\frac{\epsilon}{n}}(Q)$ and $|A_s| = k_s$. Similarly, let $A_r = \text{redn}_{f,\epsilon}(Q)$ and $|A_r| = k_r$.

From Theorem 4.3.5, we know that $\text{span}_{f,\frac{\epsilon}{n}}(Q) \subseteq \text{redn}_{\epsilon}(Q)$, so $A_s \subseteq A_r$. Let $A_r = A_s \cup A$ such that $A \cap A_s = \emptyset$.

Let $j_{\min} = \text{argmin}_{j \in A} f_Q(j)$. Since $A \notin A_s$, we have $\frac{\epsilon}{n} \leq f_Q(j_{\min}) \leq \epsilon$.

Substituting X with A_r and Y with A_s in Eq. (4.16), and using submodular function f_Q, we get:

$$f_Q(A_s) \leq f_Q(A_r) - \sum_{j \in A} f_Q(j|(A_r \setminus j)) \tag{4.20}$$

$$\overset{\text{a}}{\leq} \epsilon - (k_{\text{r}} - k_{\text{r}})(1 - \kappa_{f_{\text{Q}}}(A_{\text{r}}))f_{\text{Q}}(j_{\min}) \tag{4.21}$$

$$\overset{\text{b}}{\leq} \epsilon - (k_r - k_s)(1 - \kappa_{f_Q}(\text{redn}_{f,\epsilon}(Q)))\frac{\epsilon}{n} \tag{4.22}$$

where a holds because $f_Q(A_r) \leq \epsilon$ and from the definition of conditional submodular curvature defined as:

$$\kappa_{f_Q}(A) \triangleq 1 - \min_{a \in A} \frac{f_Q(a|(A \setminus a))}{f_Q(a)} \tag{4.23}$$

and b holds because of the lower bound on $f_Q(j_{\min})$. □

To solve Eq. (4.32) or (4.1), a modular approximation of $f(A|Q)$, denoted as $m_Q(A) = \sum_{a \in A} f(a|Q)$, can be optimized. The Majorization-Minimization algorithm based on submodular semi-differentials, as proposed in [124], can also be used for constrained submodular minimization. The approximation factor for these algorithms is expressed in terms of conditional submodular curvature with respect to Q as proved in Theorem 4.3.9 and has a worst-case upper bound of $\mathcal{O}(n)$ where $n = |V \setminus Q|$

Theorem 4.3.9. *Let A^* be the optimal solution to Eq. (4.32), then A returned by the modular approximation of $f(A|Q)$ such that $A = \text{argmin}_{A \subseteq V_Q, |A| \geq k} m_Q(A)$ satisfies:*

$$f(A|Q) \leq \frac{|A^*|}{1 + (|A^*| - 1)(1 - \kappa_{f_Q}(A^*))} f(A^*|Q)$$

Proof. Here also, lets define $f_Q(A)$ such that $f_{\text{Q}}(A) = f(A \cup Q) - f(Q) = f(A|Q)$.

Let A^* denotes the optimal solution such that $A^* = \text{argmin}_{A \subseteq V', |A| \geq k} f_Q(A)$, and A denotes the approximate solution such that $A = \text{argmin}_{A \subseteq V', |A| \geq k} m_Q(A)$ where $m_Q(A) = \sum_{a \in A} f_Q(a)$ and $V' = V \setminus Q$. Let $a_{\max} = \text{argmax}_{a \in A^*} f_Q(a)$.

$$f_Q(A) \leq \sum_{a \in A} f_Q(a) = m_Q(A) \leq m_Q(A^*) \tag{4.24}$$

$$= \sum_{a \in A^*} f_Q(a) \leq |A^*| f_Q(a_{\max}) \tag{4.25}$$

Substituting Y with $a_{\max}$, X with A^* and using the modified submodular function f_Q, we get:

$$f_Q(A^*) \geq f_Q(a_{\max}) + \sum_{a \in (A^* \setminus a_{\max})} f_Q(a | A^* \setminus a) \tag{4.26}$$

$$\geq f_Q(a_{\max}) + (|A^*| - 1) \min_{a \in A^*} f_Q(a | A^* \setminus a) \tag{4.27}$$

Combining Eq. (4.24) and (4.26), we get:

$$\frac{f_Q(A)}{f_Q(A^*)} \leq \frac{|A^*| f_Q(a_{\max})}{f_Q(a_{\max}) + (|A^*| - 1) \min_{a \in A^*} f_Q(a | A^* \setminus a)} \tag{4.28}$$

$$\leq \frac{|A^*|}{1 + (|A^*| - 1) \frac{\min_{a \in A^*} f_Q(a | A^* \setminus a)}{f_Q(a_{\max})}} \tag{4.29}$$

$$\leq \frac{|A^*|}{1 + (|A^*| - 1)(1 - \kappa_{f_Q}(A^*))} \tag{4.30}$$

Iyer et al. [124] show a bound for submodular function minimization in terms of generic submodular curvature, and our proof follows theirs.

□

4.3.3 Submodular Span Summarization

Our conditional summarization framework, S3, is formulated as a two-stage submodular optimization problem. The first stage aims to select a large subset relevant to the query set, and the second stage summarizes the output of stage one to get a diverse summary.

Stage 1 of S3 targets selecting a relatively large subset relevant to the query set. Mathematically, given a ground set V that includes the query set Q and the data being summarized

V_Q (where $V_Q = V \setminus Q$), and a submodular function f defined on the entire ground set V, the submodular span optimization problem is defined as follows:

$$\max_{A \subseteq V_Q} |A| \;\; \text{s.t.} \;\; f(A|Q) \leq \epsilon \tag{4.31}$$

where $\epsilon \geq 0$ is a small scalar controlling the desired relevance level. $f(A|Q) := f(A \cup Q) - f(Q)$ denotes the conditional gain of set A given the query set Q. As described in Sec. 2.3.2, low $f(A|Q)$ represents high conditional redundancy of A given query set Q. Thus, to optimize Eq. (4.31), we get a set A with a low Q-conditioned f-valuation. The dual to this problem is shown below, where we minimize the conditional gain $f(A|Q)$ subject to a lower-bound cardinality constraint. This first stage retrieves all data points relevant to the query, but that might be redundant, as follows:

$$\text{Stage 1:} \min_{A \subseteq V \setminus Q, |A| \geq k_1} f(A|Q). \tag{4.32}$$

That is, we minimize a monotone, non-decreasing conditional submodular function $f(A|Q)$ (representing the conditional redundancy) subject to a cardinality lower-bound constraint. This optimization does not have a constant factor approximation algorithm [313].

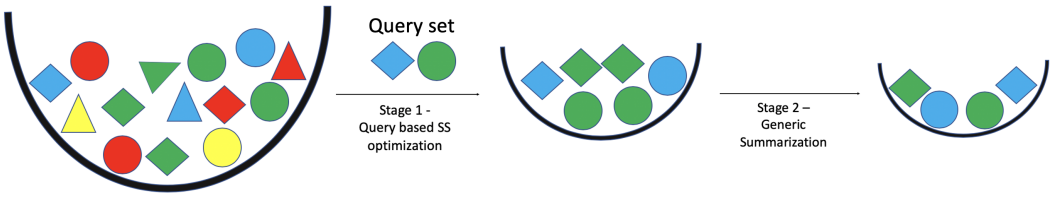


Figure 4.1: Example showing how the two stages of the S3 Framework work sequentially to obtain a query-focused summary.

Stage 2 of S3 The second stage is a standard cardinality-constrained submodular maximization problem starting from the solution of stage one as follows:

$$\text{Stage 2:} \max_{A \subseteq A^*_Q, |A| \leq k_2} f(A) \tag{4.33}$$

where A_Q^* is the solution of stage one. This second stage summarizes the redundant output of stage one and produces a diverse and succinct summary of the data that is still relevant to Q (i.e., stage two filters out the redundancy in A_Q^*). To solve stage two, we use the standard greedy algorithm [239, 231] (described in Algorithm 1). S3 utilizes a single submodular function for both stages, and the utility function does not need to be reformulated as the query set changes.

In Fig. 4.1, we provide an example of how the two stages in the S3 framework work. Given a ground set consisting of points with two attributes, color, and shape, stage 1 retrieves elements that are redundant with respect to the query set Q. Stage 2 then summarizes the resultant Q-specific span that we got from Stage 1.

Maximizing the submodular mutual information function described in Sec. 2.3.1 can also be utilized to perform query-focused summarization. However, a two-stage approach—first computing the submodular span by minimizing the conditional gain (Stage 1 of S3) and then summarizing the submodular span (Stage 2 of S3)—provides finer control over balancing query relevance and diversity. In Stage 1, S3 filters out non-relevant samples by minimizing conditional gain, ensuring that only query-relevant items are retained. This isolates the relevance objective, allowing the subsequent stage to focus exclusively on optimizing diversity within a query-relevant subset. Stage 2 summarization over the filtered set helps reduce redundancy among already relevant samples rather than across the entire dataset. This sequential focus on relevance and diversity objectives allows for better tuning of each component independently, making the S3 framework beneficial for tasks such as targeted data selection and efficient input-context selection for large language models (LLMs) and multimodal LLMs, as discussed in Sec. 3.1.2 and Sec. 3.1.3, respectively.

For all the experiments in this chapter, we use *submarine* [27] for submodular optimization.

4.4 Experiments

In this section, using optimization procedures based on the analysis given in Sec. 4.3.2, we evaluate the S3 framework on three conditional summarization tasks: (1) conditional multi-document summarization, (2) conditional video summarization, (3) conditional image corpus summarization.

4.4.1 Conditional Multi-Document Summarization

Dataset We use DUC 2005-2007 datasets, the benchmark datasets for query-focused MDS, made available by the Document Understanding Conference [1]. DUC 2005-2006 and DUC 2007 contain 50 and 45 document clusters, respectively, with each cluster containing 25 news articles (32 in the case of DUC 2005) related to the same topic. The task is to generate a query-focused summary of at most 250 words for each document cluster. As a pre-processing step, we remove special characters from the sentences, augment the query set for each document cluster with its topic, and concatenate each query sentence with the cluster topic.

Feature Representation In order to obtain sentence representations, we use the English uncased variant of the BERT-base model [72] and fine-tune it for the Rouge-2 recall score prediction task using two years of DUC 2005-2007 as the training set. For example, we fine-tune the network on the DUC 2005-2006 datasets in order to extract fixed-size sentence representations for DUC 2007 (which is the test set in this example). We do not use any oracle summarization labels for the test set. In addition to using fine-tuned BERT models, we also try a minimally supervised approach where we use the pre-trained BERT model for computing sentence representations.

Since BERT's encoder has 12 transformer layers, each of which outputs contextualized WordPiece representations, the most transferable layer l [86] for the MDS task is a hyperparameter that is tuned on the development set. Given l, we take a smoothed inverse frequency

[1] https://duc.nist.gov

(SIF) based weighted average of hidden activations of each wordpiece [251, 14] from layer l to construct 768-dimensional sentence embeddings v_{s_i} for the sentences s_i in the test set i.e., $v_{s_i} = \frac{1}{|s_i|}\sum_{w \in s_i} \frac{a}{a+p(w)} h_l(w)$. Here, $h_l(w)$ is the hidden layer representation of wordpiece w corresponding to layer l, $p(w)$ is the probability of wordpiece w estimated from the entire DUC corpus, and a is a weighting parameter fixed at 10^{-3}.

Fine-Tuning BERT We use the uncased variant of the BERT-base model and fine-tune it on a pair of DUC 2005-2007 datasets in order to generate feature representations for the remaining dataset, which is the test set. We flatten the query tokens and the sentences tokens and then insert [Q#] token before the #-th query sentence as well as insert a [CLS] token at the beginning and [SEP] token at the end of a sentence to mark the sentence boundary. We then fine-tune the BERT-based conditional summarization model as proposed in [404] for a sentence regression task. This is achieved by adding a linear layer on top of the BERT-encoder, which predicts the score for sentence s_i using h_i where $h_i = \text{BERT}(s_i)_{[\text{CLS}]}$. For the network architecture diagram, we ask the readers to refer to Fig. 2 in [404].

For each sentence in the document, the model predicts a regression score $r(s_i|\mathcal{Q}, \mathcal{D})$ based on its relevance to the query $\mathcal{Q}$ and its salience to the document $\mathcal{D}$. The training objective is to minimize the mean square error between the predicted scores and the oracle scores that we evaluate using the gold summary $\mathcal{S}^*$.

$$\mathcal{L} = \frac{1}{n}\sum_{i=n}^{n}(t(s_i|\mathcal{S}^*) - r(s_i|\mathcal{Q}, \mathcal{D}))^2 \tag{4.34}$$

Since the maximum sequence token for BERT-base is set to 512, we split long documents into multiple training examples such that the overlap of a new training example with the previously generated example is four sentences. Using this setup, we are able to generate 12704 training examples for DUC 2005, 9605 training examples for DUC 2006, and 6727n training examples for DUC 2007.

We fine-tune the BERT-encoder model three times on each pair of query-focused DUC

datasets. For training, similar to [404], we use Adam optimizer [143] with learning rate of $3e-5$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, linear decay of learning rate, and L2 weight decay of 0.01. We set the batch size as 32 and train the above-mentioned architecture for three epochs. For each pair of training sets, for example, DUC 2005-2006, we instantiate the network using the original BERT-base weights and do not make use of any fine-tuned weights obtained via training on other pairs of DUC 2005-2007. This way, we ensure that the computed sentence representations for the test set (DUC 2007 in this example) do not capture any information about the oracle summarization scores.

After fine-tuning, we use the network fine-tuned on DUC 2005-2006 sets to extract sentence representations for the left-out DUC 2007. Similarly, we use the network fine-tuned on DUC 2005 and 2007 to extract sentence representations for DUC 2006 and so on.

Summary Generation We use facility location [232] described in Sec. 2.1.2 as the objective function for stages one and two of the S3 framework. The facility location function is defined as

$$f(A) = \sum_{v \in V} \max_{a \in A} \ \text{sim}(a, v) \tag{4.35}$$

where $\text{sim}(a, v)$ is the similarity between sentence embeddings z_a and z_v of sentences a and v. We compute the similarity matrix using a Gaussian kernel of width σ, which is tuned on the development set in each case.

Stage one of the S3 framework caters to finding relevant sentences A_Q from a document set that answers given queries. In order to filter irrelevant noisy sentences that are either too small or too long, we prune the candidate set by considering sentences whose length ranges between 11 and 80 and are a subset of the top 30% nearest neighbors set of the query sentences. Once we obtain the relevant answers (A_Q) using the majorization-minimization (MMin) [124, 123] algorithm for solving Eq. (4.1), stage two removes the redundant answers and produces a succinct relevant summary for that document set via constrained submodular maximization using the greedy algorithm [239] described in Algorithm 3. The algorithm at it-

eration i selects the sentence s_i such that $s_i = \text{argmax}_{s \in A_Q} \frac{f(A_{i-1} \cup s) - f(A_{i-1})}{(c(s))^r}$ if $c(A_{i-1} \cup s_i) \leq \mathcal{B}$ (according to line 6 of Algorithm 3). $c(s)$ denotes the sentence length, $r > 0$ is the scaling factor, and $\mathcal{B}$ represents the overall budget which is 250 words for DUC 2005-2007. For DUC-2005, we use DUC-2006 to tune the hyperparameters which include $\{l, \sigma, \epsilon, r\}$. Similarly, for DUC-2006 and DUC-2007, we use DUC-2005 as the development set.

Hyperparameter Tuning Since the BERT-base has 12 transformer layers, we first set to investigate which of these layers is most transferable for the conditional summarization task. We find out that the layers close to the input are most transferable for this task. This is in assent with our feature extraction strategy which does not allow the representations to encode any information about the oracle summary. Specifically, we found that the second layer (starting from the input layer, which is layer 0) is the most transferable after comparing the Rouge-2 F-measure scores across all 12 layers. We also tried using the transformer's output corresponding to the [CLS] token but it led to rather poor performance as it is suited mostly for sentence classification tasks. Also, in the case of the minimally supervised case where we used the pre-trained BERT weights to generate sentence representations for instantiating the submodular function, we found that the first and second layers close to the input were most transferable for the summarization task. So, when using the pre-trained BERT encoder weights, average pooled sentence representations across the first and second layers lead to the best minimally supervised results on the DUC 2005-2007 datasets.

For computing ϵ such that $f(A|Q) \leq \epsilon$, we modify the inequality $f(A|Q) \leq f(V|Q)$ to $f(A|Q) \leq \epsilon' f(V|Q)$ where $0 < \epsilon' \leq 1$. Among similarity measures for instantiating the facility location function, we compare three similarity measures: (1) modified cosine similarity, which truncates all negative similarities to zero (2) d - *euclidean distance* where d is the maximum Euclidean distance between any pair of feature representations (3) Gaussian kernel with width σ. The hyperparameters set includes: (1) $\epsilon' \in \{0.05, 0.1, 0.125, 0.15, 0.175, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45\}$ (2) $\sigma \in \{0.75, 1.0, 1.25, 1.5, 2, 5, 10, 20, 50, 100, 1000, 5000, 10000\}$ (3) $r \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$. We use DUC 2005 as the

development set for DUC 2006 and DUC 2007. For DUC 2005, we use DUC 2006 as the development set for hyperparameter tuning. We found that similarity based on Gaussian kernel with width $\sigma = 1.0$, $\epsilon' = 0.25$, and $r = 0.3$ lead to the best-reported performance on all three DUC datasets.

Evaluation We use the ROUGE toolkit [193][2] which assesses the summary quality by counting the overlapping units such as n-grams, word sequences, and word-pairs between the candidate summary and the reference summaries. We report recall and F-measure corresponding to Rouge-1, Rouge-2, and Rouge-SU4.

Since our approach requires minimal learning of hyperparameters, we compare it against other state-of-the-art unsupervised and supervised approaches. In addition to existing supervised methods, we also design another strong supervised baseline method called *MixModSub* which utilizes a submodular function $f' : 2^{V'} \rightarrow \mathbb{R}_+$, i.e., the query set Q is extrinsic to the submodular function f' i.e., $V' \cap Q = \emptyset$ and it only considers V' which contains sentences s_i where $i \in \{1, 2, \ldots |V'|\}$ that are to be summarized. Here, f' is a facility location function defined using the fine-tuned BERT-based feature vectors v_{s_i} for each $s_i \in V'$. We define a relevance based modular function $m_Q : 2^{V'} \rightarrow \mathbb{R}_+$ where for any $A \subseteq V', m_Q(A) = \sum_{s_i \in A} m_Q(s_i)$. Since m captures the relevance of each sentence s_i to the query set Q, $m_Q(s_i) = \frac{1}{|Q|} \sum_{s_q \in Q} \text{sim}(s_q, s_i)$. Finally, we define a submodular function $g : 2^{V'} \rightarrow \mathbb{R}_+$ as $g(A) = \lambda f'(A) + (1 - \lambda) m_Q(A)$ which is a convex mixture of submodular f' and modular m_Q as described in Sec. 2.1.2. We use a Gaussian kernel of width σ to define the similarity matrix and perform budget-constrained submodular maximization given budget $\mathcal{B}$. Like the previous experiments, we tune the hyperparameters $\{\sigma, r, \lambda\}$ on another year of DUC as the development set.

Table 4.1 shows the average recall and F-measure with respect to Rouge-1 (R1), Rouge-2 (R2), and Rouge-SU4 (RSU4) scores on DUC 2005-2007 datasets against different methods. The performance of S3 framework is competitive with the current unsupervised state-of-the-

[2]ROUGE version 1.5.5 used with option -n 2 -x -m -2 4 -u -c 95 -r 1000 -f A -p 0.5 -t 0 -d -l 250

Table 4.1: ROUGE results on DUC 2005, 2006, and 2007 in terms of Recall and F-measure. Methods marked with (*) are supervised in terms of using oracle summarization labels for training or model fine-tuning.

	System	R1-R	R1-F	R2-R	R2-F	RSU4-R	RSU4-F
DUC 2005	MixModSub*	38.64	38.17	7.74	7.65	13.65	13.49
	SRSum [271]*	39.83	-	**8.57**	-	-	-
	Dual-CES [278]	**40.82**	38.08	8.07	7.54	**14.13**	13.17
	S3 (Ours)*	39.11	**38.66**	7.87	**7.79**	13.80	**13.65**
	S3 (Ours)	38.64	38.20	7.60	7.52	13.52	13.37
DUC 2006	MixModSub*	39.80	39.57	8.62	8.58	14.40	14.32
	DSDR [115]	-	33.17	-	6.05	-	-
	SpOpt [365]	39.96	-	8.68	-	14.23	-
	DocRebuild [216]	-	40.86	-	8.48	-	14.45
	SRSum [271]*	42.82	-	**10.46**	-	-	-
	Dual-CES [278]	**43.94**	41.23	10.09	**9.47**	**15.96**	14.97
	S3 (Ours)*	41.62	**41.42**	9.48	9.43	15.10	**15.02**
	S3 (Ours)	41.13	40.95	9.24	9.20	14.85	14.79
DUC 2007	MixModSub*	40.87	40.42	10.26	10.15	15.66	15.49
	DSDR [115]	-	39.57	-	7.44	-	-
	CTSUM [332]	43.10	42.66	10.93	10.82	16.32	16.16
	SpOpt [365]	42.36	-	11.11	-	16.47	-
	DocRebuild [216]	-	42.73	-	10.31	-	15.81
	SRSum [271]*	45.01	-	**12.80**	-	-	-
	Dual-CES [278]	**46.02**	**43.24**	12.53	**11.78**	**17.91**	**16.83**
	S3 (Ours)*	43.42	42.95	11.24	11.12	16.70	16.52
	S3 (Ours)	42.50	42.32	11.12	11.07	16.35	16.28

art method, Dual-CES [278] on each of the Rouge-1 , Rouge-2, and Rouge-SU4 F-measure.

Additional Results In Table 4.2, we provide an example of a conditional summary generated using the S3 framework and compare it with their generic summaries (sentence ordering in place). We use an AAAI-20 outstanding paper as an example here. It can be seen that the summary conditioned on the title of the paper as the query is better in terms of capturing the most salient and relevant sentences from the abstract of the paper. In this example, we set the size constraint corresponding to stage one as six and the final summary size as three sentences.

4.4.2 Conditional Video Summarization

Dataset We use the query-focused video summarization dataset from [296] which is compiled using the UT Egocentric dataset [171]. The UTE dataset consists of four daily life egocentric videos that are 3-5 hours long. Based on the overlap of the video-shot captions [369] with SentiBank [34], a lexicon of 48 concepts, such as street, tree, phone, etc., is constructed that denotes the basis for encoding the semantic information in each video shot. For each video, there are 46 different sets of queries, with each query set covering two or three concepts.

For evaluation purposes, authors in [296] have provided user-annotated semantic vectors of each video shot and have defined a similarity function between two video shots as the Intersection Over Union (IOU) of their corresponding concepts. For instance, if one shot is tagged by {*car, street*} and another by {*street, tree, sign*}, then the IOU similarity between them is $1/4 = 0.25$. Next, a maximum weight-based bipartite graph matching between the oracle summary shots and the system-generated summary shots is computed using the IOU scores, which enables us to evaluate precision, recall, and F1 score between the matched pairs. We use the oracle summaries provided in [296] and follow their video summarization evaluation strategy based on the user-annotated semantic vectors of the video shots.

Table 4.2: Generic summary and conditional summary of the abstract of the AAAI-20 outstanding paper using its title as the query. Dissimilar sentences among the summaries are marked in *italics*.

Title: WinoGrande: An Adversarial Winograd Schema Challenge at Scale
Abstract: The Winograd Schema Challenge (WSC) (Levesque, Davis, and Morgenstern 2011), a benchmark for commonsense reasoning, is a set of 273 expert-crafted pronoun resolution problems originally designed to be unsolvable for statistical models that rely on selectional preferences or word associations. However, recent advances in neural language models have already reached around 90% accuracy on variants of WSC. This raises an important question whether these models have truly acquired robust commonsense capabilities or whether they rely on spurious biases in the datasets that lead to an overestimation of the true capabilities of machine commonsense. To investigate this question, we introduce WinoGrande, a large-scale dataset of 44k problems, inspired by the original WSC design, but adjusted to improve both the scale and the hardness of the dataset. The key steps of the dataset construction consist of (1) a carefully designed crowdsourcing procedure, followed by (2) systematic bias reduction using a novel AfLite algorithm that generalizes human-detectable word associations to machine-detectable embedding associations. The best state-of-the-art methods on WinoGrande achieve 59.4-79.1%, which are 15-35% below human performance of 94.0%, depending on the amount of the training data allowed. Furthermore, we establish new state-of-the-art results on five related benchmarks - WSC (90.1%), DPR (93.1%), COPA (90.6%), KnowRef (85.6%), and Winogender (97.1%). These results have dual implications: on one hand, they demonstrate the effectiveness of WinoGrande when used as a resource for transfer learning. On the other hand, they raise a concern that we are likely to be overestimating the true capabilities of machine commonsense across all these benchmarks. We emphasize the importance of algorithmic bias reduction in existing and future benchmarks to mitigate such overestimation.
Generic Summary: *This raises an important question whether these models have truly acquired robust commonsense capabilities or whether they rely on spurious biases in the datasets that lead to an overestimation of the true capabilities of machine commonsense.* To investigate this question, we introduce WinoGrande, a large-scale dataset of 44k problems, inspired by the original WSC design, but adjusted to improve both the scale and the hardness of the dataset. Furthermore, we establish new state-of-the-art results on five related benchmarks - WSC (90.1%), DPR (93.1%), COPA (90.6%), KnowRef (85.6%), and Winogender (97.1%).
Conditional Summary: *The Winograd Schema Challenge (WSC), a benchmark for commonsense reasoning, is a set of 273 expert-crafted pronoun resolution problems originally designed to be unsolvable for statistical models that rely on selectional preferences or word associations.* To investigate this question, we introduce WinoGrande, a large-scale dataset of 44k problems, inspired by the original WSC design, but adjusted to improve both the scale and the hardness of the dataset. Furthermore, we establish new state-of-the-art results on five related benchmarks - WSC (90.1%), DPR (93.1%), COPA (90.6%), KnowRef (85.6%), and Winogender (97.1%).

Feature Representation We uniformly partition each video into five-second long shots. For each frame belonging to a shot, we construct a 2089-dimensional feature vector using an off-the-shelf deep model called DeepSentiBank [49] for fair comparison to existing methods. The network has an architecture similar to the AlexNet [153], which was pre-trained on the ImageNet classification task, and for the SentiBank classification task, the last fully connected layer is replaced to produce a softmax distribution across the 2089 class labels. Then, we average each shot's frame-level feature representations to obtain a shot-level feature representation. The SentiBank classes consist of ANP (Adjective Noun Pairs); for instance, *beautiful sky, clear sky, sunny sky* are different ANPs corresponding to the concept *sky*. We max-pool their shot-level detection scores to get one detection score for each concept belonging to the lexicon consisting of 48 concepts. This results in a 48-dimensional feature representation for each video shot with detection scores ranging between 0 and 1.

Summary Generation Similar to Sec. 4.4.1, we use facility location as the objective function, but here $\text{sim}(s_i, s_j)$ is the similarity between the DeepSentiBank-based shot-level features for shots s_i and s_j. We compute the similarity matrix using cosine similarity after carefully validating different similarity measures on a development set in terms of F1 score performance. For Video-1, we use Video-3 to tune the hyperparameters, which include $\{k_1, k_2\}$; k_1 and k_2 are the cardinality constraints for optimizing stage one and stage two, respectively. For Video 2-4, we use Video-1 as the development set.

Hyperparameter Tuning For the facility location function, we again investigate three similarity measures: (1) cosine similarity, (2) d - *euclidean distance* where d is the maximum Euclidean distance between any pair of video shots (3) Gaussian kernel with width σ. In this case, we do not need to modify the cosine similarity to be non-negative, as the video shot features are simply probability scores for each lexicon concept. For determining the cardinality constraints, k_1 for stage one and k_2 for stage two of the S3 framework, we use $k_1^{'}$ such that $k_1 = \left\lfloor k_1^{'}|V_i| \right\rfloor$ where V_i denotes the set of video shots present in the video i that

has to be summarized. Similarly, we introduce $k_2^{'}$ corresponding to k_2. The hyperparameters set includes: (1) $\sigma \in \{0.05, 0.1, 0.5, 1, 2, 10, 100, 200, 1000, 10000\}$ (2) $k_1^{'} \in \{0.2, 0.25, 0.275, 0.3, 0.325, 0.35, 0.375, 0.4, 0.45\}$ (3) $k_2^{'} \in \{0.01, 0.02, 0.025, 0.03, 0.035, 0.04, 0.045, 0.05, 0.07, 0.09, 0.1\}$. We tried both three-fold cross-validation, i.e., using three videos for hyperparameter tuning and testing them on the held-out test video, and one-fold cross-validation, which led to the best set of results. For Video 1, we use Video 3 as the development set, and for all remaining videos, we use Video 1 as the development set. Among the different similarity measures, cosine similarity led to the best results in terms of the three metrics: precision, recall, and the F1 score. For Video 1, corresponding to $k_1^{'} = 0.275$ and $k_2^{'} = 0.03$, we get the results reported in Table 4.3. For the remaining videos, where we use Video 1 as the development set, we get the reported results corresponding to $k_1^{'} = 0.375$ and $k_2^{'} = 0.03$.

Evaluation Similar to [296], we use the user-annotated semantic vectors of video shots to quantify the semantic similarity between the oracle summary's shots and our system-generated summary's shots. A maximum weight-based bipartite graph matching between them enables us to compute precision, recall, and F1 score between the matched pairs. Similar to document summarization, we also compare against the designed minimally supervised baseline *MixModSub* (not fully supervised as we are not fine-tuning any model using oracle summary labels). We tune relevant hyperparameters $\{k, \lambda\}$ on another video as the development set.

Table 4.3 shows the performance of the S3 framework against other supervised state-of-the-art methods and our designed baseline *MixModSub* in terms of precision, recall, and F1 score. In terms of recall and F1 score on Video 1 and 3, our S3 framework (requiring minimal learning) outperforms the current supervised state-of-the-art methods, which use the oracle summary labels for learning different deep neural networks; on average, it outperforms the previous supervised methods in terms of F1 score and achieves comparable results in terms of precision and recall.

Table 4.3: Results on the UTE dataset for conditional video summarization. The cited methods given in the first row corresponding to each video are supervised, in terms of using the oracle summarization labels for model training.

	System	Precision	Recall	F1 Score
Video-1	Mem-SeqDPP [296]	49.86	53.38	48.68
	HVN [129]	52.55	52.91	51.45
	DSAN [356]	48.41	52.34	48.52
	MixModSub	**53.26**	51.77	51.20
	S3 (Ours)	51.86	**55.24**	**52.18**
Video-2	Mem-SeqDPP [296]	33.71	62.09	41.66
	HVN [129]	38.66	62.70	**47.49**
	DSAN [356]	**46.51**	51.36	46.64
	MixModSub	36.29	62.37	45.56
	S3 (Ours)	37.24	**63.88**	46.71
Video-3	Mem-SeqDPP [296]	55.16	62.40	56.47
	HVN [129]	60.28	62.58	61.08
	DSAN [356]	56.78	61.14	56.93
	MixModSub	58.35	63.24	60.36
	S3 (Ours)	**60.51**	**65.72**	**62.66**
Video-4	Mem-SeqDPP [296]	21.39	**63.12**	29.96
	HVN [129]	26.79	54.21	**35.47**
	DSAN [356]	**30.54**	46.90	34.25
	MixModSub	17.78	53.51	26.44
	S3 (Ours)	26.54	52.94	34.97
Avg	Mem-SeqDPP [296]	40.03	**60.25**	44.19
	HVN [129]	44.57	58.10	48.87
	DSAN [356]	**45.56**	52.94	46.59
	MixModSub	41.42	57.72	45.89
	S3 (Ours)	44.04	59.44	**49.13**

4.4.3 Conditional Image Corpus Summarization

Dataset ImageNet-1k [71] is a large-scale image database that contains nearly 1.28 million training images and 50,000 validation images. The dataset is organized according to the WordNet [229] hierarchy, with each node depicting hundreds and thousands of images. In our experiments, we randomly sample 1,000 images, one from each class, from the dataset to form the development query set, and the remaining training images are used as the gallery set that needs to be summarized. We use this set for hyperparameter tuning and show qualitative results corresponding to test query images having no overlap with the development query set.

Feature Representation We use a Bidirectional Generative Adversarial Network (BiGAN) [76] for unsupervised learning of feature representations for the ImageNet database. BiGAN is an extension of GAN with a third component, an encoder, that maps the data space x to a latent space z. In addition, the discriminator is trained to discriminate between the joint samples (x, z) coming from the encoder distribution and the decoder (or generator) distribution.

The encoder architecture of BiGAN in [76] follows AlexNet [153]. We use its last convolutional layer, i.e., *conv5*, to encode the ImageNet images as we do not want to use any trained model that uses the class information. After building a sparse k-NN similarity graph ($k = 1000$) using the 1024-dimensional features and analyzing the pairwise cosine similarities, we found that most of the non-negative similarities are concentrated around 0.95. So, we reduce the feature dimension to 512 using PCA (retaining 95% of the total variance). We also used the averaged top-k accuracy of the nearest neighbors of the validation query images to further validate the feature dimension chosen. Here, we set $k \in \{10, 50, 100\}$.

Summary Generation We use a sparse facility location as the objective function for our S3 framework. In order to deal with this large-scale dataset, we use *faiss*[3], which is an efficient similarity search library from Facebook, to build a k-NN similarity graph using cosine

[3]https://ai.facebook.com/tools/faiss/

similarity. For stage one of the S3 framework, we prune the candidate set by taking top-k nearest neighbors of each $q \in Q$ and optimize Eq. (4.32) using modular approximation. In all experiments, k is set to 1000. In stage two, we condense A_Q to generate a conditional summary of 25 images.

Hyperparameter Tuning For the facility location function, we experimented with two similarity measures: (1) cosine similarity and (2) Gaussian kernel with width σ. Our hyperparameters set includes: (1) $\sigma \in \{1, 10, 50, 100, 200, 500, 1000, 10000\}$ (2) $k_1 \in \{50, 75, 100, 125, 150, 175, 200, 225, 250\}$. We find that cosine similarity with $k_1 = 225$ leads to the best relevance and diversity scores on the validation query set.

Table 4.4: Qualitative Results on the ImageNet Dataset when $|Q| = 1$. The first column shows the top-25 nearest neighbors given a query, and the second column shows the query-focused summary. Images marked with a surrounding green box and light blue box denote the query image and common images among the two columns, respectively.

Table 4.5: Qualitative Results on the ImageNet Dataset when $|Q| = 1$. The first column shows the top-25 nearest neighbors given a query, and the second column shows the query-focused summary. Images marked with a surrounding green box and light blue box denote the query image and common images among the two columns, respectively.

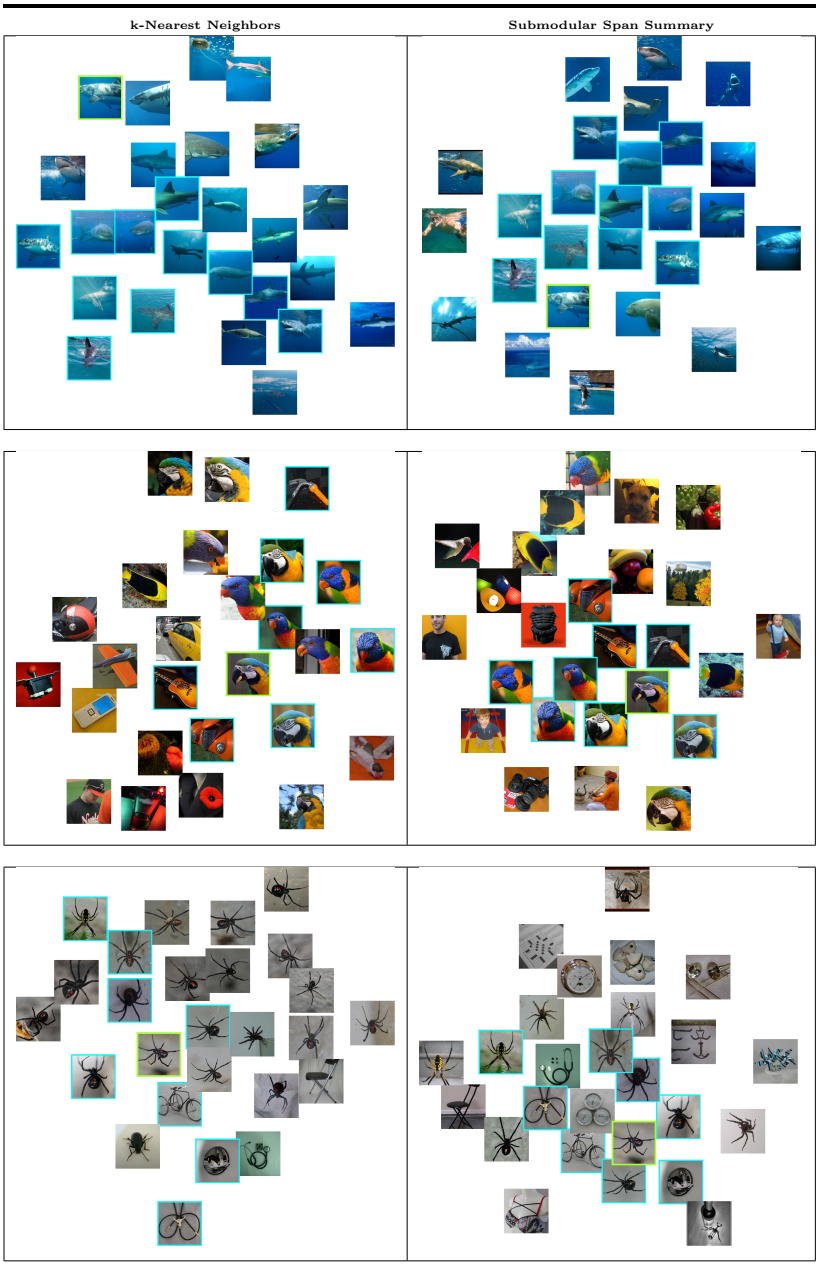


Table 4.6: Qualitative Results on the ImageNet Dataset when $|Q| = 1$. The first column shows the top-25 nearest neighbors given a query, and the second column shows the query-focused summary. Images marked with a surrounding green box and light blue box denote the query image and common images among the two columns, respectively.

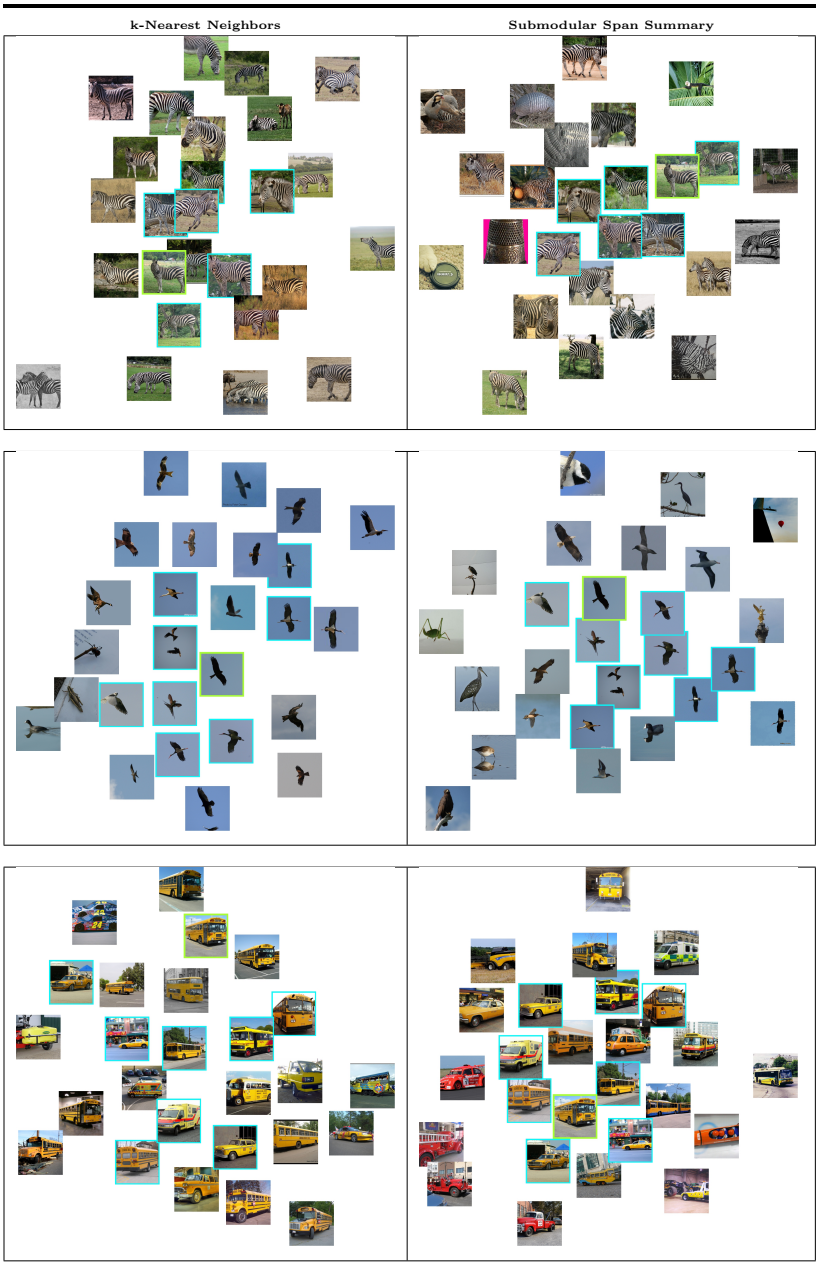


Table 4.7: Qualitative Results on the ImageNet Dataset when $|Q| = 1$. The first column shows the top-25 nearest neighbors given a query, and the second column shows the query-focused summary. Images marked with a surrounding green box and light blue box denote the query image and common images among the two columns, respectively.

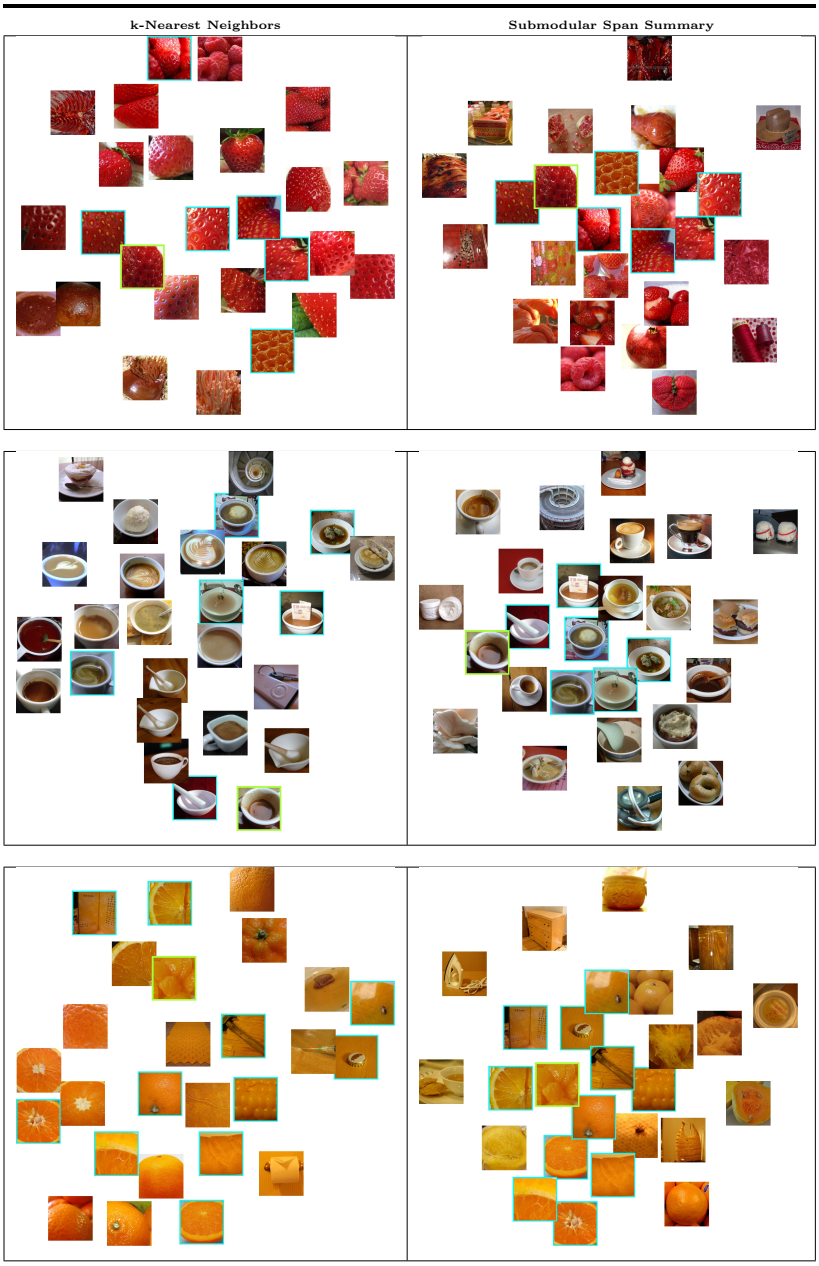


Table 4.8: Qualitative Results on the ImageNet Dataset corresponding to $|Q| > 1$

Query	Submodular Span Summary
	$\mathcal{R}(A\|Q) = 0.2,\ f(A) = 4.07 \times 10^{-3}$ 
	$\mathcal{R}(A\|Q) = 0.24,\ f(A) = 4.79 \times 10^{-3}$ 
	$\mathcal{R}(A\|Q) = 0.1,\ f(A) = 7.38 \times 10^{-3}$ 
	$\mathcal{R}(A\|Q) = 0.22,\ f(A) = 4.79 \times 10^{-3}$ 

Table 4.9: Qualitative Results on the ImageNet Dataset corresponding to $|Q| > 1$

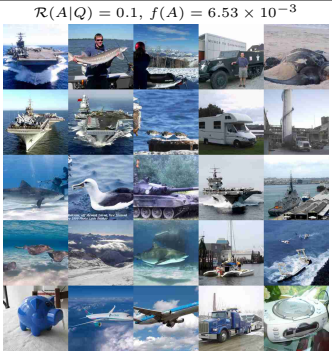
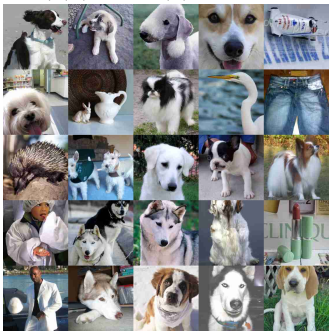
Query	Submodular Span Summary
	$\mathcal{R}(A\|Q) = 0.1,\ f(A) = 6.53 \times 10^{-3}$ 
	$\mathcal{R}(A\|Q) = 0.2,\ f(A) = 5.45 \times 10^{-3}$ 
	$\mathcal{R}(A\|Q) = 0.22,\ f(A) = 1.99 \times 10^{-3}$ 
	$\mathcal{R}(A\|Q) = 0.24,\ f(A) = 5.09 \times 10^{-3}$ 

Table 4.10: Qualitative Results on the ImageNet Dataset corresponding to $|Q| > 1$

Query	Submodular Span Summary
	$\mathcal{R}(A\|Q) = 0.08,\ f(A) = 6.62 \times 10^{-3}$ 
	$\mathcal{R}(A\|Q) = 0.14,\ f(A) = 5.96 \times 10^{-3}$ 
	$\mathcal{R}(A\|Q) = 0.22,\ f(A) = 3.64 \times 10^{-3}$ 
	$\mathcal{R}(A\|Q) = 0.08,\ f(A) = 6.13 \times 10^{-3}$ 

Evaluation To assess the quality of the query-focused summary, we propose a function, $\mathcal{R}(A|Q)$, that captures the relevance of the summary images to the query set.

$$\mathcal{R}(A|Q) = \frac{1}{|Q|} \sum_{q \in Q} \frac{1}{|A|} \sum_{a \in A} \mathbb{1}_{y_q = y_a} \tag{4.36}$$

Here, y_a denotes the label/class of the image a. In the case of single query image-based summarization, $\mathcal{R}(A|Q)$ is simply the precision of retrieving the query class in the summary. We also assess the diversity of the summary using $f(A)$, which is the sparse-facility location function valuation of summary A with respect to the ground set V.

Comparing kNN and S3 In Table 4.4, 4.5, 4.6, and 4.7, we show a comparison between the submodular span summary and the k-nearest neighbors for different query images when $|Q| = 1$. We generate a submodular span summary of 25 images and compare it side-by-side with the top-25 nearest neighbors of the query. We utilize tSNE plots to better visualize the similarity/dissimilarity among the submodular span summary images. It can be seen that the tSNE plot corresponding to k-nearest neighbors consists of clusters of similar images that are redundant among themselves, while the submodular span summary consists of diverse views of images belonging to the query class or images that share visually similar features with the query image. For example, in the first example in Table 4.5 corresponding to the query *great white shark*, the submodular span summary consists of images with diverse views of *shark*, images of different aquatic creatures as well as images of *scuba diver* inside water. Similarly, corresponding to the query *zebra* in Table 4.6, the k-nearest neighbors consist of redundant images belonging to the query class while the submodular span summary consists of diverse images of *zebra* in different backgrounds as well as images from other classes having features similar to stripes. This shows that the submodular span summary can capture different properties associated with the query set, thus leading to a diverse query-focused summary that is highly related (or redundant) to the query set.

In Tables 4.8, 4.9, and 4.10, for different query sets, we show the conditional summaries after fitting the two-dimensional tSNE vectors of their image embeddings onto a square grid

using the Jonker-Volgenant algorithm [131]. As it can be seen, apart from consisting of diverse images belonging to the query classes, the conditional summary also consists of images that share properties of both query classes. For example, given a query set comprising {*strawberry, kite*} images, the conditional summary also consists of bird images having red hues and a lizard (sharing *kite* hues) on a red flower. In the last example, where the queries comprise {*bee, daisy*}, the summary consists of two images of a *bee* on *daisy* as well as birds resting on twigs.

4.5 Summary

In this chapter, we propose and study the *submodular span problem* as an extension to the matroid span problem in terms of retrieving elements relevant to a query set. We design a minimally supervised two-stage query-focused summarization framework called S3 and show its applications for conditional data summarization of different data modalities. Our analysis and results also shed light on the feasibility and scalability of the modular approximation algorithm for the polymatroid submodular span problem. Our results on three real-world datasets, DUC 2005-2007, UT-Egocentric video dataset, and ImageNet, verify the significance of the two stages (retrieval followed by summarization) of the S3 framework.

Chapter 5

SUBMODULARITY FOR DATA-EFFICIENT IN-CONTEXT LEARNING USING LLMS

5.1 Introduction

Pretrained large language models (LLMs) [136, 36, 54] have become foundational for a wide range of Natural Language Processing (NLP) tasks, demonstrating impressive success across various domains [31, 37] through in-context learning (ICL) [78]. As described in Sec. 3.1.3, ICL enables these pretrained LLMs to perform new tasks by using task-specific prompts containing a limited number of input-output demonstrations (also referred to as shots, exemplars, or prompts) in the natural language format. This approach facilitates deployment across different downstream tasks and reduces the need for labeled downstream training data since ICL does not require any task-specific training.

Recent studies [201, 311, 220] show that providing exemplars most relevant to the current input instance is beneficial. Moreover, in [396, 213, 202], it is observed that LLMs attend more to the exemplars that are closer in the sequence to the input instance. Therefore, to achieve the best performance of ICL, the selection of exemplars and their ordering in the LLM prompt are crucial.

In practice, an extensive collection of unlabeled exemplars is easily available (e.g., posts and discussions on forums like Stack Exchange or user-generated content on social media platforms). However, to use them effectively as in-context exemplars, one needs to annotate them, which can be exceptionally costly at scale. To annotate and select the exemplars optimally for a given target task, we follow the two-stage approach shown in Fig. 5.1: (1) **Exemplar Annotation**: select a subset of exemplars for annotation under a fixed budget (performed only once) and (2) **Exemplar Retrieval**: identify limited-sized exemplars in an

ordering that are most influential for a given input instance from the annotated subset of exemplars. Intuitively, for the first stage, we aim to find the subset with maximal diversity so that, given any input, we can find corresponding labeled exemplars. For the second stage, in addition to the diversity requirement similar to the first stage, we emphasize the relevance of the exemplars to the given input query and order exemplars so that their relevance to the input query decreases as the exemplars are farther away from the input instance.

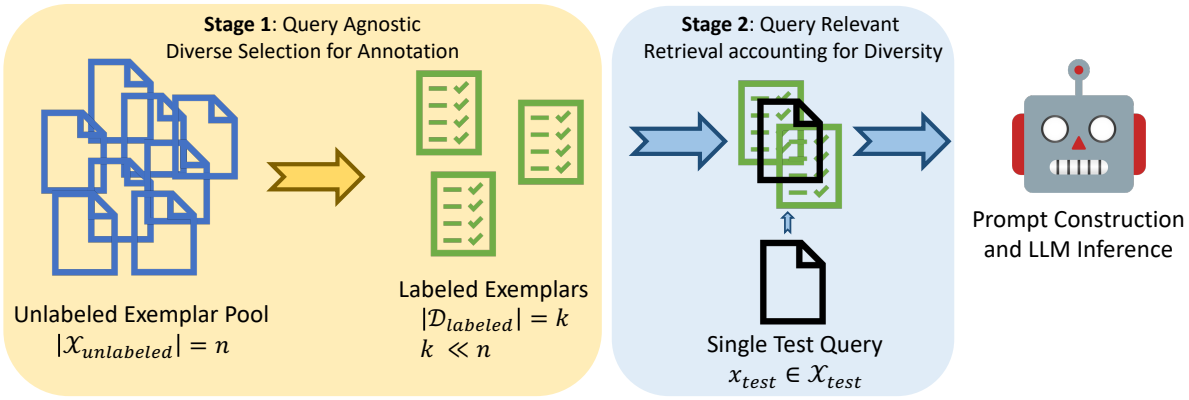


Figure 5.1: Workflow of our proposed framework for data-efficient In-Context Learning using LLMs. For the first stage, we use cardinality-constrained submodular optimization to identify diverse exemplars for annotation. For the second stage, given a test query, we use Submodular Span Summarization (S3 from Chapter 4) to find a diverse set of labeled exemplars that are most relevant to the query.

In this chapter, we propose a framework *Div-S3* based on submodular optimization that unifies the above-mentioned two stages. For Exemplar Annotation, we model the problem as a submodular optimization problem under a cardinality constraint to find as **Div**erse a subset as possible within a budget. For Exemplar Retrieval, we formalize the problem as a Submodular Span Summarization (**S3**) problem [155] (described in Chapter 4) with a knapsack constraint, which finds a diverse subset most relevant to the input query under a token length limit. Also, we naturally order the resulting exemplars based on the gains represented by the submodular function. The name of our proposed framework *Div-S3* captures the op-

timization objectives used for both the exemplar annotation (*Div*) and exemplar retrieval (*S3*) stages. In Fig. 5.2, we show a sample test query where using *Div-S3* for exemplar selection leads to a more diverse and query-relevant exemplar set.

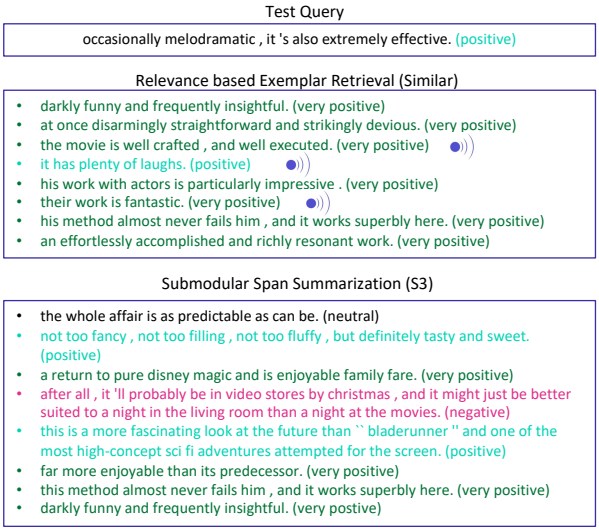


Figure 5.2: A sample test query from the SST-5 dataset with its corresponding set of exemplars selected using *Similar* (focusing only on query-based relevance) and *S3* (focusing on both relevance and representativeness) from a limited exemplars pool. Exemplars are colored based on their class names. We use echo lines to denote the redundant exemplars chosen by *Similar* and used as a part of the input context during ICL.

Our framework is general, as any submodular function can be plugged into our method. For models beyond LMs, e.g., for text-image multi-modality models, we may use pre-existing submodular functions that are powerful for expressing diversity in the image domain. In addition, we account for relevance, diversity, and ordering for the exemplar retrieval stage, where one or two aspects typically get overlooked by previous methods.

This chapter is based on the work published in: **Lilly Kumari**, Shengjie Wang, Arnav Das, Tianyi Zhou, and Jeff Bilmes. *An End-to-End Submodular Framework for Data-Efficient In-Context Learning. In Findings of the Association for Computational Linguistics: NAACL, 2024.*

5.2 Notations

ICL as a few-shot learning method [36] enables LLMs to adapt to new tasks by utilizing only a small number of context demonstration examples. Formally, given an LLM f_θ parameterized by θ, a test query $(x_{\text{test}}, y_{\text{test}})$, and a set of k labeled demonstrations $\mathcal{D}_{\text{context}} = \{(x_i, y_i)\}_{i=1}^{k}$, the probability of generating the target label y_{test} using f_θ is formulated as:

$$p(y_{\text{test}}|c, x_{\text{test}}) = f_\theta(\mathcal{V}(y_{\text{test}})|c, \mathcal{T}(x_{\text{test}}, \cdot)), \tag{5.1}$$

where $\mathcal{V}(\cdot)$ is a verbalizer that maps the task labels y to words $\mathcal{V}(y)$ in the LLM's vocabulary. $\mathcal{T}(\cdot)$ denotes the process of wrapping up the input using the instruction prompt template. c is the input context formed by concatenating the k in-context demonstrations from $\mathcal{D}_{\text{context}}$, i.e., $c = \mathcal{T}(x_1, y_1) \oplus \mathcal{T}(x_2, y_2) \oplus \ldots \oplus \mathcal{T}(x_k, y_k)$ where $\oplus$ denotes concatenation. Fig. 5.3 shows an example of the input sequence formatting for a specific case of performing in-context learning for a translation task.

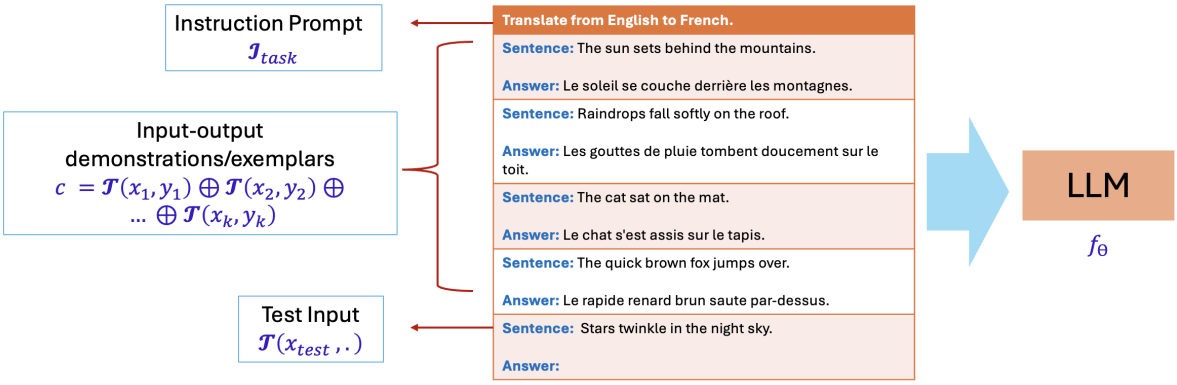


Figure 5.3: Input Sequence Construction for In-context Learning

5.3 Related Work

In this section, we outline recent studies investigating ICL, specifically focusing on works in the context of exemplar annotation selection and retrieval. With the introduction of ICL in the GPT-3 paper [36], numerous studies have emerged trying to understand the mechanics of ICL [357, 230, 41, 331, 345, 65, 111, 188] and its inherent strengths and limitations [343, 212, 127, 160, 370].

A recent work by [230] shows that ICL fails to learn label relationships from the in-context exemplars, as its performance diminishes only slightly when substituting the demonstration labels with random labels. However, recent work [149] has discovered that LLMs indeed utilize the in-context label information, and label relationships learned during the pre-training phase have a more lasting effect than in-context demonstrations. Also, prior works [396, 201, 213] have demonstrated the sensitivity of ICL to the choice and order of in-context exemplars in the final LLM prompt. This has prompted investigations into determining which samples should be included in the exemplar pool and how to select exemplars at test time effectively.

5.3.1 Active Learning for Exemplar Annotation

We review existing active learning-based data selection algorithms in Sec. 3.1.1. Here, we focus on recent active learning methods designed for exemplar annotation in in-context learning (ICL). This can be viewed as performing a single iteration of pool-based active learning where the base model weights are not updated. Recent works [311, 391, 147] have explored ICL under a fixed annotation budget. VoteK [311], described in detail in Sec. 5.4, uses a combination of graph-based and uncertainty sampling-based approaches to select diverse samples for annotation and relies on the model's confidence for the uncertainty sampling. Furthermore, another work [391] formulates exemplar selection as an iterative decision problem and proposes a reinforcement learning algorithm to train policies for active exemplar selection.

DataModels proposed in [42] trains a linear regression model to predict the LLM's outcome given an exemplar and its position in the final prompt. The CondAcc method proposed in the same work scores each exemplar by its dev-set ICL performance when combined with other randomly sampled in-context exemplars. However, both methods require multiple rounds of inference using the target LLM and can be pretty costly in practice. Another recent work [220] highlights that uncertainty sampling for selecting annotated exemplars results in inferior performance and is described in detail in Sec. 5.4. In contrast, similarity-based sampling performs better, albeit with the drawback of increasing the annotation budget, as each test query is considered independently.

To find supporting examples for ICL, authors in [182] employ a two-stage framework: (1) a progressive filtering stage, which extracts informative examples via a new metric based on LLMs' feedback, and (2) a diversity-guided beam search method to select the final supporting examples.

5.3.2 Exemplar Retrieval

This section primarily covers prior works studying exemplar retrieval at test time, a crucial component of ICL. In Sec. 3.1.3, we highlight the importance of test-time input context selection in large language models (LLMs) from a resource-efficiency perspective. Most existing ICL methods that do not involve fine-tuning rely on cosine similarity-based ranking for exemplar retrieval [282, 201, 311, 220, 352]. This is described in Sec. 5.4. However, these methods treat each exemplar independently, failing to capture the interactions and diversity among selected exemplars, which can limit overall retrieval effectiveness.

Among the learning-based methods, Rubin et al. [282] train a lightweight retriever model using a contrastive learning objective to score each exemplar independently. CEIL proposed in [366] similarly trains a retriever by utilizing a DPP to score a subset of exemplars. They show that selecting a set of diverse and non-redundant exemplars is vital for the overall performance of ICL. However, to learn the DPP retriever, CEIL needs to create training data by scoring multiple sets of exemplars formatted in a prompt template using the inference

LLM, thus adding to the computational costs besides the costs associated with fine-tuning.

5.4 Set Function Optimization View of Prior Work

Exemplar Annotation:

VoteK [311] is an exemplar selection method designed to construct informative and diverse subsets of unlabeled data for the purpose of annotation. It operates in two stages: the first stage focuses on diversity-based selection using a scoring function defined over a similarity graph, while the second stage focuses on uncertainty-based selection using a heuristic confidence-based strategy. It allocates its fixed annotation budget M proportionally across these stages: 10% of the budget is used in the first stage, and the remaining 90% is allocated to the second stage.

VoteK - Stage 1 (Diversity-based Selection via Scoring Function): In the first stage, VoteK constructs a kNN-based similarity graph $G = (V, E)$ using sentence embeddings, where $V = \mathcal{X}^0_{\text{unlabeled}} \cup \mathcal{X}^0_{\text{labeled}}$ denotes the set of all samples. Here, $\mathcal{X}^t_{\text{labeled}}$ denotes the set of examples chosen for annotation at step t, and $\mathcal{X}^t_{\text{unlabeled}}$ denotes the current unlabeled set, such that $|\mathcal{X}^t_{\text{labeled}}| + |\mathcal{X}^t_{\text{unlabeled}}| = |V| = n$. At $t = 0$, $\mathcal{X}^0_{\text{unlabeled}} = V$ and $\mathcal{X}^0_{\text{labeled}} = \emptyset$.

A scoring function is defined to quantify the utility of selecting a candidate unlabeled sample based on its impact on the remaining unlabeled set. Formally, the utility of adding a new sample $u \in \mathcal{X}^t_{\text{unlabeled}}$ to the labeled set is computed as:

$$\text{score}(u; \mathcal{X}^t_{\text{labeled}}) = \sum_{v \in \{v \mid (v,u) \in E,\, v \in V \setminus \mathcal{X}^t_{\text{labeled}}\}} s(v), \tag{5.2}$$

where $s(v) = \rho^{-|\{\ell \in \mathcal{X}^t_{\text{labeled}} \mid (v,\ell) \in E\}|}$, with $\rho > 1$, is an exponentially decaying function based on the number of labeled neighbors of v in the similarity graph E.

Since the score function $s(\cdot)$ decreases as more samples are labeled, and the scoring objective in Eq. (5.2) aggregates over the unlabeled set—which itself shrinks over time—the overall utility of adding a new sample to the labeled set naturally diminishes as the labeled set $\mathcal{X}^t_{\text{labeled}}$ grows. This behavior reflects a *diminishing returns property*, where the marginal gain of adding a new sample decreases with the size of the labeled set.

Although the paper does not formally define a set function $f : 2^V \to \mathbb{R}$, the scoring function in Eq. (5.2) can be interpreted as a form of conditional gain (Sec. 2.3.2), denoted $f(u\ ; \mathcal{X}^t_{\text{labeled}}) = \text{score}(u\ ; \mathcal{X}^t_{\text{labeled}})$, where the utility of selecting a new sample depends on the current labeled context. The use of a semicolon here reflects that $f(u\ ; \mathcal{X}^t_{\text{labeled}})$ is not assumed to be a marginal gain derived from a well-defined set function, but rather a context-dependent utility function.

While the authors do not explicitly characterize or prove this *diminishing returns property*, the scoring mechanism unknowingly leverages it, ensuring that redundant selections are naturally discouraged. However, this does not imply that such a set function f necessarily exists. Whether there exists a function f such that $f(u \mid \mathcal{X}^t_{\text{labeled}}) = f(\mathcal{X}^t_{\text{labeled}} \cup \{u\}) - f(\mathcal{X}^t_{\text{labeled}})$ exactly corresponds to $\text{score}(u; \mathcal{X}^t_{\text{labeled}})$ remains an open question.

VoteK - Stage 2 (Uncertainty-based Heuristic Selection): After an initial subset is selected using the scoring function, the model is prompted with the remaining examples in $\mathcal{X}^T_{\text{unlabeled}}$ (here $T = 0.1 \times M$) to obtain *confidence scores* over them. The unlabeled set is partitioned into M equal-sized buckets based on confidence scores, and the most confident $M/10$ buckets are discarded. From each of the remaining $9M/10$ buckets, the highest-confidence sample is selected, ensuring coverage across different uncertainty levels.

However, this uncertainty-based selection strategy considers only the uncertainty of individual samples, treating each selection as independent and without regard to its contribution in the context of the already selected labeled set. Capturing diversity and uncertainty using two distinct phases in this order presents a potential limitation: it fails to jointly optimize for both criteria. As a result, the confidence-based stage may select samples that, while uncertain, are redundant with previously selected examples, leading to inefficiencies in the annotation process.

Uncertainty-based exemplar selection proposed in [220] ranks samples in the unlabeled set $\mathcal{X}_{\text{unlabeled}}$ based on their *perplexity* computed using the inference LLM. It uses perplexity as a proxy for uncertainty under the assumption that higher perplexity examples are more informative for downstream learning.

Formally, let $\mathcal{X}_{\text{unlabeled}} = \{x_1, x_2, \ldots, x_n\}$ denote the candidate unlabeled examples, and let $u(x_i) \in \mathbb{R}_{\geq 0}$ be the perplexity score for example x_i. The objective of this single iteration active learning can be seen as selecting a subset $\mathcal{X}_{\text{labeled}} \subseteq \mathcal{X}_{\text{unlabeled}}$ of size at most k that maximizes the total uncertainty score:

$$\mathcal{X}_{\text{labeled}} \in \underset{\mathcal{X} \subseteq \mathcal{X}_{\text{unlabeled}}, |\mathcal{X}| \leq k}{\text{argmax}} \sum_{x_i \in \mathcal{X}} u(x_i) \tag{5.3}$$

This defines a *modular utility function* $f(\mathcal{X}) = \sum_{x_i \in \mathcal{X}} u(x_i)$, where the contribution of each item is additive and independent of others. That means that the optimal subset can be obtained via a simple *top-k sorting operation* over the perplexity scores, and the selected examples may be redundant amongst themselves, leading to inefficiencies in the data annotation stage.

Exemplar Retrieval:

Similarity-based exemplar selection [282, 201] at test time involves computing the similarity between the test query and a pool of candidate labeled exemplars using sentence-level representations obtained via a pretrained sentence encoder (e.g., Sentence-BERT). *Cosine similarity* is commonly used to compute the similarity between the query and exemplar embeddings.

Let $q \in \mathbb{R}^d$ denote the embedding of the test query, and let $\mathcal{E} = \{e_1, e_2, \ldots, e_n\}$ denote the pool of candidate exemplars, where each $e_i \in \mathbb{R}^d$ is the sentence embedding of an exemplar $x_i \in \mathcal{X}_{\text{labeled}}$. The utility score of each exemplar is computed as the cosine similarity between the test query and the exemplar:

$$u(x_i) = \text{sim}(q, e_i) = \frac{q^\top e_i}{\|q\| \cdot \|e_i\|}. \tag{5.4}$$

Exemplar selection is then formulated as a subset selection problem, where the goal is to choose a subset $\mathcal{X}_{\text{selected}} \subseteq \mathcal{X}_{\text{labeled}}$, subject to a cardinality constraint $|\mathcal{X}_{\text{selected}}| \leq k$, that maximizes the overall utility:

$$\mathcal{X}_{\text{selected}} \in \underset{\mathcal{X} \subseteq \mathcal{X}_{\text{labeled}},\ |\mathcal{X}| \leq k}{\text{argmax}} \sum_{x_i \in \mathcal{X}} u(x_i). \tag{5.5}$$

This selection can also be subject to knapsack constraint such that $\sum_{x_i \in \mathcal{X}} c(x_i) \leq b$ where $c(x_i)$ denotes the cost associated with the input exemplar x_i in terms of tokens count and b denotes the context window of the inference LLM.

This defines a *modular utility function* $f(\mathcal{X}) = \sum_{x_i \in \mathcal{X}} u(x_i)$, where each exemplar's contribution to the score is independent of the others. The marginal gain of including any exemplar x_i is always $u(x_i)$, regardless of the composition of the rest of the set. As a result, the selection reduces to a simple top-k sorting operation over the test query similarity scores, ignoring pairwise interactions and redundancy amongst the selected exemplars.

5.5 Div-S3

Div-S3 addresses two essential questions pertinent to data and label-efficient ICL:

(1) *Given an unlabeled pool of target task samples $\mathcal{X}_{unlabeled}$, how can we identify the most informative set of examples $\mathcal{X}_{labeled}$ to annotate and use as demonstrations for the target tasks' queries?*

This question focuses on the data and label efficiency aspect of ICL using LLMs. Similar to [220], this stage of "**Exemplar Annotation**" can be viewed as one iteration of pool-based active learning aimed at choosing a set of the most informative and diverse demonstrations/exemplars for annotation. After augmenting the samples in $\mathcal{X}_{\text{labeled}}$ with their respective labels, we denote the resultant labeled set as $\mathcal{D}_{\text{labeled}}$.

(2) *Given a query x_{test}, how can we select the most relevant and non-redundant labeled exemplars from $\mathcal{D}_{labeled}$?*

This addresses the "**Exemplar Retrieval**" stage of ICL, where the goal is to retrieve the most relevant exemplars from an annotated pool of exemplars given a particular test query.

5.5.1 Exemplar Annotation

In this stage, we assume access to a large pool of unlabeled samples denoted by $\mathcal{D}_{\text{unlabeled}}$ belonging to the target task. Unlike the similarity-based active learning approach explored in [220], we do not assume access to the full test set during the exemplar annotation phase.

Performing similarity-based sampling of exemplars for each test query individually would not be an efficient use of labeling resources, as the annotation budget could increase linearly with the number of test queries considered, overlooking the shared information among exemplars.

Given $\mathcal{X}_{\text{unlabeled}}$ of size n, this stage aims to select $\mathcal{X}_{\text{labeled}} \subseteq \mathcal{X}_{\text{unlabeled}}$ such that $|\mathcal{X}_{\text{labeled}}| = k$ and $k << n$. This process is performed only once and can be viewed as one iteration of pool-based active learning [292]. As this stage determines the examples utilized as in-context demonstrations in the second stage, we aim to curate a set of diverse and representative samples that can comprehensively cover the target task space. To achieve this, we utilize a submodular function instantiated on the entire ground set $V = \mathcal{X}_{\text{unlabeled}} = \{x_i\}_{i=1}^{n}$.

While our framework applies to any submodular functions, for the experiments in this chapter, we use a popular submodular function called *facility location* [58, 232] described in Sec. 2.1.2, which is closely related to but more general than k-medoid clustering.

To obtain our $\mathcal{D}_{\text{labeled}}$ set, we first use a pre-trained language model g_θ (sentence-BERT [270] in this work) to compute text embeddings of each sample x_i in the unlabeled pool $\mathcal{X}_{\text{unlabeled}}$. Using the text embeddings of samples in $\mathcal{X}_{\text{unlabeled}}$, we compute a similarity matrix using a tuned similarity kernel, use it to instantiate the facility location function $f(\cdot)$, and then maximize the submodular objective using the greedy algorithm [239, 231] (described in Algorithm 1 in Sec. 2.2.1) to obtain our final set of demonstrations for annotation as follows:

$$\mathcal{X}_{\text{labeled}} \in \underset{A \subseteq \mathcal{X}_{\text{unlabeled}}, |A| \leq k}{\operatorname{argmax}} f(A) \tag{5.6}$$

5.5.2 Exemplar Retrieval

After obtaining a set $\mathcal{D}_{\text{labeled}}$ of diverse and representative annotated exemplars, the next step is to select exemplars from $\mathcal{D}_{\text{labeled}}$ given a test query x_{test}. Previous studies in ICL [282, 201, 311, 220] have employed cosine similarity-based ranking to select exemplars that are most similar to the test query. However, when each example's relevance (or similarity) to the test query is considered independently, it may yield relevant but similar exemplars, carrying redundant information that is wasteful for inference.

To reduce the redundancy amongst the selected in-context exemplars, we formalize the exemplar retrieval stage as a conditional submodular subset selection problem. We use the same submodular function utilized in the prior stage for exemplar annotation, but unlike the previous stage, which employs a generic summarization approach as shown in Eq. (5.6), this stage focuses on conducting query-based (or conditional) summarization, aiming to summarize the labeled set $\mathcal{X}_{\text{labeled}}$ given a query set containing x_{test}. When performing ICL using LLMs, the query x_{test} is the test input instance.

In this work, we utilize the two-phase method named **Submodular Span Summarization (S3)** [155] described in Chapter 4 to perform query-focused summarization of an annotated pool given a test query. Our objective is to obtain a set of exemplars that are not only relevant to the test query but also encompass diverse aspects crucial for aiding the LLM in the target task.

Phase 1 of S3 targets selecting a relatively large subset relevant to the query set. Mathematically, given a ground set V that includes the query set Q and the data being summarized V_Q (where $V_Q = V \setminus Q$), and a submodular function f defined on the entire ground set V, we minimize the conditional gain $f(A|Q) := f(A \cup Q) - f(Q)$ subject to a lower-bound cardinality constraint:

$$\min_{A \subseteq V_Q} f(A|Q) \;\; \text{s.t.} \;\; |A| \geq k_1 \tag{5.7}$$

Similar to Chapter 4, we utilize a modular approximation of $f(A|Q)$, denoted as $m_Q(A) = \sum_{a \in A} f(a|Q)$, to optimize Eq. (5.7). Note that $m_Q(A) \geq f(A|Q)$ is an upper bound of $f(A|Q)$.

For the Exemplar Retrieval stage of ICL, the data to be summarized V_Q is the annotated pool of exemplars $\mathcal{X}_{\text{labeled}}$. We denote our solution of S3's Phase 1 as A_Q, essentially the annotated exemplars relevant to the input query Q.

Phase 2 of S3 focuses on diverse aspects of various relevant exemplars, ensuring the final set of exemplars for ICL is both relevant and non-redundant. We summarize the S3 Phase 1 resultant set A_Q by performing submodular maximization subject to a cardinality

(or a knapsack) constraint as shown in Eq. (5.8):

$$\max_{A \subseteq A_Q, |A| \leq k_2} f(A) \tag{5.8}$$

In this chapter, we use the same submodular function for the Exemplar Annotation stage and the two phases of S3 for the Exemplar retrieval stage. Our framework is general, and different submodular functions can apply if various diversity properties are desirable.

Since the final set of exemplars selected will be used as in-context demonstrations, we can also apply a knapsack constraint for the constraint associated with Eq. (5.8).

$$\max_{A \subseteq A_Q} f(A) \;\; \text{s.t.} \;\; \sum_{a \in A} c(a) \leq b \tag{5.9}$$

The budget b can be set to the difference between the pre-set context window of the inference LLM and the token length of the formatted test query. The cost $c(a)$ denotes the cost associated with exemplar a that can be set to the token length of the instruction formatted exemplar a, i.e., $\mathcal{T}(x_a, y_a)$. Note that the knapsack constraint generalizes the cardinality constraint: for cardinality constraints, $c(a)$ is simply 1, and the budget is k_2.

To optimize Eq. (5.9), we use the modified greedy algorithm [194] described in Algorithm 3 (in Sec. 2.2.1) with a $(1 - 1/\sqrt{e})$ constant factor approximation factor under certain conditions. The modified greedy at each iteration i selects exemplar s_i with the largest ratio of objective conditional gain to the scaled cost, i.e., $s_i = \text{argmax}_{s \in A_Q} \frac{f(A_{i-1} \cup s) - f(A_{i-1})}{c(s)^r}$ if $\sum_{s \in A_{i-1} \cup s_i} c(s) \leq b$.

Since both stages discussed in Sec. 5.5.1 and 5.5.2 involve certain hyperparameters such as the similarity kernel, kernel width (in case of using the RBF kernel), the budget of Phase 1 of S3 (k_1), budget associated with Phase 2 of S3 (k_2 when using a cardinality constraint), or the scaling factor (r when using a knapsack constraint), we utilize a separate validation set for hyperparameter tuning . Our approach, thus, needs minimal supervision and is learning-free, as it does not involve fine-tuning the inference LLM on any task-specific task.

For all the experiments in this chapter, we use *submarine* [27] for submodular optimization.

5.5.3 Complexity Analysis

Exemplar Annotation To select the set of exemplars to label, we are required to maximize the facility location function using greedy selection. This step requires (1) constructing a pairwise similarity matrix and (2) applying greedy selection. Given $|\mathcal{X}_{\text{unlabeled}}| = n$ and $|\mathcal{X}_{\text{labeled}}| = k$, constructing the pairwise similarity matrix requires $\mathcal{O}(n^2)$ operations while greedy selection requires $\mathcal{O}(nk)$, yielding a final complexity of $\mathcal{O}(n^2 + nk)$. In practice, the quadratic cost of computing the similarity matrix can be reduced by utilizing sparse matrices, while greedy selection can be sped up significantly by using a priority queue [231]. The exemplar annotation cost is incurred only once for each task.

Exemplar Retrieval At inference time, we retrieve exemplars annotated in the first stage that are relevant to a particular query set by using S3. The first phase of S3 requires minimizing a modular function $m_Q(A)$ to select k_1 query-relevant samples from a set of k labeled exemplars. Thus, this phase has a time complexity of $\mathcal{O}(k + k \log k_1)$.

The second phase of S3 summarizes the k_1 samples down to a diverse set of k_2 samples by maximizing a facility location function subject to a knapsack constraint. Similar to the process of exemplar annotation, this phase also requires constructing a pairwise similarity matrix ($\mathcal{O}(k_1^2)$, though this can simply be reused from the exemplar annotation phase as we do not alter the similarity metric. Thus, the complexity of this phase is $\mathcal{O}(k_1 k_2)$, which is incurred by performing greedy selection.

In our experiments, we focus on the low-data regime and set the annotation budget, i.e., k, as 100 in the active-learning setting. As can be seen in Sec. 5.6.2, $k1$ is often less than 30, making the entire process utilizing $S3$ for exemplar retrieval very efficient and significantly faster than the LLM inference call by several orders of magnitude.

5.6 Experiments

To demonstrate the effectiveness of our proposed framework for exemplar annotation and retrieval for performing ICL, we conduct experiments over a diverse set of 7 NLP datasets

using a suite of five different LLMs as in-context learners. Since our proposed framework *Div-S3* does not perform any task-specific LLM fine-tuning, we only compare it to existing learning-free ICL methods to ensure a fair comparison.

Datasets We evaluate *Div-S3* on the following 7 NLP datasets, which cover five distinct tasks:

1. **SST-5** [306]: This dataset involves sentiment classification of movie reviews into five distinct sentiment categories: very negative, negative, neutral, positive, and very positive.

2. **SST-2** [306]: Similar to SST-5, this dataset also deals with sentiment classification into positive and negative labels.

3. **RTE** [25]: The Recognizing Textual Entailment dataset focuses on discerning textual entailment and belongs to a broader scope of tasks studied under Natural Language Inference. Given a pair of sentences, the goal is to determine whether the premise (also called text) entails (or implies) the hypothesis.

4. **MRPC** [75]: This dataset deals with paraphrase detection task where given a pair of sentences, the objective is to determine whether they are semantically equivalent.

5. **TREC** [184]: This dataset involves a question classification task. In this work, we focus only on the six coarse labels: Abbreviation, Entity, Description, Human being, Location, and Numeric value.

6. **DBpedia** [172]: Here, we have a topic classification dataset constructed by selecting 14 non-overlapping classes from the base DBpedia 2014. The 14 different classes are as follows: company, educational institution, artist, athlete, office holder, means of transportation, building, natural place, village, animal, plant, album, film, and written work.

7. **HellaSwag** [380]: In this case, we have another NLI dataset studying grounded commonsense reasoning. It consists of multiple-choice questions with four answer choices. Three out of four choices are incorrect and designed in an adversarial way to deceive machines without misleading humans.

In Table 5.1, we provide dataset statistics of 7 datasets described above in terms of the size of the train/validation/test splits.

Table 5.1: Datasets Statistics of different ICL datasets

Dataset	Size of train set	Size of validation set	Size of test set
SST-5	8544	1101	2210
SST-2	67349	872	-
RTE	2490	277	-
MRPC	3668	408	-
TREC	5452	-	500
DBPedia	560000	-	70000
HellaSwag	39905	10042	-

Prompt template For the majority of tasks, we use the same prompt templates as [311]. In Table 5.2, we provide the template used for each task explored in this work.

Models We evaluated our proposed framework and compared baseline methods for ICL on five LLMs of varying complexity belonging to the GPT [259, 36] and OPT [389] families. Specifically, the largest model that we used as the inference LLM is GPT-J-6B [335], and from both GPT-Neo [29] and OPT families, we used their respective 2.7B and 1.3B variants.

Baselines We compare *Div-S3* to the learning-free baselines listed below. In all listed methods, the first part of the name preceding the hyphen indicates the strategy used for exemplar annotation (stage 1), while the later part denotes the strategy used for exemplar retrieval (stage 2). The second stage method named "*Similar*" uses cosine similarity-based ranking to extract the most relevant exemplars at test time.

Table 5.2: Templates of different tasks. Text marked in blue denotes the manual instruction template. Depending on the task, placeholders $\langle X \rangle$, $\langle Y \rangle$, and $\langle Z \rangle$ will be replaced by their available components.

Dataset	Prompt Template	Class names (Verbalizer)
SST-5	How do you feel about the following sentence? $\langle X \rangle$ answer:	very negative, negative, neutral, positive, very positive
SST-2	$\langle X \rangle$ It was	terrible, great
RTE	$\langle X \rangle$ question: $\langle Y \rangle$. True or False? answer:	true, false
MRPC	Are the following two sentences 'equivalent' or 'not equivalent'? $\langle X \rangle$ $\langle Y \rangle$ answer:	not equivalent, equivalent
TREC	Categories: Description, Entity, Abbreviation, Human, Numeric, Location What category best describes: $\langle X \rangle$ Answer:	description, entity, abbreviation, human, numeric, location
DBpedia	title: $\langle X \rangle$; content: $\langle Y \rangle$	company, educational institution, artist, athlete, office holder, mean of transportation building, natural place, village, animal, plant, album, film, written work
HellaSwag	The topic is $\langle X \rangle$. $\langle Y \rangle$ $\langle Z \rangle$	4 answer choices provided

1. **Random-Similar**: randomly selects exemplars from the unlabeled pool for annotation and then uses Similar for exemplar retrieval.

2. **Random-S3**: randomly selects exemplars during the annotation phase and then uses S3 to select relevant exmplars during the retrieval phase.

3. **VoteK-Similar** [311]: uses graph-based method named VoteK to select $k/10$ samples which are diverse in the feature space used. For selecting the remaining $9k/10$ samples, the inference LLM is used to compute the average log probability over the generated output to select diverse exemplars in terms of confidence scores. For the retrieval stage, Similar is used.

4. **Div-Similar** [58, 232, 20]: maximizes a submodular facility location objective to select a subset of Diverse and representative samples for annotation. This strategy differs from *Div-S3*, where we use Submodular Span Summarization for exemplar retrieval instead of Similar.

5. **Div-MixModSub**: uses a facility location-based submodular maximization to select Diverse exemplars for annotation. Similar to [155], we consider another baseline named MixModSub, which uses a mixture of a submodular function (facility location) and a modular function (similarity scores) to balance representativeness and query-relevance in the final retrieved set jointly.

We report average performance across three runs for the following baselines: Random-Similar, Random-S3, and VoteK-Similar.

5.6.1 Results

We compare *Div-S3* to the baselines discussed in Sec 5.6 in Tables 5.3, 5.4, and 5.5 using GPT-J-6B, GPT-Neo, and OPT models respectively. We fix the annotation budget as 100, i.e., $|\mathcal{D}_{\text{labeled}}| = 100$ for all tasks and methods. For Div-S3, when using a knapsack constraint,

Table 5.3: ICL performance on different NLP datasets using GPT-J-6B as our inference LLM. Here, we compare *Div-S3* against the baselines described in Sec. 5.6. Our proposed methods are marked with an asterisk (*).

Method	SST-5	SST-2	RTE	TREC	MRPC	HellaSwag	DBPedia	Average
Random-Similar	47.18	88.15	57.40	71.00	63.32	66.09	90.22	69.05
VoteK-Similar	44.30	89.19	52.43	71.09	63.91	**66.48**	89.51	68.13
Div-Similar	45.02	90.48	53.07	77.60	67.89	66.14	90.94	70.16
Div-MixModSub	45.93	90.25	54.87	76.80	67.89	66.18	**91.06**	70.43
Random-S3	46.73	88.23	58.24	70.80	63.32	66.18	89.49	69.00
Div-S3 (cardinality)*	49.59	**91.28**	**60.65**	80.00	**68.63**	66.32	89.87	**72.33**
Div-S3 (knapsack)*	**49.73**	-	59.57	**80.60**	67.40	66.44	90.69	72.24

Table 5.4: ICL performance on four candidate datasets using two *GPT-Neo* models. Our proposed methods are marked with an asterisk (*).

Method	SST-5	SST2	RTE	TREC
Random-Similar	41.01	70.26	53.91	62.60
VoteK-Similar	39.97	64.71	54.17	62.11
Div-S3 (cardinality)*	40.36	**77.64**	53.79	66.40
Div-S3 (knapsack)*	**41.49**	-	**54.87**	**68.80**

(a) GPT-Neo 2.7B

Method	SST-5	SST-2	RTE	TREC
Random-Similar	35.78	78.63	48.26	55.00
VoteK-Similar	**39.06**	80.34	48.70	52.99
Div-S3 (cardinality)*	37.29	**82.80**	**52.71**	62.80
Div-S3 (knapsack)*	38.51	-	48.74	**65.00**

(b) GPT-Neo 1.3 B

Table 5.5: ICL performance on four datasets using two *OPT* models.

Method	SST-5	SST2	RTE	TREC
Random-Similar	37.29	74.27	51.38	58.27
VoteK-Similar	36.72	73.96	**53.52**	50.39
Div-S3 (cardinality)*	37.74	**79.82**	53.43	64.40
Div-S3 (knapsack)*	**38.69**	-	51.26	**65.60**

(a) OPT 2.7B

Method	SST-5	SST-2	RTE	TREC
Random-Similar	**32.64**	81.69	52.47	63.53
VoteK-Similar	28.52	81.38	52.73	61.07
Div-S3 (cardinality)*	32.08	**86.12**	53.79	**67.60**
Div-S3 (knapsack)*	31.54	-	**54.87**	67.20

(b) OPT 1.3 B

the budget (in terms of the token length) is the inference LLM's pre-set context window size minus the formatted test query length. The pre-set context window size is 1,024 for the models we study in this work.

On the SST-5 and RTE datasets, *Div-S3* demonstrates roughly 3% absolute gain compared to the baseline models in terms of accuracy. Across all tasks, we see that *Div-Similar* is a strong baseline that outperforms the other two baselines: *Random-Similar* and *VoteK-Similar*. This indicates that selecting a diverse and representative set of exemplars during the annotation stage is crucial to the overall performance of ICL. Integrating *Div* (Stage 1) with *S3* (Stage 2) results in additional improvements, affirming our hypothesis that the final stage of exemplar retrieval at test time should be treated as a subset selection problem rather than independently selecting exemplars using modular similarity-based values.

When using the smaller LLMs belonging to the GPT-Neo and OPT families on four candidate datasets, we observe that *Div-S3* outperforms other baselines. Notably, across SST-2 and TREC datasets, it achieves a maximum absolute gain of approximately 10% in terms of accuracy. This demonstrates that our proposed framework *Div-S3* exhibits consistent improvements across various tasks and LLM variants.

5.6.2 Hyperparameter Tuning

Div-S3 involves a set of hyperparameters listed at the end of Sec. 5.5.1. To tune these hyperparameters, we use the validation set of each dataset. In case the validation set is unavailable, we randomly select 10% of samples from the training set while using the remaining 90% as the unlabeled pool of exemplars. Specifically, for the similarity metric needed to instantiate the facility location function, we compare the following kernels: cosine similarity with negative entries truncated to zero, modified cosine similarity adding a constant 1 to each entry, and RBF kernel with kernel widths $\sigma \in \{0.1, 0.2, 0.5, 1.0, 2.0, 5.0, 10.0\}$. While the hyperparameter search space for the kernel is quite broad, we prune the set of candidate kernels by inspecting the gains of the submodular function when performing greedy maximization, as shown in Fig. 5.4. Specifically, if a kernel configuration results in the gains diminishing

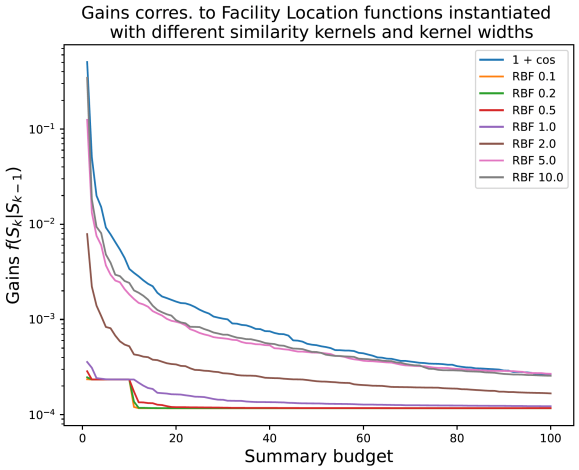


Figure 5.4: Plot displaying the Submodular Gains on the SST-5 dataset for different similarity kernel configurations. All kernels that use the RBF configuration with kernel width less than 1.0 result in gains that saturate (approach 0) prematurely. Such configurations are not useful for the exemplar annotation task, so they can be discarded by inspection.

prematurely, then samples chosen towards the later stages of the optimization procedure may be selected randomly. Pruning such kernel configurations is relatively inexpensive, since we do not need to run the end-to-end pipeline to assess their utilities on the downstream task.

Table 5.6: ICL performance on four candidate datasets using GPT-J-6B model in the non-active learning setting where only Stage 2 focusing on exemplar retrieval is active.

Method	SST-5	SST-2	RTE	TREC
Similar	50.95	89.91	54.15	**90.80**
S3 (cardinality)*	50.81	**91.28**	55.96	89.00
S3 (knapsack)*	**52.08**	-	**57.04**	90.00

The annotation budget (k) for stage 1 is set as 100 for each task and across all LLMs studied. The budget for phase 2 of S3 (k_2) when using a cardinality constraint is roughly

Table 5.7: ICL performance on four candidate datasets using two *OPT* models in the non-active learning setting where only Stage 2 focusing on exemplar retrieval is active. Our proposed methods are marked with an asterisk (*).

Method	SST-5	SST2	RTE	TREC
Similar	45.25	83.94	48.01	79.60
S3 (cardinality)*	44.34	**85.78**	**53.07**	**80.80**
S3 (knapsack)*	**45.97**	-	50.18	**80.80**

(a) OPT 2.7B

Method	SST-5	SST-2	RTE	TREC
Similar	36.06	87.50	51.99	77.60
S3 (cardinality)*	37.92	**88.30**	**53.07**	78.20
S3 (knapsack)*	**38.55**	-	51.26	**79.00**

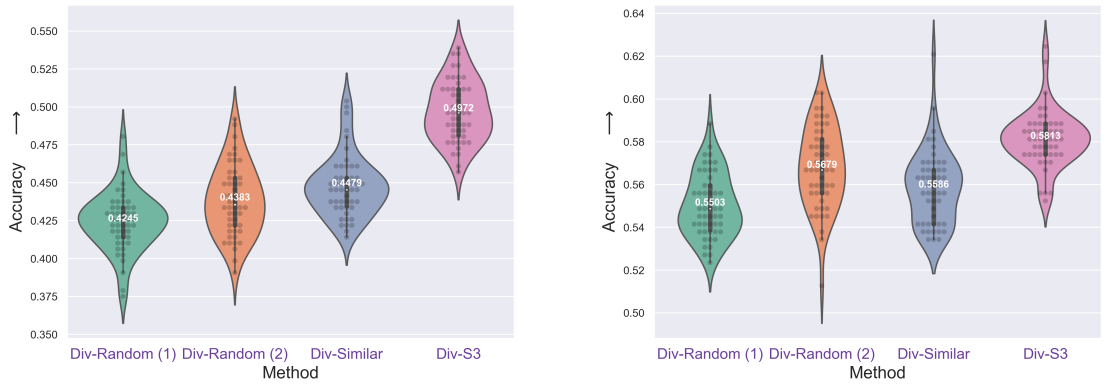
(b) OPT 1.3 B

determined based on the average number of exemplars l that are selected by *Similar* method on each task. We search for k_2 by using neighboring values of l. To tune the budget (k_1) associated with phase 1 of Stage 2, we search over $k_1 \in \{15, 20, 25, 30, 35, 40, 45, 50\}$. For the scaling factor r when optimizing Phase 2 under a knapsack constraint, we search over $r \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$. Next, we list the best-found hyperparameters for each task in the given format (similarity kernel, k_1, r): SST-5 - (1+cosine, $40, 0.1$), RTE - (1+cosine, $30, 0.1$), MRPC - (RBF kernel with width 2.0, $15, 0.1$), SST-2 - (RBF kernel with width 5.0, 30, -), TREC - (1+cosine, $30, 0.1$), DBPedia - (1+cosine, $25, 0.4$), HellaSwag - (1+cosine, $25, 0.1$)

5.6.3 Sensitivity Analysis

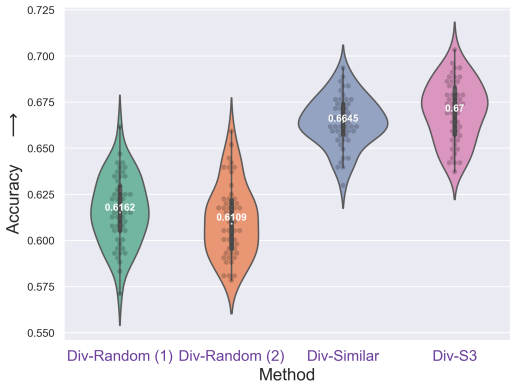
Non-Active Learning setting: In this section, we verify the effectiveness of the *S3* method for exemplar retrieval in a non-active learning setting where there are no constraints on the annotation budget. Here, we consider the entire training set as the annotated pool of exemplars, meaning $|\mathcal{D}_{\text{labeled}}| = n$. This renders the *Exemplar Annotation* stage redundant. In Table 5.6 and 5.7, when using GPT-J-6B and OPT models, respectively, we compare *S3* (when using cardinality and knapsack constraints for phase 2 of S3) to *Similar*. As depicted

in Table 5.6 and 5.7, *S3* consistently outperforms the modular selection method *Similar*, reinforcing our hypothesis of approaching exemplar retrieval as a subset selection problem.



(a) SST-5 dataset

(b) RTE dataset



(c) MRPC dataset

Figure 5.5: Sensitivity analysis of the set of exemplars retrieved by different *Exemplar Retrieval* methods to their ordering in the LLM's input prompt. The first stage of Exemplar annotation is identical in all four methods studied. When using *Random* as the second stage method, we report the sensitivity results for two different random seeds.

Order Sensitivity: Prior works such as [396] and [213] have shown that ICL's performance is extremely sensitive to the ordering of the exemplars in the input prompt, with

performances varying between random-guess levels to fine-tuning based state-of-the-art levels. In this section, our goal is to demonstrate how *Div-S3* is less sensitive to the ordering of the retrieved exemplars in the LLM's input prompt. To do this, we fix the strategy used during the exemplar annotation stage as *Div* and select exemplars from the annotated pool of exemplars using three different methods: *Random, Similar* and *S3*. Given m selected exemplars, overall m possible order permutations exist. Scoring all m orderings by making inference calls to the LLM would be computationally challenging, so we randomly sample 50 different orderings and score those across different *Stage 2* methods. In Figure 5.5, we demonstrate the performance variation corresponding to these random orderings on three candidate datasets, and it can be seen that *Div-S3* achieves better average accuracy while being less sensitive to the order of the exemplars compared to other baselines due to its ability to balance relevance and diversity during the exemplar retrieval stage.

5.7 Summary

In this chapter, we propose *Div-S3*, which unites the two stages of exemplar selection using submodular optimization. For the Exemplar Annotation stage, we select a diverse subset of exemplars for annotation under a fixed budget utilizing submodular maximization under a cardinality constraint. For the Exemplar Retrieval stage, we utilize a Submodular Span Summarization approach that finds diverse annotated exemplars most relevant to the test instance. Compared to previous methods on the exemplar selection task, *Div-S3* models both diversity and relevance for the second stage and incorporates the rich class of submodular functions. On multiple NLP tasks using various LLMs, *Div-S3* shows consistent improvements and is also more robust to the ordering of the exemplars empirically as we account for diversity in the exemplar retrieval stage.

Chapter 6

SUBMODULARITY FOR KV CACHE SUMMARIZATION IN LLMS

6.1 Introduction

The multi-headed self-attention [328] serves as the building block of several state-of-the-art Transformers-based models for various tasks such as language understanding and generation [262], image recognition [81], and recommendations [350] using Large Language Models (LLMs). It enables models to learn long-range dependencies and complex patterns by focusing on multiple sequence sections simultaneously, regardless of their proximity. However, deploying LLMs is challenging for at least two main reasons: (1) Quadratic scaling of attention: the attention mechanism scales quadratically with sequence length, leading to high computational costs and memory requirements for processing longer sequences; (2) Autoregressive decoding wherein the LLM generates the output sequence token by token conditioned on the previously generated results and the input sequence, which requires accessing or recomputing the key and value representation of all previous tokens.

Existing approaches to tackle the first problem include input sequence truncation [72, 175], sliding window-based sequence chunking [177, 103], memory augmentation methods [66, 261, 222, 381], and retrieval-based methods [137, 351, 32, 400]. However, conventional techniques like chunking suffer from well-known problems, such as context fragmentation [66]. Similarly, retrieval-based non-parametric methods to augment an input query with relevant context require constructing an external memory bank that is several orders of magnitude larger than the dataset size. In addition, a retriever index must be constructed on top of the memory bank.

As discussed in Sec. 3.1.3, KV caching mitigates recomputation costs in autoregressive

decoding by storing previously computed key and value vectors [253]. However, the KV cache size grows linearly with sequence length, resulting in GPU memory saturation during long-sequence processing [197, 308, 113]. Additionally, as model size increases, the memory footprint of a single token increases as well. While it is possible to offload the KV cache to the host DRAM, this will incur a host-to-device latency for each inference call. Given the growth in model size and context lengths of publicly available models [269, 12], reducing the memory overhead of the KV cache while retaining model accuracy becomes even more important.

Existing system-level techniques such as FlexGen [298], Paged Attention [163], and Flash Attention [67] improve the utilization of GPU resources and throughput when dealing with the attention mechanism and KV caches. However, they do not consider the impact of ever-increasing KV cache sizes. Modeling techniques such as multi-query [297] and group-query attention [3] help mitigate the size of the KV cache by removing unnecessary heads, but they require expensive retraining or fine-tuning.

Given the wide deployment of accelerators and models already in the field, we need to leverage inference time techniques that reduce the size of the available context at all layers (KV cache) without impacting accuracy. Existing related works, such as Heavy Hitters [393], Scissorhands [207], KeyFormer [1], and FastGen [100], rely on heuristic-based techniques to preserve only the important key-value attention states while evicting the non-important attention states. They employ **modular scores** (described in Sec. 6.4) such as attention scores attributed to different tokens accumulated over different time steps. This does not take into account the importance of keeping a particular KV attention state in the context of other states already present in the KV cache and thus does not capture long-term dependencies between tokens. We hypothesize that KV cache selection should be framed as a subset selection problem where we evaluate the utility of different key-value pairs as a set instead of independently evaluating each key-value attention state.

In this chapter, we present a novel yet simple approach called **BumbleBee**[1] for KV

[1]Named after our favorite Transformer who is known for being "efficient", highly "acute", and "adaptable".

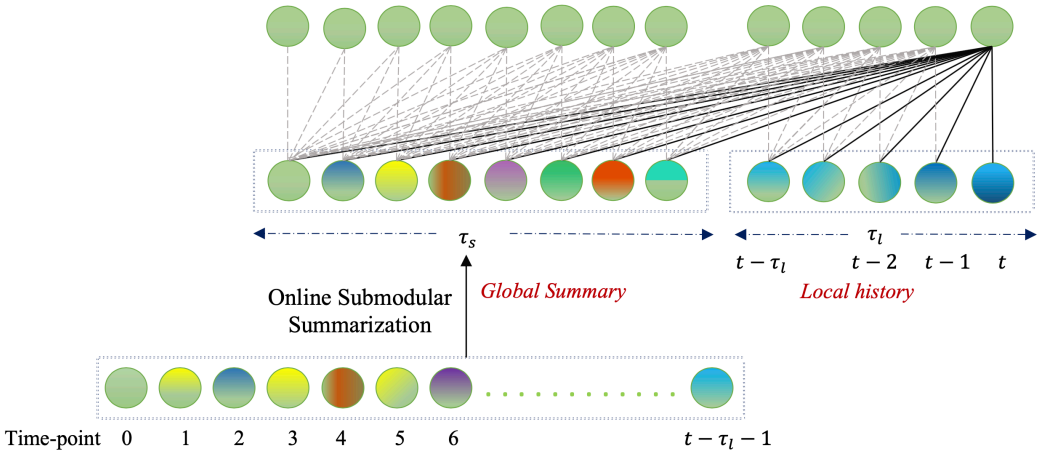


Figure 6.1: Illustration of the attention mechanism in a self-attention head of one of the *BumbleBee*'s decoder layers. τ_l and τ_s denote the local context length and the limited global summary length, respectively.

cache summarization, thus enabling existing LLMs to be used for longer contexts without any additional fine-tuning, CPU offloading [298], or context truncation [24] in case of longer sequences (Fig. 6.1). The temporal span of the context (which is defined as the difference between the time of the latest and the time of the earliest context token) may be unboundedly long in a *BumbleBee* model. This is true even though the total number of tokens in a *BumbleBee* context does not grow unboundedly. Indeed, *BumbleBee* draws inspiration from the following aspects of human psychology:

1. Selective attention [56, 324] allows us to focus on relevant information and filter out distractions or irrelevant details. This plays a critical part in determining what subset of information should get successfully encoded into our memory. Similarly, in *BumbleBee*, we maintain a memory of diverse, representative, and important time points of the sequence observed so far by utilizing online submodular summarization.

2. Humans process information in an online and dynamic fashion over an extended period,

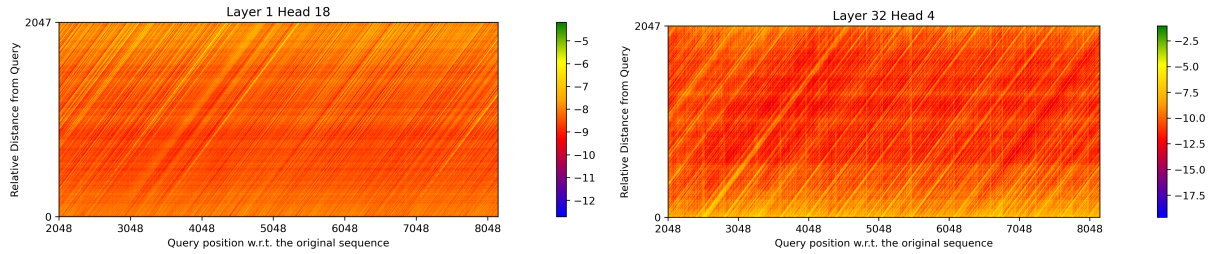
possibly all the way to very early in life. This involves relying on our (possibly extremely long) past memory to make sense of incoming information through associations, selectively attending to relevant information, and organizing our existing memory in light of new information. This top-down processing style [99] enables us to prioritize and encode pertinent information for future recall. Similarly, in the case of *BumbleBee*, we process an incoming segment or chunk by contextualizing it with the online summary (or memory) obtained thus far and update the summary (or memory) based on the segment's informativeness and its association with the current memory. Like human memory, *BumbleBee*'s online summary may include tokens anywhere from the earliest point in the stream all the way to the present.

To realize the above inspiration, *BumbleBee* utilizes online submodular summarization to maintain a diverse subset of important key-value attention states on the fly that are representative of long-term global history. Incorporating a fixed-size summary of the entire historical sequence allows the *BumbleBee* to capture long-range dependencies (with unbounded temporal extent) without increasing the memory consumption or the length of input fed to the Transformer. We further combine the global submodular summary with the most recent local context as shown in Fig. 6.1, so *BumbleBee*'s predictions are dependent on both the latest context as well as the global wide-ranging patterns observed so far. Thus, our approach boosts the Transformer's capabilities to capture both the long-term and short-term contextual dependencies while maintaining similar (and thus feasible) memory and computational costs for a fixed-size input sequence length.

This chapter is based on the work published in: **Lilly Kumari**, Shengjie Wang, Tianyi Zhou, Nikhil Sarda, Anthony Rowe, and Jeff Bilmes. *BumbleBee: Dynamic KV-Cache Streaming Submodular Summarization for Infinite-Context Transformers. In First Conference on Language Modeling. COLM, 2024.*

6.1.1 Empirical Evidence

LLMs are pre-trained using a fixed context window, and prior work has shown their limited generalization on sequences significantly longer than the pre-trained context window [135]. Recently, some of the closed sourced models that have come out have a context window longer than 128k tokens; for example, GPT-4 [245] has a context window of 128k, Claude 2.1 [11] can process 200k tokens while Gemini 1.5 pro [269] can handle a 1M input context. To utilize such a long context, caching of key-value states is utilized to keep the latency associated with the inference phase low. However, the KV cache grows linearly with the sequence length. For example, for LLaMA-13B model [321], the KV cache for storing 128k tokens would roughly take 40 (number of decoder layers) $\times$ 5120 (hidden size) $\times$ 2 (kv pair) $\times$ 2 (fp16) $\times$ 128000 = 105 GB.



(a) Test sample 1 (b) Test sample 2

Figure 6.2: Attention maps for two different WikiText-103 articles using LLaMA-7B model.

To address the memory costs associated with the KV cache, we focus on the following question:

Is it possible to maintain the LLM's performance on downstream tasks without storing every observed token representation in the KV cache?

We conduct experiments on the WikiText-103 [225] dataset to support our hypothesis that self-attention is selective and keeping only a set of **both** diverse and important tokens is sufficient to maintain performance. Specifically, we use the LLaMA-7B model (with a pre-trained context window size of 2048) for a next-token prediction task on randomly sampled

articles from wikitext-103 and visualize the attention scores attributed to different tokens present in the context window of size 2048 in Fig. 6.2. On the x-axis, each token represents the query used for next-token prediction, while each vertical slice (column) illustrates the logarithm of normalized attention scores for the 2048 in-context tokens (or keys). As we move from the bottom row to the topmost row along the y-axis, the relative distance of the in-context keys from the query token increases from 0 to 2047. The anti-diagonal pattern shown in Fig. 6.2 shows that there is a small subset of tokens that are strongly attended to while they are present in the context window.

However, the constraint of the context window restricts tokens in longer sequences to only attend to nearby local tokens, limiting their ability to capture broader contextual information. In this chapter, we show that by using a submodular function to summarize the entire historical contextual information in the KV cache across different attention heads, one can reduce the memory footprint associated with the KV cache while maintaining acceptable performance compared to using the entire cache.

6.2 Notations & Background

In this section, we provide a concise background about self-attention used in Transformer-based LLMs. We also briefly discuss the KV cache mechanism used to avoid re-computations during the decoding stage, as well as the submodular functions used to model different key attributes.

Notations We denote a sequence of tokens as $x = \{x_1, x_2, \ldots x_n\}$ where $x_t \in \mathbf{R}^d$ is the t-th token embedding. In a decoder-only Transformer model, we denote the number of decoder layers present as n_l and the number of attention heads present in each layer as n_h. The self-attention mechanism of head h in layer $(l+1)$, utilizing distinct weight matrices for query (W_q), key (W_k), and value (W_v), operates on the hidden states from the preceding layer, represented as H^l. Specifically, we get the following query, key, and value embeddings after the linear projections: $Q^{l+1} = H^l W_q$, $K^{l+1} = H^l W_k$, and $V^{l+1} = H^l W_v$. Row t in the key

embedding matrix denotes the key vector corresponding to the t-th token, and this holds for both query and value matrices as well. Given query q_t, its attention output o_t using scaled dot product attention is computed as shown in Eq. (6.1). Here, we use S_t to denote the KV cache accumulated at step t. From here onwards, we omit notation pertaining to the specific layer and head for the sake of simplicity.

$$\mathrm{o}_t = \sum_{(k_j, v_j) \in S_t} \frac{\exp\left(k_j^T q_t\right)}{\sum_{k_i \in S_t} \exp\left(k_i^T q_t\right)} v_j = \sum_{(k_j, v_j) \in S_t} a(q_t, k_j, S_t)\, v_j \tag{6.1}$$

KV Cache Decoder-only transformers operate in an autoregressive manner, predicting tokens sequentially based on previously generated (and observed) ones. By caching previously computed representations, specifically the embeddings for keys and values corresponding to observed tokens, the model can avoid redundant computations in each decoding step. Specifically, the KV cache initially computes the attention states for an input prompt, represented by $S_0 = \{(k_i, v_i) | i \leq n\}$, and caches them in memory. For every subsequent step $j \leq k$, the model reuses the cached states $S_j = \{(k_i, v_i) | i < n + j\}$ to compute the attention state (k_{n+j}, v_{n+j}) of the new token s_{n+j}. This significantly reduces the floating-point computations used for matrix operations to compute the new attention states. After each step, the newly computed attention states are appended to the cache for subsequent use, such that $S_j = S_{j-1} \cup \{(k_{n+j}, v_{n+j})\}$.

Note that the KV cache is an approximation and trades off precision for speed. The attention state computation for token s_j is limited to the sequence available at step j, namely $\{s_i | i < n + j\}$ as opposed to over the entire sequence $\{s_i | i < n + k\}$. KV caches significantly reduce latency and result in a minimal loss in accuracy, making them an essential ingredient in practical LLM deployment.

Since the KV cache is updated with a new token at each decoding step, its size grows linearly with the overall sequence length (including both input prompt and generated tokens) and the batch size. This becomes a major bottleneck when dealing with longer sequences whose KV cache can not fit in the GPU's high bandwidth memory (HBM). Therefore, in

this chapter, we study how to summarize the KV cache by keeping only a small subset of important and diverse and thus representative key-value embeddings while discarding the rest.

Submodularity To model diversity and representativeness, we utilize the facility location function described in Sec. 2.1.2. The FL function utilizes similarity scores $\text{sim}(v, v')$ computed over every pair $v, v' \in V$ of items. A valuation is then the sum of similarities from any item in the ground set V to its closest representative in the given set A as shown in Eq. (6.2).

$$f(A) = \sum_{v' \in V} \max_{v \in A} \text{sim}(v, v'). \tag{6.2}$$

$$c(A) = \sum_{u \in U} \phi_u(\sum_{v \in A} m_u(v)). \tag{6.3}$$

Another widely used submodular function is the feature-based function described in Sec. 2.1.2, and it has the form shown in Eq. (6.3). Here, we have $\phi_u(\cdot)$ as a monotone non-decreasing non-negative concave function, and $m_u(\cdot)$ is a non-negative weight associated with the u-th feature of every item $v \in A$. Due to the diminishing property of $\phi_u(\cdot)$, to have a large function valuation for a set A, we would require the sum of every feature across items to be uniformly large, thus inducing diversity and fairness over a feature representation of the selected subset.

6.3 Related Work

Attention speedup Self-attention is a critical component of the Transformer [328] mechanism powering modern language models but suffers from quadratic complexity. Techniques such as Linformer [338], Performer [53], Linear Transformers [134], and Reformer [145] aim to reduce the time complexity of self-attention via low-rank approximations or hashing techniques. Longformer [24] introduces sliding-window attention to reduce the computational overhead of self-attention. FlashAttention [67] implements an IO-aware kernel to compute

self-attention and uses tiling to reduce memory overhead. Keyformer [1] speeds up attention computation by exploiting the empirical observation that 90% of attention weights focus on a small subset of tokens. For workloads that share similar inputs, prompt caching [102] and prefix sharing [367] have emerged to reduce the computation and memory overhead of self-attention.

KV cache compression and management We discuss existing KV cache compression methods in Sec. 3.1.3.

Applications of submodularity to LLMs INGENIOUS [273] is a technique that uses submodular optimization for selecting representative subsets of the training data such that the language models trained thereof achieve comparable performance to models trained on the full dataset. SMART [272] proposes a data mixture strategy for instruction tuning, leveraging a submodular function for importance score assignment to tasks that are used to determine the mixture weights. Submodular functions have also been used to augment LLMs for multi-document summarization [162]. To the best of our knowledge, ours is the first work to apply submodularity for KV cache summarization.

6.4 Set Function Optimization View of Prior Work

Heavy Hitter Oracle (H_2O) [393] is a training-free KV cache reduction strategy designed to reduce memory usage during autoregressive decoding in decoder-only Transformer models. Instead of retaining all past tokens in the cache, H_2O selects a subset based on their *accumulated attention scores*, aiming to preserve only the heavily attended tokens—referred to as *Heavy Hitters*. The method builds on the observation that only a small fraction of past tokens contribute significantly to future decoding steps, making it possible to safely evict others without degrading model performance.

Let $K_n = \{k_i \mid i = 1, 2, \ldots, n\}$ and $V_n = \{v_i \mid i = 1, 2, \ldots, n\}$ denote the keys and values corresponding to the first n tokens in the KV cache. Using the attention formulation in

Eq. (6.1), H_2O computes an *accumulated attention scores vector* $a_n \in \mathbb{R}^n$, where each element a_n^i represents the total attention attributed to key k_i across all subsequent query tokens. This accumulation naturally arises due to the causal structure of decoder self-attention. The accumulated attention score for key k_i is computed by summing the attention it receives from all queries at time steps $t \geq i$, as follows:

$$a_n^i = \sum_{t=i}^{n} a(q_t, k_i, S_t). \tag{6.4}$$

Using the accumulated scores a_n, H_2O selects a subset of indices $\mathcal{I}_{\text{selected}} \subseteq \{1, \ldots, n\}$, subject to a KV cache budget constraint $|\mathcal{I}_{\text{selected}}| \leq k$, such that the total accumulated attention is maximized:

$$\mathcal{I}_{\text{selected}} \in \underset{\mathcal{I} \subseteq \{1,\ldots,n\},\ |\mathcal{I}| \leq k}{\operatorname{argmax}} \sum_{i \in \mathcal{I}} a_n^i. \tag{6.5}$$

This defines a *modular utility function* $f(\mathcal{I}) = \sum_{i \in \mathcal{I}} a_n^i$, where each token's contribution is additive and independent of others. Consequently, token (key-value) selection simply reduces to a sorting operation over the a_n vector. This does not account for *redundancy or diversity* in the selected subset of key-value tokens. Moreover, applying concave transformations to a_n^i does not affect the selection order, hence resulting in the same top-k tokens. Although the original paper references submodular optimization, it does not leverage submodularity or employ an iterative greedy algorithm (Algorithm 1) in its implementation at https://github.com/FMInference/H2O.

FastGen [100] is another training-free KV cache compression method aimed at reducing memory consumption during autoregressive decoding in decoder-only LLMs. The method leverages the observation that attention patterns in different attention heads of the model exhibit consistent behaviors. FastGen introduces an offline *attention profiling* stage that analyzes the behavior of each attention head and assigns it one of several KV cache reduction policies: local context, special tokens, punctuation tokens, heavy hitters, or full cache retention. Each policy prescribes a fixed, rule-based strategy for deciding which tokens' key-value (KV) pairs should be retained in the cache during inference. This head-specific policy is

applied independently to each attention head.

Instead of retaining all keys K_n and values V_n, FastGen selects a subset of token indices $\mathcal{I}_{\text{selected}} \subseteq \{1, \ldots, n\}$ based on the attention profile type. For local context patterns, only the most recent τ_l tokens are retained, i.e., $\mathcal{I}_{\text{selected}} = \{n - \tau_l + 1, \ldots, n\}$. For special token patterns, $\mathcal{I}_{\text{selected}} = \{i \mid x_i \in \mathcal{T}_{\text{special}}\}$, where $\mathcal{T}_{\text{special}}$ denotes a pre-defined set of tokens such as [INST], $\langle s \rangle$, etc. For punctuation token patterns, only the key-value pairs corresponding to punctuation tokens are retained. For heavy hitters based on frequently attended keys, the H_2O algorithm is used to identify the heavy hitters, and only those tokens are retained in the KV cache for that attention head. This token selection is applied independently per attention unit, but the selection criteria are fixed and non-adaptive once the profile is assigned.

Although FastGen reduces cache size by exploiting structural regularities in attention behavior, it does so using simple heuristics rather than through any principled optimization framework. Moreover, there is no explicitly defined utility function $f(\mathcal{I}_{\text{selected}})$, except in the case of heads assigned a heavy hitter compression policy, where accumulated attention scores guide token selection similar to H_2O. While the utility of retained tokens likely exhibits diminishing returns, the method does not formalize or leverage this property. As a result, FastGen may retain redundant tokens, limiting its potential to achieve more efficient cache reduction.

L2-norm-based selection [73] is another training-free KV cache selection method. Instead of selecting tokens based on attention scores, this method leverages the observation that key embeddings with low L2 norms tend to attract higher attention weights across different attention heads. This behavior remains consistent across different decoder layers, suggesting that the L2 norm of keys can be used as a static proxy for their future importance in attention computation. Given the set of keys K_n for n tokens, this method defines a utility score $u_i = \frac{1}{\|k_i\|_2}$ for each key independently, assigning higher utility to keys with lower norms. It then selects a subset of token indices $\mathcal{I}_{\text{selected}} \subseteq \{1, \ldots, n\}$ under a size constraint $|\mathcal{I}_{\text{selected}}| \leq k$, maximizing the total utility over the selected set.

Formally, the selection objective is:

$$\mathcal{I}_{\text{selected}} \in \underset{\mathcal{I}\subseteq\{1,\ldots,n\},\ |\mathcal{I}|\leq k}{\operatorname{argmax}} \sum_{i\in\mathcal{I}} u_i, \tag{6.6}$$

which defines a *modular set function* $f(\mathcal{I}) = \sum_{i\in\mathcal{I}} u_i = \sum_{i\in\mathcal{I}} \frac{1}{\|k_i\|_2}$, where each token's contribution is additive and independent of others. Similar to the H_2O work described earlier, this selection reduces to a simple top-k sorting operation over the utility scores. However, because each token is considered in isolation, this method does not control for *redundancy* in the selected key-value pairs, and maintaining low-norm keys with highly similar semantic content can lead to wasted memory capacity.

6.5 BumbleBee

In Sec. 6.1.1, we saw that some keys, despite their distance from the query, are heavily attended to, showing that it is important to preserve these keys in the overall context, even if they are quite distant. While the importance of keys captured by the attention scores is one aspect that should be considered for context (or KV cache) selection, the diversity and representativeness of the selected keys are equally critical. That is, we require the context summary (which has a fixed size) to be both relevant to and representative of the entire context. This is how efficiency is achieved—amongst sets of a given fixed size, a diverse set means a non-redundant set, while a redundant set means that certain concepts are inefficiently over-represented while some concepts are poorly represented. To capture these two properties (diversity and importance), we thus design our final scoring function g_λ, which is a mixture of submodular functions described in detail in Sec. 2.1.2. g_λ is defined as a convex combination of the facility location function (Eq. 6.2) and the feature-based function (Eq. 6.3).

$$g_\lambda(A) = \lambda f(A) + (1-\lambda)c(A) \tag{6.7}$$

Here, $\lambda \geq 0$ is a hyperparameter that controls the trade-off between representativeness and relevance. Both component functions are monotone, non-negative, submodular, and assumed

to be normalized, i.e., $f(\emptyset) = 0$ and $f(V) = 1$ (and the same for $c(\cdot)$). This normalization ensures the compatibility among the mixture components, and the resulting mixture function g_λ inherits these properties.

Algorithm 4 Offline Submodular KV Cache Summarization during Prefill/Encoding Phase

1: **Input:** Submodular functions f and c capturing diversity in the key embeddings space and importance in terms of attention frequency respectively for layer l and attention head h, mixture function $g_\lambda(\cdot) = \lambda f(\cdot) + (1-\lambda)c(\cdot)$, a set of n KV attention states $K_n = \{(k_i)\}_{i=1}^n, V_n = \{(v_i)\}_{i=1}^n$ corresponding to the n prompt tokens, budget τ_s

2: **Output:** A final summary S_n such that $S_n \subseteq \{(k_i, v_i)\}_{i=1}^n$ and $|S_n| \leq \tau_s$.

3: **Initialize:** $S_n = \emptyset$, Compute accumulated attention scores vector a_n for each key $k \in \{k_i\}_{i=1}^n$. a_n^i denotes the accumulated attention scores attributed to the key k_i across all n query tokens.

4: **for** $j = 1$ to τ_s **do**

5: $\quad k_{\text{imp}} \leftarrow \text{argmax}_{e \in K_n \setminus S_n} g_\lambda(S_n \cup e) - g(S_n)$

6: $\quad S_n \leftarrow S_n \cup \{(k_{\text{imp}}, v_{\text{imp}})\}$ where v_{imp} is the value embedding associated with k_{imp}.

7: **end for**

6.5.1 Offline BumbleBee Algorithm

In Algorithm 4, we present the offline KV cache summarization pseudo-code to compute a summary of the KV cache under a cardinality constraint τ_s. This algorithm is used for the KV cache summarization in each self-attention head present in different decoder layers. In Line 3, we first compute the attention scores for all keys accumulated over different time steps and then use that to instantiate our feature-based function $c(\cdot)$. In our current setting, we only experiment with one feature function, meaning $|U| = 1$ in Eq. (6.3). Using the key embeddings, we instantiate the facility location function $f(\cdot)$ on a similarity matrix computed using pairwise cosine similarities followed by the ReLU transformation.

Algorithm 5 *BumbleBee*: Streaming Submodular KV cache Summarization for Transformers

Input: Submodular functions f and c capturing diversity in the key embeddings space and importance w.r.t. attention frequency resp. for layer l and attention head h, mixture function $g_\lambda(\cdot) = \lambda f(\cdot) + (1-\lambda)c(\cdot)$, stream of QKV attention states $\{(q_i, k_i, v_i)\}_{i=1}^{n}$, budget τ_s

Output: A running summary S_t of for every time step t such that $S_t \subseteq \{(k_i, v_i)\}_{i=1}^{t}$.

Initialize: $S_0 = \emptyset$, $a_0 = \emptyset$ where $a_t \in \mathbf{R}^{|S_t|}$ denotes the accumulated attention scores corresponding to keys present in S_t across t timesteps.

for $t = 1, \ldots, n$ **do**

  Update a_t for each $k \in S_{t-1}$ by adding $a(q_t, k, S_{t-1} \cup k_t)$

  if $t < \tau_s$ **then**

    $S_t \leftarrow S_{t-1} \cup \{(k_t, v_t)\}$

    Append $a(q_t, k_t, S_t)$ to a_t s.t. $|a_t| = |S_t|$

  else

    Let $S_t^{\text{tmp}} = S_{t-1} \cup \{(k_t, v_t)\}$

    $k_{\text{discard}} \leftarrow \text{argmin}_{k_i \in S_t^{\text{tmp}}} g_\lambda(k_i | S_t^{\text{tmp}} \setminus k_i)$

    $S_t \leftarrow S_t^{\text{tmp}} \setminus \{(k_{\text{discard}}, v_{\text{discard}})\}$

    if $k_{\text{discard}} \neq k_t$ **then**

      Evict a_t^j from a_t where a_t^j denotes the accumulated attention score for the discarded key k_{discard}.

      Append $a(q_t, k_t, S_t)$ to a_t

    end if

  end if

end for

In Lines 4-7, we use the greedy algorithm [239, 231, 234] outlined in Algorithm 1 in Sec. 2.2.1 to perform a cardinality-constrained submodular maximization. Thanks to submodularity, the resultant set is within a factor of $(1 - 1/e)$ from the optimal summary [239]. This offline routine is suitable for KV cache summarization after the prefill stage of LLMs, particularly in serving systems where prompts are shared across user requests and their KV embeddings are pre-computed and cached [163, 102].

6.5.2 Streaming BumbleBee Algorithm

In Algorithm 5, we outline the KV cache summarization algorithm suitable for a streaming setting where we do not have prior knowledge of the complete sequence, restricting us from using existing offline summarization algorithms to obtain a global summary of the sequence. In light of this limitation, we propose to summarize the sequential data observed so far and produce an online summary of a fixed size that serves as its representative set. We use the same mixture function as Algorithm 4 as our final scoring function.

In Lines 6-8, we keep caching the (key, value) pairs for different attention heads unconditionally until the summarization budget is exhausted. Once the KV cache is full, we utilize the key embeddings in S_t and the accumulated attention score vector a_t to update the submodular function components, $f(\cdot)$ and $c(\cdot)$ of the final mixture function $g_\lambda(\cdot)$. In Line 10, we create set S_t^{tmp} that includes the newest incoming (k_t, v_t) pair and evaluate the conditional gain of keeping an item around in the context of the remaining summary set as shown in Line 11. The element with the least conditional gain is removed from S_t^{tmp} and the accumulated attention vector a_t is modified in Lines 13-15 to include the attention score associated with the newest key when the discarded key is one of the keys in the previous summary set S_{t-1}.

The streaming summarization algorithm is suitable for memory-constrained settings where the KV cache for the entire context can not be maintained in the GPU memory. Also, in multi-turn dialogue systems [82, 218], after a certain number of interactions, it can be challenging to keep track of the entire conversation context. In such scenarios, *BumbleBee*

can enable modern LLM-based serving systems to maintain a representative yet important summary of past interactions while minimizing the KV cache-based memory utilization.

6.5.3 Complexity Analysis

Compute costs In BumbleBee (Algorithm 5), we maintain a global summary of size τ_s. So, constructing the pairwise similarity matrix for f (FL function) has a time complexity of $\mathcal{O}(\tau_s^2 \times d)$. Identifying the item with the least conditional gain (Line 11 of Algorithm 5) requires $\mathcal{O}(\tau_s^2)$. However, if we cache the similarity matrix, we only need to compute the similarity of the incoming item to the others in the summary, resulting in an overall complexity of $\mathcal{O}(\tau_s \times d + \tau_s^2)$ for each new token.

Memory Costs Caching the similarity matrix incurs $\mathcal{O}(\tau_s^2)$ memory costs.

6.6 Experiments

6.6.1 Synthetic Data Experiments

We are given a dataset having inputs defined as $\boldsymbol{x} = \{x_1, x_2, \ldots, x_n\}$ where $x_t \in \mathbf{R}^{d_{\text{in}}}$ denotes the t-th input token and d_{in} is the input feature dimensionality. Similarly, we have the outputs defined as $\boldsymbol{y} = \{y_1, y_2, \ldots, y_n\}$ where $y_t \in \mathbf{R}^{d_{\text{out}}}$ and d_{out} is the dimension of the output embedding. We hypothesize that *BumbleBee* can achieve significant improvements in the task involving estimating $y_t \in \mathbf{y}$ over existing methods across different dimensions if y_t is a function of both of the following:

(a) The local context window, $local_t$, a sequence $\{x_{t-h}, x_{t-(h-1)}, \ldots, x_{t-1}\} \subseteq \boldsymbol{x}$ where h is the local context window length.

(b) The summary sequence, $summ_t$, comprising k relevant vectors chosen from $\{x_1, x_2, \ldots, x_{t-h-1}\} \subseteq \boldsymbol{x}$. Specifically, let $S_t \subseteq \{1, 2, \ldots, (t-h-1)\}$ and say $S_t = (s_1, s_2, \ldots, s_k)$ is ordered by time (or position) in the original sequence so that $s_1 < s_2 < \cdots < s_k$. Then we have that $summ_t = (x_{s_1}, x_{s_2}, \ldots, x_{s_k})$.

Also, assume in this discussion that $t \geq h + k$ to avoid any negative indices.

To understand the context where we expect *BumbleBee* to function well, we define an (undesirable) conditional independence property as follows:

$$y_t \perp\!\!\!\perp summ_t | local_t \tag{6.8}$$

This property states that the predicted variable y_t is independent of the summary given the local context—this means that the local context is sufficient to predict y_t, and once we have this local context, the summary is not needed. This property is precisely what we must **not** have in the data for *BumbleBee* to function properly. That is, this property should not hold for any of the estimation tasks under consideration.

In terms of conditional mutual information, the property that we would like to have is that $I(y_t; summ_t | local_t) > 0$, meaning that even if we have the local context, the summary is still informative of y_t. This property being true means *BumbleBee* should be effective.

To demonstrate the effectiveness of the *BumbleBee* on such tasks, we define an ideal experimental setup as follows:

We obtain $\boldsymbol{x} \sim \mathcal{N}(0, 1)$, for each $t \in \{(k + h), (k + h + 1), \ldots, N\}$, we sample $local_t \in \boldsymbol{x}$ using a sliding window mechanism and $summ_t \in \boldsymbol{x}$ uniformly at random based on the window size and summary budget size defined previously. We define a generative process to obtain $\boldsymbol{y}$ as follows:

$$\boldsymbol{y} = f_{\text{ground-truth}}(summ, local)$$

$f_{\text{ground-truth}}$ is chosen in such a way that the summary and local context tokens are both equally important to $\boldsymbol{y}$. This is done by obtaining,

$$\boldsymbol{y}^{local} = f_{\text{ground-truth}}(local)$$

$$\boldsymbol{y}^{summ} = f_{\text{ground-truth}}(summ)$$

where the above model $f_{\text{ground-truth}}$ is achieved via standard transformer masking/padding (setting the corresponding attention values in the matrix to zero), and then comparing their

Mean Squared Error (MSE) scores with respect to $\boldsymbol{y}$ to be comparable as shown in Table 6.1. Note that we are not training the ground-truth model but rather initializing it with a seed for which the predictions made using (a) only the local tokens and (b) only using the summary tokens receive similar MSE values. In terms of the architecture, $f_{\text{ground-truth}}$ is a single-layer encoder-decoder transformer model with a linear projection layer applied to the outputs of the decoder. Here, $h = 32$, $k = 32$, $d_{in} = 16$, $d_{out} = 1$, $n = 80{,}000$.

Table 6.1: Performance comparison of the generated ground-truth model on the entire dataset when only the summaries are used and when only the local context is used

Model	**MSE (Mean Squared Error)**
$f_{\text{ground-truth}}(summ)$	8.2689×10^{-4}
$f_{\text{ground-truth}}(local)$	8.3301×10^{-4}

Table 6.2: Model performance on held-out test set

Model	**MSE (Mean Squared Error)**	**MAE (Mean Absolute Error)**
$f_{learnt}^{\text{summ,local}}$	9.14×10^{-7}	7.7279×10^{-4}
f_{learnt}^{local}	4.21×10^{-5}	5.09×10^{-3}
f_{learnt}^{summ}	4.83×10^{-4}	1.685×10^{-2}

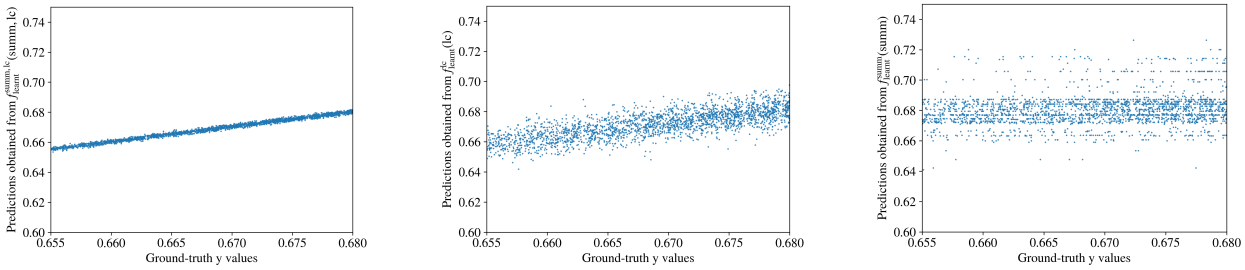
To demonstrate our initial hypothesis, we now perform training and obtain three different learned models, as follows:

a. $f_{learnt}^{summ,local}$ - trained using both summaries and local context tokens.

b. f_{learnt}^{local} - trained using only the local context tokens.

c. f_{learnt}^{summ} - trained using only the summary tokens.

For the entire training setting, unless mentioned otherwise, we perform an 80:10:10 train, validation, and test split, respectively (random seed initialization same across all three settings), and train for 50 epochs. The results are evaluated on the held-out test set. From Table 6.2 and Fig. 6.3, it is clear that $f_{learnt}^{summ,local}$ performs significantly better than f_{learnt}^{local} and f_{learnt}^{summ}, exhibiting the importance of both summary and local context input tokens for the final predictions. In Fig. 6.4, we provide qualitative results showing the predicted values using different learned models.



(a) (b) (c)

Figure 6.3: Performance comparison on a subset of the held-out set for different models trained using (a) summaries and local context; (b) only local context; (c) only summaries. Note that the plot in (c) looks granular in the vertical dimension since the summaries are not changing at every time point, i.e., time is segmented into regions where the summary is fixed, meaning $S_t = S_{t-1}$ for certain regions.

We observe the stripe pattern in Fig. 6.3c because the input summary tokens remain constant across multiple episodic periods due to the design of the data sampling process.

We further demonstrate that using a large local context does not necessarily yield comparable results using the same training setting. To achieve this, we simply increase the size of the local context window by 10 times the original to obtain, $f_{learnt}^{local\times 10}$. We compare the performance of these models on metrics like MAC (Multiply and Accumulate), runtime memory, and MSE. The MAC and runtime memory metrics are computed for a single training

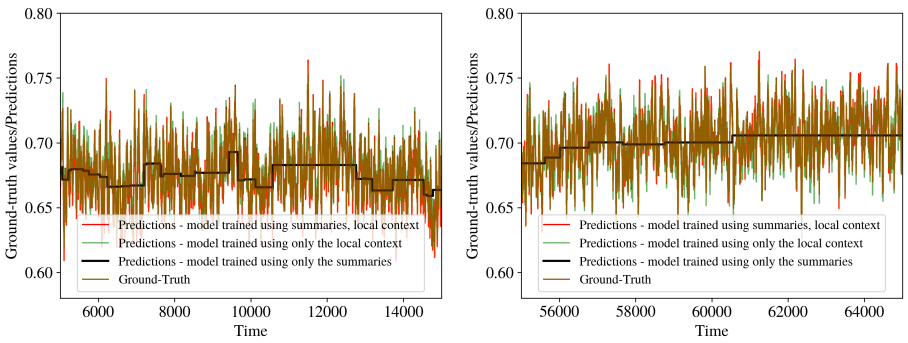


Figure 6.4: Plot for error comparison with respect to ground-truth data of different trained models for two different time windows

example in a mini-batch.

Table 6.3 clearly shows the effectiveness of having a mechanism to summarize the historical context over naively fitting longer inputs in the transformer's context in terms of metrics such as runtime memory used, MSE, etc.

Table 6.3: Performance comparison of a model trained using a naive large local context to a *BumbleBee* model

Model	**MSE (Mean Squared Error)**	**MAC**	**Runtime Memory**
$f_{learnt}^{\text{summ,local}}$	9.14×10^{-7}	6.33×10^{5}	3.43 MB
$f_{learnt}^{local \times 10}$	3.43×10^{-4}	3.12×10^{6}	5.98 MB

6.6.2 NLP Experiments

Datasets

The datasets studied in this work are derived from three benchmarks: *lm-eval-harness* [96], *HELM* [191], and *LongBench* [19]. Following Heavy-Hitters [393], we select the following six few-shot datasets from lm-eval-harness: OpenbookQA [228], COPA [277], RTE [334], MathQA [9], PiQA [28], and Winogrande [283]. From the HELM benchmark, we choose the single document summarization dataset XSUM [238].

From the LongBench benchmark [19], which is specifically focused on evaluating the long context understanding of LLMs, we select four tasks and their associated datasets: (1) Single document question answering: Qasper [69] and MultiFieldQA, (2) Multi-document question answering: HotpotQA [364] and 2WikiMultihopQA [117], (3) Summarization: QMSum [399], and (4) Few-shot learning: TREC [185].

Evaluated Models

We use LLaMA 7B and 13B models [320] for tasks belonging to the lm-eval-harness benchmark. For the XSUM dataset, we use LLaMA2 7B and 13B models [321]. On the LongBench selected datasets, we use the Llama-2-Chat 7B fine-tuned model and the LongChat-32k 7B model [178].

Baselines & Methods

To show the effectiveness of *BumbleBee* for the context (KV cache) summarization task, we compare it to the following baselines: (1) **All**: we use the entire KV cache and do not perform any cache reduction (2) **Local**: only the most recent context x% tokens are maintained in the KV cache and the remaining old KV states are evicted (3) **Random + Local**: randomly selected tokens along with the most recent tokens are retained in the KV cache (4) **Attention sinks + Local**: the first four tokens known as attention sinks [355] along with the most recent tokens are kept in the KV cache (5) **H2 + Local**: only the tokens

that are most frequently attended to, referred to as Heavy Hitters [393] are maintained in the KV cache along with the most recent tokens.

In the case of *BumbleBee*, we experiment with two different concave function choices for Eq. (6.3): (♥) log-based: $\phi(x) = \log(1 + x)$ and (♦) power-based: $\phi(x) = g^{-1}(x)$ where $g(y) = \alpha y^{1/\alpha} + \beta y$. For the experiments in this chapter, we set β and α as 1 and 0.04, respectively, to ensure that the function saturation curve is compatible with that of the facility location function, the first submodular function component of $g_\lambda(\cdot)$ in Eq. (6.7).

Table 6.4: Results on the few shot tasks from the lm-eval-harness benchmark using LLaMA 7B and 13B [320] models. In each of the above methods except All, we perform a 10x context reduction, meaning our KV cache summarization budget is 10% of the input sequence length.

Model	Methods	OpenBookQA	COPA	RTE	MathQA	PiQA	Winogrande
LLaMA-13B	**All**	47.4	85	73.28	31.86	80.36	75.69
	Local	28.4	64	53.43	23.25	58.32	49.88
	Random + Local	27.6	58	54.63	21.76	54.13	50.64
	Attn Sinks + Local	44.4	80	67.51	29.78	79.22	70.48
	H2 + Local	44.2	83	64.98	29.71	**79.49**	70.32
	BumbleBee ♥	**47.6**	**85**	**71.48**	**31.02**	79.38	71.98
	BumbleBee ♦	46.6	83	67.15	30.82	**79.49**	**73.01**
LLaMA-7B	**All**	44.6	81	68.95	29.85	80.03	71.51
	Local	28.4	56	50.90	23.02	58.27	51.38
	Random + Local	28.0	63	51.26	21.76	53.94	49.30
	Attn Sinks + Local	41.6	**82**	58.12	27.40	78.07	67.80
	H2 + Local	41.4	78	63.54	27.50	77.31	65.82
	BumbleBee ♥	**43.2**	79	**68.95**	27.74	78.24	**68.75**
	BumbleBee ♦	**43.2**	79	63.90	**28.51**	**78.56**	68.19

Results

LM-eval-harness Tasks In Table 6.4, we compare *BumbleBee* to existing KV cache reduction methods on five-shot learning tasks from the lm-eval-harness benchmark [96]. Across both LLaMA [320] variants we used for inference, *BumbleBee* consistently outperforms other baselines in terms of accuracy, showing a performance comparable to the best-case

Table 6.5: Results on six datasets from the LongBench benchmark using *Llama2-7B-chat-4k* [321] and *LongChat-v1.5-7B-32k* [178]. * indicates that the reported numbers are sourced directly from [19].

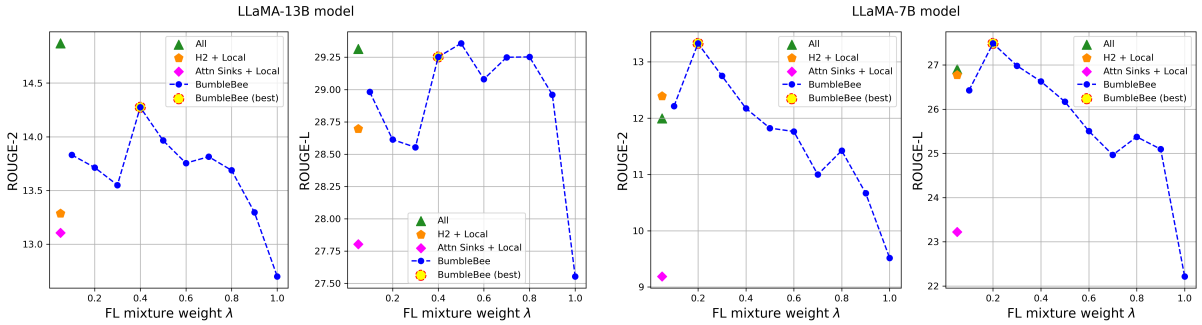
Model	Method	Qasper	MultiFieldQA-en	HotpotQA	2WikiMQA	QMSum	TREC
LLaMA-7B-chat	**All***	19.20	36.80	25.40	32.80	20.80	61.5
4k	**All** (self)	21.60	36.76	27.55	31.58	20.78	64.0
	Attn Sinks + Local	14.74	22.93	22.08	29.73	19.25	56.0
	H2 (20%)	**19.82**	26.60	26.28	25.69	**21.45**	60.0
	BumbleBee (20%) ♥	19.37	27.73	26.14	27.67	20.68	**61.5**
	BumbleBee (20%) ♦	19.59	**28.60**	**28.99**	**30.19**	21.05	59.0
LongChat-7B	**H2** (SW, 20%)	21.64	30.72	14.07	15.10	18.11	40.5
32k	**BumbleBee** (SW, 20%) ♦	**23.27**	**33.16**	**22.52**	**17.58**	**20.27**	**44.5**

no-compute-constraint setting when *All* of the KV cache is used without any summarization/reduction.

LongBench Tasks In Table 6.5, we report the results on six different datasets from LongBench [19]. For all datasets, excluding QMSum and TREC, we use F1 score for evaluation. For QMSum and TREC, we use Rouge-L and accuracy, respectively, as the evaluation metric. When using the *LLaMA-7B-chat-4k* model pre-trained with a 4k context window, we truncate the middle part of the input contexts if their length exceeds 4k as suggested in [19]. As can be seen in Table 6.5, *BumbleBee* outperforms other context reduction methods such as *H2* on four datasets even when the summarization budget is 20% of the entire context size.

However, when using the *LongChat-7B-32k*, which was trained further to generalize to longer sequences, we adopt a Sliding Window (SW) strategy to process smaller chunks (or segments) of longer contexts, aggregate their KV embeddings, and the attention scores accumulated for keys in different chunks. This is done to process the entire context/sequence in a memory-manageable way while avoiding context truncation. This strategy, however, is

sub-optimal as tokens in faraway segments can not attend to tokens at the beginning of the original sequence. Despite this, we observe that *BumbleBee* outperforms the state-of-the-art KV cache reduction method *H2* by strong margins (1.6%-8.5% in absolute terms)



(a) LLaMA-13B (b) LLaMA-7B

Figure 6.5: ROUGE-based results on XSUM dataset, a few-shot summarization task from the HELM benchmark [191] using two different LLaMA models [321]. To reduce the pressure on the context window across all decoder layers, we perform a 5x KV cache reduction for each of the above methods except *All*.

XSUM Summarization Task In Fig. 6.5, we compare different context reduction methods to the full cache setting. Specifically, we use LLaMA2 models [321] to assess the downstream summarization performance in a 3-shot learning setting. *BumbleBee* outperforms other SOTA cache reduction techniques and even performs better than the *All* cache setting when using LLaMA-7B.

Sensitivity analysis of mixture weight λ

We use the XSUM dataset to show the overall performance sensitivity of *BumbleBee* to the convex mixture weight hyperparameter λ in Eq. (6.7). In the case of the LLaMA-13B model, $\lambda \in [0.4, 0.8]$ performs comparably, showing that a representative subset of the KV cache is preferred for the downstream task. In LLaMA-7B, we see that for $\lambda > 0.2$, the performance

starts to drop as we increase the mixture weight λ corresponding to the Facility location function $f(\cdot)$. However, the first submodular component with its relative weight of 0.2 still outperforms the *H2 + Local* method, showing that both representativeness and relevance of the selected cache subset are desirable to maintain a performance comparable to the entire cache setting.

Reproducibility

We implement *BumbleBee* in PyTorch. To update the KV cache in the streaming setting, we modify the *LlamaAttention* class from the huggingface library. For the submodular optimization, we use *submarine* [27], an internal highly optimized submodular toolkit, and we plan to open-source the integrated codebase shortly.

On the LongBench dataset, we first compute the key embeddings for all the context tokens. This is performed for each self-attention head in each decoder layer. We then use stochastic greedy algorithm [234] with $\epsilon = 1\text{e}{-5}$ to compute the offline KV cache summaries for each of $h \times l$ heads in a parallel fashion. Here, h denotes the number of attention heads present in one decoder layer, and l denotes the number of decoder layers.

For similarity matrix computation, we use ReLU truncated cosine similarity to ensure pairwise similarities $\text{sim}(i, j) \geq 0$. We have also experimented with other cosine similarity-based metrics such as $1 + \cos(i, j)$ and $|\cos(i, j)|$ but find that $\text{ReLU}(\cos(i, j))$ works the best.

Mixture weight λ: on the tasks from the lm-eval-harness benchmark [96], $\lambda \in \{0.2, 0.3\}$ perform the best across both evaluated models, i.e., LLaMA 7B and 13B. However, we did not perform a more fine-grained search/tuning for λ in this range. On LongBench [19], we report the results for $\lambda = 0.3$ across all six datasets. For the XSUM [238] few-shot summarization task, we use a held-out validation set to tune the mixture weight and set $\lambda = 0.2$ when using LLaMA2-7B and $\lambda = 0.4$ for LLaMA-13B.

Compute: we use an NVIDIA-A100 GPU to perform our inference-based experiments and a multi-threaded CPU for all submodular computation using Submarine [27]. For the

experiments involving training on synthetic data in Sec. 6.6.1, we use an NVIDIA RTX 2080.

Table 6.6: Decoding speed (in ms/token) for two KV cache reduction ratios (5:1 and 10:1) and the baseline KV cache method using the entire context (1:1) across all heads. All experiments are performed on an A100 80GB GPU using the LongChat-7B-32k with a batch size of 1.

Context reduction ratio	**Original Context Length**	
	16k	**100k**
1:1	59.30 ± 0.39	OOM
5:1	47.49 ± 4.16	71.50 ± 0.10
10:1	39.74 ± 1.31	48.16 ± 0.09

Detailed Analysis of Results

For one of the **lm-eval-harness** [96] tasks, namely COPA, we analyze how the KV cache across different self-attention heads gets updated as new queries are processed in a streaming setting. In Figures 6.6 & 6.7, we show such visualization for two samples selected from the test set of COPA [277]. Since each incoming query attends to its key vector, the offset diagonal line simply indicates that property. Overall, we see that *H2* is heavily biased towards selecting the initial set of keys (close to position 0) and maintaining them in the KV cache. This pattern holds across most of the attention heads. However, *BumbleBee* summaries appear more time-diverse across the majority of heads, showing that our framework is capable of balancing both representativeness and relevance in the final summary of the KV cache.

Next, we visualize how good the subsets selected by Heavy Hitters (H2) and *BumbleBee* are on one of the samples chosen from the 2WikiMultihopQA [117] dataset from the LongBench task. We visualize the t-SNE embeddings of the keys across randomly selected self-attention heads in certain layers, along with the selected subset of the KV cache. Fig. 6.8 shows that *BumbleBee* can select a representative subset of keys from the entire KV cache

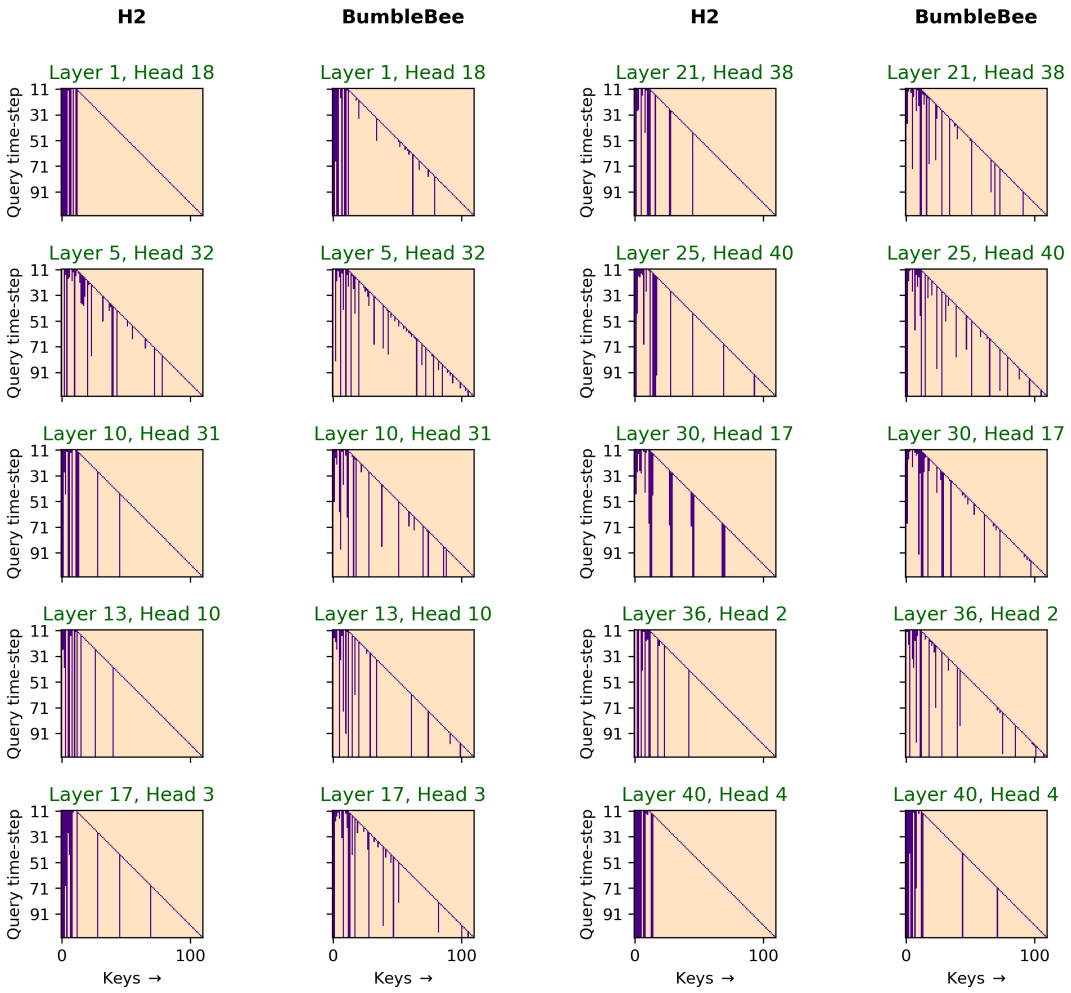


Figure 6.6: Test sample 1: Visualization of the keys selected at different time-steps (y-axis) when using *H2* and *BumbleBee* for online KV cache reduction and summarization, respectively. The x-axis represents the keys, and the y-axis represents the queries. We show the evolution of the selected KV cache (marked by purple points) as we process an incoming query across different attention heads and determine how to update the KV cache under the budget constraint.

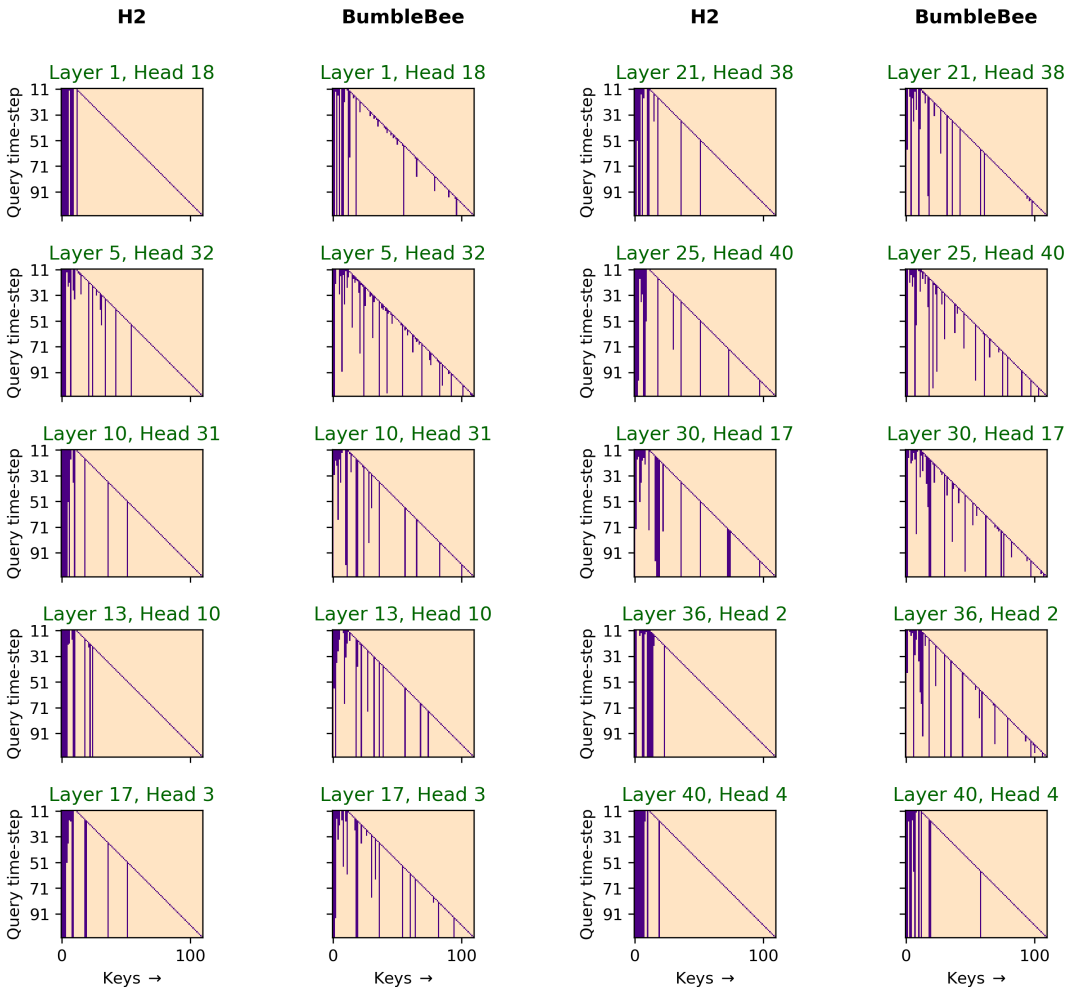


Figure 6.7: Test sample 2: Visualization of the keys selected at different time-steps (y-axis) when using *H2* and *BumbleBee* for online KV cache reduction and summarization, respectively. The x-axis represents the keys, and the y-axis represents the queries. We show the evolution of the selected KV cache (marked by purple points) as we process an incoming query across different attention heads and determine how to update the KV cache under the budget constraint.

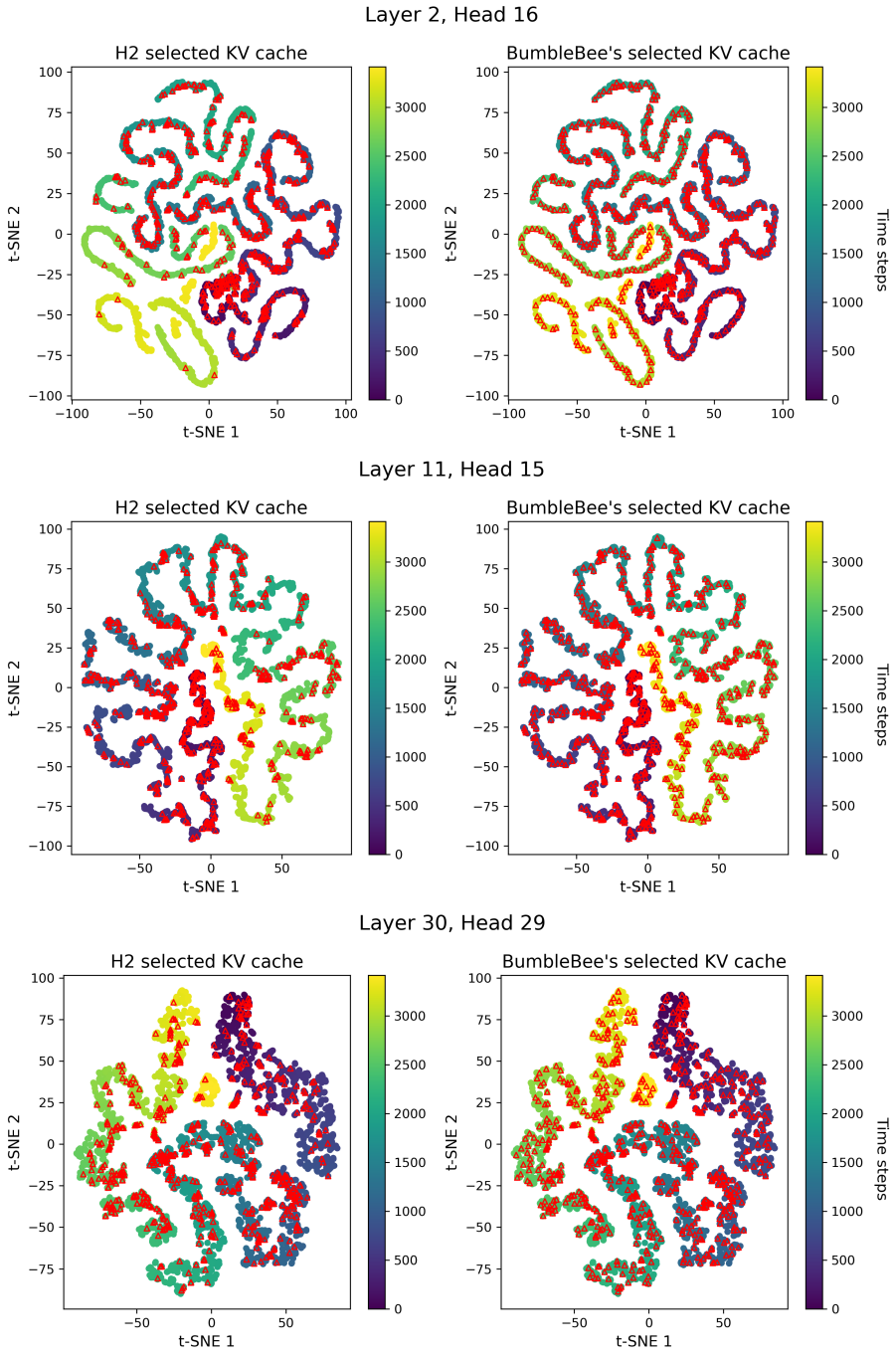


Figure 6.8: t-SNE visualization of the key embeddings for different attention heads in the LLaMA-7B-chat-4k model. The keys selected by *H2* and *BumbleBee* as a part of the final KV cache summary are marked by △.

when using the offline greedy algorithm, thus explaining its strong performance on various LongBench datasets.

Table 6.6 compares the decoding speeds (in ms/token) for two different context reduction ratios, demonstrating faster decoding speed as the context reduction ratio becomes higher.

6.7 Summary

With LLMs having longer contexts, KV cache reduction techniques are being considered to manage their overhead. However, existing techniques do not capture the diversity and long-term dependencies between the tokens in the KV cache. In this chapter, we propose *BumbleBee*, an inference-time KV cache summarization technique that uses a submodular mixture of functions to do so. We show that *BumbleBee* has superior accuracy compared to existing methods on long context benchmarks, with models having a sizeable context window with identical rates of KV cache reduction.

Chapter 7

SUBMODULARITY FOR VISUAL TOKENS SELECTION IN MLLMS

7.1 Introduction

Recent advancements in Large Language Models (LLMs) [259, 36, 320, 83, 244] have driven significant progress in multimodal learning, enabling the extension of powerful reasoning capabilities to Multimodal Large Language Models (MLLMs) [371] such as GPT-4 [245], Gemini 1.5 [314], LLaVA 1.5 [198], and Qwen2-VL [337]. These models integrate a visual encoder (e.g., CLIP [257]) to transform visual inputs into sequential representations and an LLM (e.g., LLaMA-3 [83]) for text generation, facilitating seamless fusion of visual and textual modalities. This integration has led to remarkable advancements in multimodal perception and reasoning, powering applications such as visual question answering [305, 211], document understanding [186, 205], etc. However, a fundamental challenge in MLLMs arises from the high computational cost of processing visual tokens, which often make up the majority of the input sequence. While increasing image resolution can improve model performance [215, 108], it increases the number of visual tokens, further escalating computational overhead—particularly in video-based models such as Video-LLaVA [192].

A key factor contributing to this inefficiency is the quadratic scaling of multi-headed self-attention [328], which causes computational and memory costs to grow non-linearly with sequence length. As MLLMs process increasingly long sequences of visual tokens—whether from high-resolution images [108], detailed document layouts, or multi-frame video inputs [308, 113]—the memory footprint and inference time become prohibitively high. This bottleneck severely limits the scalability and practical deployment of MLLMs in resource-constrained real-world applications such as robotics, autonomous systems, etc., where ef-

ficiency is a critical requirement.

To reduce the inference time and memory footprint of MLLMs, prior works [30, 299, 349, 48, 392, 361, 388] have explored various visual token reduction strategies, broadly categorized into token pruning, token merging, and early-stage token filtering. Token pruning methods, such as SparseVLM [392] and FastV [48], dynamically remove less informative visual tokens based on attention scores, significantly reducing sequence length while maintaining task performance. However, cross-attention-based pruning can be suboptimal due to attention misalignment [388], and pruning too aggressively risks losing crucial visual information.

Token merging approaches, like VisionZip [361], cluster visually similar tokens and replace them with compact representations, preserving semantic content while reducing computational costs. Meanwhile, early-stage token filtering, as seen in FasterVLM [388], removes redundant tokens directly at the visual encoder stage, preventing unnecessary tokens from reaching the LLM and thereby optimizing efficiency. Although FasterVLM [388] does not explicitly formalize the notion of redundancy, it is implicitly defined as tokens that receive low attention from the [CLS] token in the vision encoder. While these methods show notable improvements in reducing FLOPs and improving inference speed, maintaining a balance between efficiency and accuracy remains a key challenge, especially for tasks that require capturing long-range dependencies and fine-grained visual details.

In this chapter, we introduce *VisionBee*, a novel visual token summarization framework that efficiently reduces redundancy in the visual token space using a principled submodular optimization approach. By leveraging submodularity, VisionBee ensures that MLLMs retain critical visual information while significantly improving efficiency. Inspired by the BumbleBee framework described in Sec. 6.5 of Chapter 6, we formulate visual token selection as a subset selection problem, where the utility of each visual token is evaluated in the context of other tokens to optimize the selection process.

Unlike prior methods that rely on text-visual cross-attention in the LLM decoder, VisionBee operates entirely within the vision encoder, making token selection independent of textual inputs. Specifically, as shown in Fig. 7.1, it leverages the key embeddings from a

later layer of the vision encoder along with the attention weights attributed by the [CLS] token to different visual tokens. By integrating key embedding similarity and [CLS] attention scores into a submodular function, VisionBee applies the greedy algorithm described in Alg. 1 to perform cardinality-constrained submodular summarization, ensuring an optimal balance between informativeness and token reduction. This approach allows VisionBee to remove redundant visual tokens before they reach the LLM decoder, achieving a superior efficiency-accuracy trade-off compared to existing methods.

Our contributions are as follows:

1. **Training-free token selection**: VisionBee does not require additional training, thus making it a lightweight, scalable, and efficient token selection strategy for MLLMs. By leveraging the vision encoder's internal representations, it selects informative tokens without encountering the misalignment issues [388] often observed in text-visual cross-attention-based pruning methods. Importantly, VisionBee operates during inference time, significantly reducing compute and memory overhead—an increasingly critical aspect as inference sequence lengths grow larger in modern LLM and MLLM deployments [377, 354].

2. **Broad applicability**: Designed as a model-agnostic framework, our approach is plug-and-play and can be integrated into any vision-language pipeline without modifying the underlying model architecture.

3. **Efficient context reduction with competitive performance**: VisionBee outperforms existing visual tokens pruning and merging approaches while reducing the number of visual tokens by up to 9×, significantly improving inference speed and memory efficiency without compromising accuracy.

This chapter is based on the following ongoing work: **Lilly Kumari** and Jeff Bilmes. *VisionBee: A Submodular Framework for Context Summarization in Multimodal Large Language Models, 2025.*

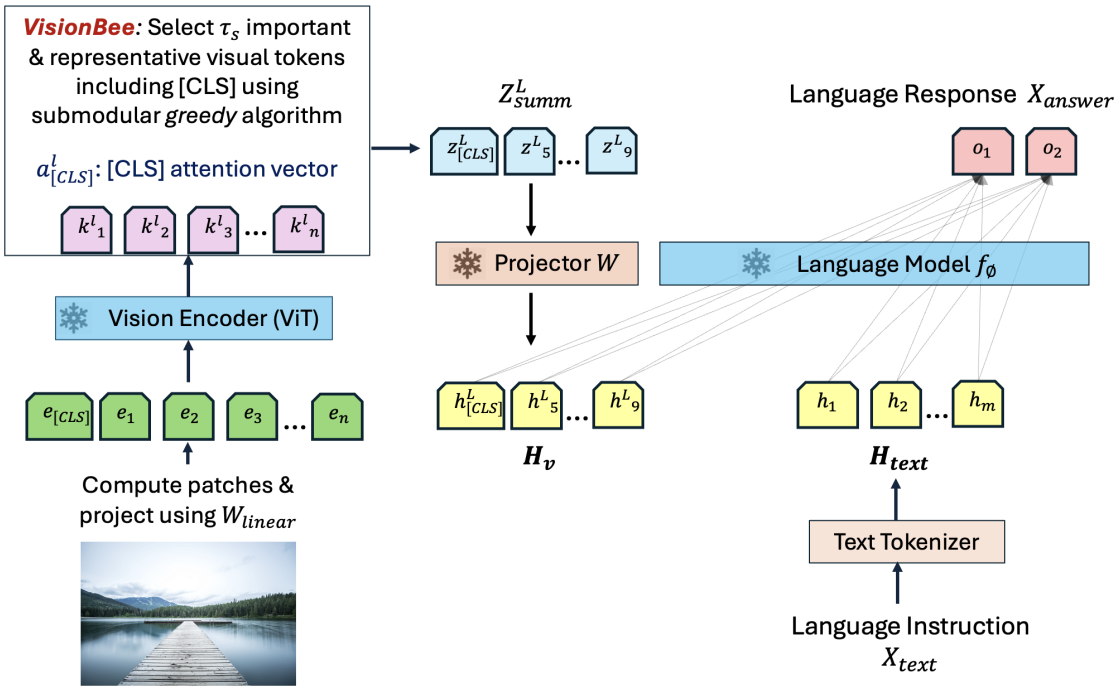


Figure 7.1: Proposed VisionBee Framework: Given an input image, VisionBee first utilizes the Vision Encoder to extract key embeddings and [CLS] attention weights for all visual tokens from a selected layer l. These features are used to instantiate a submodular mixture function that captures both diversity and importance. A greedy selection algorithm is then applied to summarize the n visual tokens into a compact subset constrained by a summarization budget τ_s. The selected visual tokens are subsequently projected through the multimodal alignment module and combined with the language instruction tokens before being passed to the LLM decoder for response generation.

7.2 Notations & Background

MLLMs such as LLaVA 1.5 [200, 198] consist of two transformer-based components: a visual encoder [257] and a LLM decoder [321, 51, 83]. The visual encoder (e.g., Vision Transformer (ViT) [81, 257]) processes an image by first dividing it into fixed-size non-overlapping patches, which are then linearly projected into embeddings. These patch embeddings, along with a special [CLS] token embedding, are concatenated and passed through a multi-headed self-attention mechanism, where each token attends to all others, capturing global context. In the self-attention head, the input embedding matrix $X \in \mathbb{R}^{(n+1)\times d_k}$—comprising n patch embeddings and one **[CLS]** embedding—is projected into query, key, and value matrices using learnable weights:

$$Q = XW_q, \quad K = XW_k, \quad V = XW_v \tag{7.1}$$

where $W_q, W_k, W_v \in \mathbb{R}^{d_k \times d}$ are the projection matrices and d is the dimensionality of the self-attention head.

The scaled dot-product attention logits are computed as:

$$A[i,j] = \frac{Q[i] \cdot K[j]}{\sqrt{d}}, \quad i,j = 1,\ldots,n+1, \tag{7.2}$$

followed by softmax normalization over j to obtain the attention weights:

$$\alpha[i,j] = \frac{\exp(A[i,j])}{\sum_{j'=1}^{n+1} \exp(A[i,j'])}. \tag{7.3}$$

Here, each attention weight $\alpha[i,j]$ represents the extent to which token i (acting as a query) attends to token j (acting as a key). In particular, the first row $\alpha[0,j]$ captures how strongly the [CLS] token attends to each visual patch token j. This attention distribution helps understand which parts of the image contribute most to the final representation, making it useful for downstream tasks such as token selection, pruning, or summarization in vision-language models.

The alignment module, for example, the linear layer in LLaVA [200], bridges the gap between visual embeddings produced by the vision encoder and the token embeddings used by

the LLM. Since the visual token representations generated by the vision encoder are typically in a different feature space than the word embeddings of the LLM, the alignment module aligns them with the LLM's input embedding space. This ensures that the vision and text modalities can be processed jointly, allowing for effective cross-modal reasoning. Some models, such as BLIP-2 [181] and Qwen2-VL [337], employ query-based alignment mechanisms to enhance interaction between the two modalities, refining how visual information is passed to the LLM decoder.

Once the visual tokens are aligned with text tokens, they are fed into the LLM decoder, which integrates both modalities to generate responses. The LLM processes the sequence of visual and textual tokens using multi-headed self-attention, enabling it to attend to both visual information and prior textual context when making predictions. Importantly, the LLM decoder uses causal attention, meaning that each token can only attend to the provided tokens in the sequence and previously generated ones but not future ones. This ensures that during autoregressive generation, the model produces responses without leaking information from later positions. For multimodal tasks where visual tokens are referenced by textual tokens to answer multimodal queries such as image captioning, visual question answering (VQA), or document understanding, the memory and compute costs grow quadratically with respect to the sequence length l_{seq} where $l_{\text{seq}} = l_{\text{visual}} + l_{\text{textual}}$. Since l_{visual} is several orders of magnitude larger than l_{textual}, efficient visual token selection is crucial for reducing inference latency and optimizing associated memory costs.

7.3 Related Work

In this section, we first introduce different architectural components of modern large multimodal models in the vision-language space in Sec. 7.3.1. We, then, cover existing state-of-the-art methods for visual token pruning and merging in Sec. 7.3.2 and Sec. 7.3.3, respectively.

7.3.1 Multimodal Large Language Models (MLLMs)

MLLMs [371, 314, 198, 337] extend the capabilities of Large Language Models (LLMs) to multimodal tasks by incorporating visual inputs alongside textual inputs, enabling applications such as image captioning, visual question answering, and video understanding [305, 211, 308, 113]. These models typically consist of a visual encoder (e.g. CLIP [257]), an LLM decoder for text generation, and an alignment module such as a simple linear layer [200], MLP projector [198], or a deep query network [181, 337] to align the visual token and textual token representations. A fundamental challenge in MLLMs arises from the rapid increase in token length when processing high-resolution images and videos, leading to significant computational and memory overhead. For instance, LLaVA [198] encodes a 336×336 image into 576 tokens, while LLaVA-NeXT [199] expands this to 2880 tokens for 672×672 images. This challenge becomes even more pronounced in video-based MLLMs, such as Video-LLaVA [192] and Video-Poet [148], where multiple frames contribute hundreds of thousands to millions of additional visual tokens, making inference computationally challenging in terms of memory bottlenecks and computational inefficiencies (e.g., a 1-hour video sampled at 1 frame per second with 576 tokens per frame results in over 2 million visual tokens). As MLLMs continue to scale, addressing these challenges requires principled approaches that optimize inference efficiency while maintaining task performance. This is particularly critical in resource-constrained environments with memory and compute limitations.

7.3.2 Visual Token Pruning for MLLMs

A common approach to visual tokens compression involves selectively pruning visual tokens with low importance based on their attention weights. FastV [48] identifies redundant visual tokens in MLLMs as those receiving low attention based on the intuition that such tokens contribute less to the final output. Although it does not explicitly formalize redundancy, it adopts an implicit definition guided by attention scores—pruning low-attention tokens after layer 2 of the LLM under the observation that deeper layers progressively assign diminishing

attention to less informative visual tokens. SparseVLM [392] adopts a text-guided token pruning approach, progressively sparsifying visual tokens based on their text-vision cross-attention scores, which we cover in detail in Sec. 7.4. This method dynamically determines the sparsification ratio at each layer based on the rank of attention matrices. However, recent studies such as [388] have shown that cross-attention does not always align well with true visual token importance due to attention drift, where attention in the LLM decoder gradually shifts toward later-position tokens irrespective of their semantic relevance, and attention dispersion, where attention in the LLM decoder becomes broadly spread across many tokens, including irrelevant ones like padding regions. In contrast, [CLS] attention in the vision encoder tends to be more focused on semantically meaningful regions. To address this, FasterVLM [388] proposes an alternative strategy, leveraging the [CLS] token attention from the vision encoder to rank and prune visual tokens before they are processed by the LLM. We discuss it in detail in Sec. 7.4.

7.3.3 Visual Token Merging for MLLMs

Instead of simply pruning tokens, another line of research focuses on merging redundant tokens while preserving their semantic content. Existing token merging strategies in MLLMs primarily fall into two categories: Bipartite Soft Matching (BSM) approaches and clustering-based merging techniques. ToMe (Token Merging) [30] employs a BSM algorithm to merge visual tokens using key-states similarity at each transformer block, effectively reducing redundancy—implicitly defined as overlapping semantic content between image tokens with similar key representations—while preserving multimodal coherence. In contrast, clustering-based methods group visually similar tokens and replace them with more compact representations, enhancing efficiency without significantly impacting model performance. LLaVolta [47] introduces a visual context compressor, which applies average pooling at the vision encoder's output to progressively reduce token redundancy while maintaining task accuracy.

VisionZip [361] further advances this concept by incorporating a clustering-based token aggregation mechanism, achieving up to 8× computational speedup while retaining over

94% of the model's original accuracy. Besides the token merging mechanism, we cover its attention-based token pruning algorithm in Sec. 7.4. SparseVLM [392] enhances token sparsification by adopting k-nearest neighbor (kNN) density peak aggregation, a method that identifies and merges redundant tokens based on local feature density, allowing for a more structured merging process while preserving fine-grained visual details.

7.4 Set Function Optimization View of Prior Work

SparseVLM [392] proposes a training-free visual token pruning strategy leveraging cross-modal attention mechanisms in MLLMs. It introduces a two-stage scoring pipeline. First, it selects the most relevant text tokens using *Visual* $\rightarrow$ *Text* attention, where the goal is to identify the most semantically relevant text tokens. Let $Q_V \in \mathbb{R}^{n\times d}$ be the embedding of the n vision tokens after projection into the shared embedding space of the MLLM and $K_T \in \mathbb{R}^{m\times d}$ be the embeddings of the m text tokens where d denotes the embedding dimension. The cross-attention logits are computed as:

$$A_{VT}[i,j] = \frac{Q_V[i]\cdot K_T[j]}{\sqrt{d}}, \quad i = 1,\ldots,n, \quad j = 1,\ldots,m, \tag{7.4}$$

followed by row-wise softmax over j to obtain normalized attention weights:

$$\alpha_{VT}[i,j] = \frac{\exp(A_{VT}[i,j])}{\sum_{j'=1}^{m}\exp(A_{VT}[i,j'])}. \tag{7.5}$$

The importance score for each text token t_j is then computed by aggregating its attention weight across all visual tokens: $s_j = \sum_{i=1}^{n}\alpha_{VT}[i,j]$. The top-$k$ text tokens with the highest importance score s_j are selected as the most semantically relevant for text-guided visual token pruning.

In the second stage, *Text* $\rightarrow$ *Visual* attention is used to assess the importance of each visual token conditioned on the top-k selected text tokens. Let $Q_T \in \mathbb{R}^{k\times d}$ be the queries from the selected text tokens and $K_V \in \mathbb{R}^{n\times d}$ be the keys from visual tokens in an LLM decoder block. The attention logits are computed as:

$$A_{TV}[i,j] = \frac{Q_T[i]\cdot K_V[j]}{\sqrt{d}}, \quad i = 1,\ldots,k, \quad j = 1,\ldots,n, \tag{7.6}$$

with row-wise softmax over j to obtain normalized attention weights:

$$\alpha_{TV}[i,j] = \frac{\exp(A_{TV}[i,j])}{\sum_{j'=1}^{n} \exp(A_{TV}[i,j'])}. \tag{7.7}$$

Each visual token v_j is then assigned an importance score $w_j = \sum_{i=1}^{k} \alpha_{TV}[i,j]$, representing the cumulative attention it receives from the selected text tokens. Visual token pruning is performed by retaining the top-r tokens with the highest w_j scores, and this process is applied progressively across layers of the MLLM decoder.

The pruning mechanism naturally defines a set scoring function over subsets of visual tokens. Given a subset $\mathcal{I} \subseteq \{1, \ldots, n\}$ of selected visual tokens, the overall utility of the subset is defined as:

$$f(\mathcal{I}) = \sum_{j \in \mathcal{I}} w_j \tag{7.8}$$

This set function is modular as it is a linear sum of individual accumulated attention weights from relevant text tokens to each visual token. Adding more visual tokens simply has an additive effect and does not take into account the interaction between different selected visual tokens in S.

FasterVLM [388] is also a training-free visual token pruning strategy similar to SparseVLM. However, instead of relying on text-visual cross-attention from the LLM decoder, it leverages the [CLS] token attention from the vision encoder itself. The motivation stems from the observation that cross-attention computed within the LLM is subject to undesirable phenomena such as *attention shift* and *attention dispersion*, which degrade pruning accuracy and result in significant performance drops at high token reduction ratios. In FasterVLM, pruning is performed based on the attention attributed by the [CLS] token to different visual tokens and is done *before* the multimodal projector and the LLM, resulting in substantial reductions in inference costs.

Formally, let $Z^L_{\text{visual}} = \{z^L_1, z^L_2, \ldots, z^L_n\}$ denote the set of hidden states from the final layer of the vision encoder, corresponding to the sequence of n visual tokens. Here, L denotes the number of layers in the vision encoder. Let $q^l_{[\text{CLS}]} \in \mathbb{R}^d$ denote the query vector of the [CLS]

token in the l-th layer of the vision encoder, which is typically the penultimate layer meaning $l = (L-1)$ and $K^l = \{k_1^l, \ldots, k_n^l\} \in \mathbb{R}^{n \times d}$ denote the key embeddings corresponding to the n visual tokens. The attention logits using the [CLS] token as query are computed as:

$$A_{[\text{CLS}]}^l[j] = \frac{q_{[\text{CLS}]}^\top k_j}{\sqrt{d}}, \quad j = 1, \ldots, n, \tag{7.9}$$

followed by softmax over the column index j to obtain normalized attention weights:

$$\alpha_{[\text{CLS}]}^l[j] = \frac{\exp(A_{[\text{CLS}]}^l[j])}{\sum_{j'=1}^n \exp(A_{[\text{CLS}]}^l[j'])}. \tag{7.10}$$

To aggregate attention across multiple heads, an attention score vector $a_{[\text{CLS}]}^l \in \mathbb{R}^n$ is computed, where each element $a_{[\text{CLS}],j}^l$ represents the sum (or mean) of attention weights that visual token j receives from the [CLS] query across all self-attention heads in the l-th layer:

$$a_{[\text{CLS}],j}^l = \sum_{h=1}^H \alpha_{[\text{CLS}]}^{l,h}[j], \tag{7.11}$$

where H is the number of heads.

The importance score of each visual token j is thus given by $a_{[\text{CLS}],j}^l$, indicating how strongly the [CLS] token attends to that visual token. The token selection process aims to find a subset of token indices $\mathcal{I}_{\text{selected}} \subseteq \{1, \ldots, n\}$, constrained by $|\mathcal{I}_{\text{selected}}| \leq k$, that maximizes the total attention score:

$$\mathcal{I}_{\text{selected}} \in \underset{\mathcal{I} \subseteq \{1,\ldots,n\},\ |\mathcal{I}| \leq k}{\text{argmax}} \sum_{j \in \mathcal{I}} a_{[\text{CLS}],j}^l. \tag{7.12}$$

The final token representations selected for feeding into the multimodal projector and LLM are given by $Z_{\text{selected}}^L = \{z_j^L \mid j \in \mathcal{I}_{\text{selected}}\}$.

This selection strategy defines a set scoring function over subsets of visual tokens. Given a subset $\mathcal{I} \subseteq \{1, \ldots, n\}$, the utility of the subset is given by:

$$f(\mathcal{I}) = \sum_{j \in \mathcal{I}} a_{[\text{CLS}],j}^l. \tag{7.13}$$

Again, similar to SparseVLM, this function is modular as it is a linear sum of individual [CLS] attention-based importance scores and does not take into account setwise interactions amongst its constituents.

VisionZip [361] also performs early-stage visual token selection at the vision encoder stage in a training-free manner. It adopts the same pruning strategy as FasterVLM to select 85% of the overall visual token budget, leveraging a modular set function that we defined in Eq. (7.13). For the remaining 15% of tokens, VisionZip employs a heuristic-based token merging strategy, to further compress the visual representation while preserving semantic content. Specifically, it initializes cluster centroids by selecting k target visual tokens with evenly spaced indices from the full sequence of visual token indices. Each remaining token is then assigned to the closest target token based on the similarity in the key embedding space extracted from the penultimate layer of the vision encoder. Finally, the final layer representations of each target token (cluster centroid) are updated by averaging the representations of all tokens assigned to its cluster, compressing redundant visual information.

7.5 [CLS] Attention in MLLMs

To analyze the [CLS] attention distribution in MLLMs, we first randomly sample $1,000$ image-text pairs from the LLaVA-mix665k data [198] similar to [388]. We use these samples to prompt LLaVA-1.5-7B to generate textual response and also extract the [CLS] attention across different layers of LLaVA's vision encoder, which is based on the ViT architecture.

In Fig. 7.2, we plot the histograms of [CLS] attention scores across different layers of the vision encoder (ViT). The x-axis represents the [CLS] attention scores, while the y-axis denotes the proportion of visual tokens that receive a given attention value. In early layers, the attention distribution is broad and spread out, meaning the [CLS] token attends almost uniformly to a wide range of visual tokens. This suggests that the model has not yet identified the most critical visual regions and is still in the feature extraction stage, where it treats all visual tokens with relatively similar importance. As we move to mid-level layers (Layers 13-17), the [CLS] attention distribution becomes increasingly sparse, as a majority of the

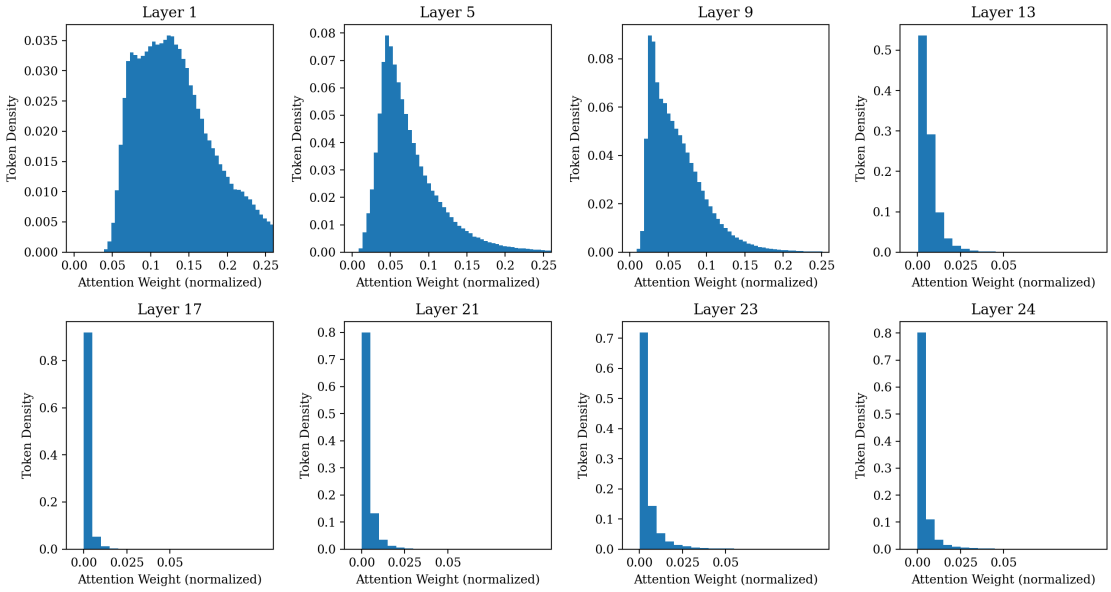


Figure 7.2: [CLS] attention histogram across different layers of the vision encoder.

visual tokens receive near-zero attention. Finally, in the late layers (Layers 21-24), the [CLS] token strongly focuses on only a few selected tokens, with the penultimate layer (Layer 23) demonstrating the sharpest separation between attended and ignored tokens. This indicates that at Layer 23, the model has optimally filtered out unimportant tokens while retaining key visual information. However, the [CLS] attention in the last layer (Layer 24) is more restrictive, focusing on only a few key tokens while suppressing most of the visual field and becoming over-specialized with respect to the task.

In Fig. 7.3, we analyze the progression of [CLS] attention by tracking the mean attention scores of the top-10 attended tokens, along with the standard deviation range across layers. The plot shows that in early layers, the mean attention remains low and stable, reflecting the model's uniform focus on all tokens. However, from the middle layer (Layer 12) onward, there is a sharp increase in the mean attention values, indicating that the [CLS] token started focusing on a smaller, more relevant subset of visual tokens. The standard deviation (shaded region) is highest amongst mid-layers (Layers 14-18), suggesting that different images still

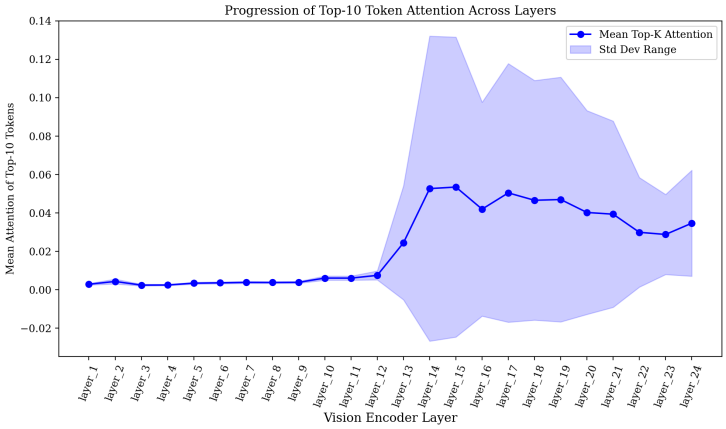


Figure 7.3: Progression of [CLS] attention: mean and variability of attention weights attributed to top-10 visual tokens across different vision encoder layers.

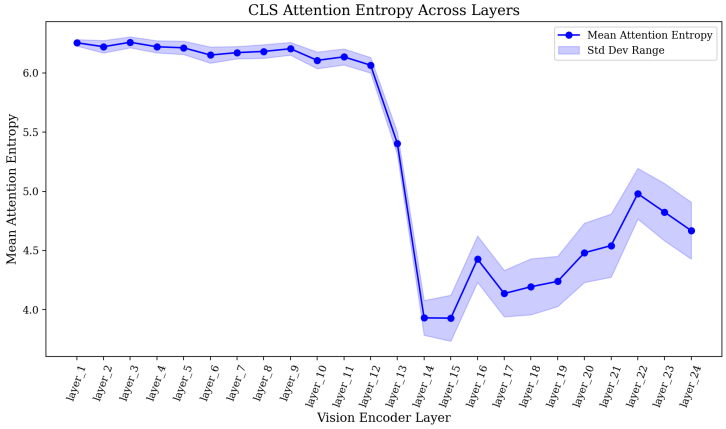


Figure 7.4: Progression of [CLS] attention entropy: mean and variability of [CLS] attention entropy across different vision encoder layers.

exhibit some variability in their attention patterns during these layers. By Layer 20, the mean top-10 attention weights stabilize, meaning that the model has consistently identified the most crucial tokens across different images.

In Fig. 7.4, we plot the [CLS] attention entropy across layers, where entropy quantifies the spread of attention across visual tokens. In early layers, entropy is high and stable, confirming that the [CLS] token initially distributes attention uniformly. However, there is a sharp drop in entropy in the mid-layers (Layers 12-14), signaling a transition where the model begins filtering out redundant tokens and prioritizing important regions. Entropy remains low in deeper layers, but the penultimate layer (Layer 23) shows a more balanced state, indicating that the model has refined its attention without over-compressing information. The last layer (Layer 24), in contrast, shows a further drop, suggesting restrictive attention. This further supports the idea that the [CLS] attention in the penultimate layer can help identify important visual tokens without losing essential visual details.

7.6 VisionBee

To address the computational bottlenecks in MLLMs caused by long sequences of visual tokens, we introduce VisionBee, a principled visual token selection framework based on submodular optimization. Inspired by the BumbleBee framework for KV cache summarization in LLMs (Chapter 6, Sec. 6.5), VisionBee formulates visual token selection as a cardinality-constrained submodular maximization problem. It balances two key objectives: **importance**, quantified via [CLS] attention weights, and **diversity**, captured through key embedding similarity.

While BumbleBee operates on textual KV cache representations during autoregressive decoding, VisionBee targets the visual token space during the prefill stage, allowing for early-stage summarization before visual tokens are even passed to the LLM decoder. This makes VisionBee model-agnostic and training-free, requiring no architectural modifications to MLLMs and ensuring easy integration across vision-language pipelines.

As observed in our empirical analysis in Sec. 7.5, [CLS] attention weights from deeper

layers of the vision encoder (e.g., Layer 23 in case of LLaVA-1.5-7B) exhibit strong sparsity and selectivity, clearly distinguishing between informative and redundant visual tokens. However, relying on attention alone for token selection may result in redundancy, where multiple tokens from similar regions receive high scores. To counter this, VisionBee integrates both [CLS] attention and key embedding diversity into a unified selection objective. The goal is to select a diverse set of informative visual tokens, maximizing inference efficiency while preserving task performance.

[CLS]-based Attention Importance Function While [CLS] attention scores inherently provide a token-level importance signal, directly summing them—as in a modular function—does not account for diminishing returns in the selection process. Similar to BumbleBee, we transform this modular importance signal into a submodular scoring function by applying a concave transformation over the accumulated attention scores, ensuring diminishing marginal gains as more tokens are selected.

Formally, let $a^l_{[\text{CLS}],j}$ denote the attention weight attributed to the j-th visual token by the [CLS] query in the penultimate layer $l = (L-1)$ and computed using Eq. (7.11). Here, $j \in \{1, 2, ..;n\}$ and n denotes the number of visual tokens. Using a monotone non-decreasing concave function h, we transform this modular function into a feature-based submodular function defined in Sec. 2.1.2. Using this, the attention-based utility of a subset $\mathcal{I} \subseteq \{1, \ldots, n\}$ is defined as:

$$c(\mathcal{I}) = h(\sum_{j \in \mathcal{I}} a^l_{[\text{CLS}],j}). \tag{7.14}$$

Key Embedding Diversity Function To prevent redundancy among highly attended tokens, VisionBee utilizes the facility location function defined in Sec. 2.1.2 over the key embeddings from the penultimate layer $l = (L-1)$. This function encourages the selection of a diverse set of representative tokens and is defined as:

$$f(\mathcal{I}) = \sum_{j \in [n]} \max_{i \in \mathcal{I}} \text{sim}(k^l_i, k^l_j), \tag{7.15}$$

where $\text{sim}(\cdot,\cdot)$ denotes cosine similarity after non-negative transformation, and k_i^l denotes the mean key vector of the i-th visual token, averaged across the self-attention heads in the l-th layer of the vision encoder. This ensures that the selected subset of visual tokens covers diverse regions in the key embedding space, promoting wider representation.

Algorithm 6 Offline Submodular Visual Token Summarization during Prefill/Encoding Phase

Input: Submodular functions f and c capturing diversity in the visual tokens embeddings space and importance in terms of attention weight respectively for penultimate layer $l = (L-1)$ of vision encoder, mixture function $g_\lambda(\cdot) = \lambda f(\cdot) + (1-\lambda)c(\cdot)$, a set of n l-th layer key states $K^l = \{k_1^l, k_2^l, \ldots, k_n^l\}$ corresponding to n visual tokens, budget τ_s

Output: A final summary set of indices $S_{\text{idx}} \subseteq [n]$ and corresponding set of visual token representations $Z^L_{\text{summ}} = \{z_i^L | i \in S_{\text{idx}}\} \cup \{z^L_{[\text{CLS}]}\}$ such that $|Z^L_{\text{summ}} \cup z^L_{[\text{CLS}]}| \leq \tau_s$ where z_i^L and $z^L_{[\text{CLS}]}$ denotes the attention output/state of the i-th patch and the [CLS] token respectively after passing through the last vision encoder layer denoted by L.

Initialize:

Set the initial summary set: $S_{\text{idx}} = \emptyset$.

Compute the attention score vector $a^l_{[\text{CLS}]} \in \mathbb{R}^n$, where each element $a^l_{[\text{CLS}],i}$ represents the sum of attention weights that patch (key) i receives from the [CLS] query across all self-attention heads in the l-th layer.

Instantiate the submodular function f defined in Eq. (7.15) using the key states K^l.

Instantiate the submodular function c defined in Eq. (7.14) using the modular attention score vector $a^l_{[\text{CLS}]}$.

Instantiate the submodular mixture function g_λ as a convex combination of the functions f and c with mixture weights λ and $1-\lambda$, respectively.

Run greedy algorithm (Alg. 1 in Chapter 2) using the submodular function g_λ with cardinality constraint $(\tau_s - 1)$ and return the summary set S_{idx}

Mixture Function & Submodular Optimization Similar to BumbleBee (Sec. 6.5), to capture both importance and representativeness in the final selected summary of visual tokens, VisionBee defines a mixture submodular function as:

$$g_\lambda(\mathcal{I}) = \lambda\, f(\mathcal{I}) + (1-\lambda)\, c(\mathcal{I}), \tag{7.16}$$

where $\lambda \in [0,1]$ is a hyperparameter controlling the trade-off between importance and diversity. As both components are monotone and normalized, g_λ inherits submodularity and guarantees near-optimal summarization using the greedy algorithm defined in Algorithm 1.

Given the monotone submodular mixture function g_λ, VisionBee starting with an empty set S_{idx} applies the greedy algorithm (Algorithm 1) under a fixed budget $(\tau_s - 1)$ selecting tokens iteratively by maximizing the marginal gain as shown below:

$$i_{\max} \leftarrow \underset{i \in [n] \setminus S_{\text{idx}}}{\operatorname{argmax}} \; g_\lambda(S_{\text{idx}} \cup \{i\}) - g(S_{\text{idx}}) \tag{7.17}$$

This selection continues until $|S_{\text{idx}}| = \tau_s - 1$, after which the [CLS] token is included in the final visual token summary set. This process is outlined in Algorithm 6.

7.7 Experiments

In this section, we validate our proposed framework, *VisionBee*, on different visual question understanding datasets against several state-of-the-art visual token pruning and merging strategies designed for MLLMs.

Datasets We evaluate VisionBee on the following visual question-answering datasets:

1. **POPE** [187]: The POPE benchmark evaluates object hallucination in vision-language models through binary questions that assess whether specific objects are present in an image. Built using images from the MSCOCO dataset, it contains 8,910 image-question pairs. The evaluation relies on Accuracy, Precision, Recall, and F1 score, with the final performance reported as the average F1 score across three sampling strategies. POPE provides a rigorous measure of a model's ability to ground its predictions in actual visual

content, making it particularly effective for identifying hallucination-related errors in multimodal systems.

2. **TextVQA** [305]: The TextVQA benchmark assesses a model's ability to perform visual-text reasoning by answering questions based on text embedded within images, such as signs and billboards. It consists of $5,000$ image-question pairs sourced from the Open Images v3 dataset. The task emphasizes OCR integration and contextual language understanding, with answers requiring either direct text extraction or reasoning over visual-text content.

3. **ScienceQA** [211]: ScienceQA is a multimodal benchmark designed to evaluate scientific reasoning and generalization across natural, language, and social sciences. It features hierarchically organized multiple-choice questions, with the SQA-IMG subset comprising $2,017$ image-question pairs. The task is framed as a multiple-choice question-answering problem, targeting multimodal understanding and multi-step scientific reasoning.

4. **GQA** [119]: GQA is a benchmark for evaluating structured scene understanding and visual reasoning using images and scene graph annotations from the Visual Genome dataset. Questions are automatically generated based on these semantic scene graphs, ensuring grounded and interpretable reasoning paths. GQA requires models to demonstrate fine-grained visual comprehension and logical inference. Performance is evaluated on the test-dev set containing $12,578$ image-question pairs, using answer accuracy as the evaluation metric.

Models We use VisionBee across two LLaVA-1.5 [198] models, namely LLaVA-1.5-7B and LLaVA-1.5-13B. For inference, we follow the settings as in the original paper.

Baselines We compare VisionBee to three different token selection methods: (1) SparseVLM [392], (2) FasterVLM [388], and (3) VisionZip [361]. The first one leverages cross-

Table 7.1: Performance on LLaVA-1.5-7B: The upper bound is 576 tokens extracted from 336×336 image. For each method, the first row reports the benchmark accuracy, while the second row indicates the percentage of upper bound accuracy retained. The last column shows the average percentage of accuracy retained across all benchmarks.

Method	**POPE**	**VQA$^{\text{Text}}$**	**SQA**	**GQA**	**Avg.**
Upper Bound, 576 Tokens **(100%)**					
Vanilla	85.70	58.30	69.50	61.94	100%
	100%	100%	100%	100%	
Retain 192 Tokens (↓ **66.7**%)					
SparseVLM	81.90	**57.66**	69.10	57.18	96.55%
	95.57%	98.90%	99.42%	92.32%	
FasterVLM	85.30	57.60	69.04	59.30	98.35%
	99.53%	98.80%	99.34%	95.74%	
VisionZip	85.50	57.18	**69.33**	59.22	98.31%
	99.77%	98.08%	99.76%	95.61%	
VisionBee	**86.90**	57.60	69.21	**59.90**	**99.12%**
	101.40%	98.80%	99.58%	96.71%	
Retain 128 Tokens (↓ **77.8**%)					
SparseVLM	80.10	**57.53**	**69.10**	55.84	95.43%
	93.47%	98.68%	99.42%	90.15%	
FasterVLM	82.40	57.02	68.75	57.90	96.59%
	96.15%	97.80%	98.92%	93.48%	
VisionZip	83.10	56.78	**69.10**	57.63	96.85%
	96.97%	97.39%	100%	93.04%	
VisionBee	**86.50**	57.09	**69.10**	**59.37**	**98.53%**
	100.93%	97.92%	99.42%	95.85%	
Retain 64 Tokens (↓ **88.9**%)					
SparseVLM	53.00	51.30	**69.27**	46.18	81.02%
	61.84%	87.99%	99.67%	74.56%	
FasterVLM	76.60	55.29	68.98	55.01	93.07%
	89.38%	94.84%	99.25%	88.81%	
VisionZip	76.90	55.65	68.98	55.13	93.36%
	89.73%	95.45%	99.25%	89.00%	
VisionBee	**83.50**	**55.87**	69.15	**57.52**	**96.41%**
	97.43%	95.83%	99.50%	92.86%	

Table 7.2: Performance on LLaVA-1.5-13B: The upper bound is 576 tokens extracted from 336×336 image. For each method, the first row reports the benchmark accuracy, while the second row indicates the percentage of upper bound accuracy retained. The last column shows the average percentage of accuracy retained across all benchmarks.

Method	POPE	VQA^{Text}	SQA	GQA	Avg.
Upper Bound, 576 Tokens **(100%)**					
Vanilla	85.80	61.32	72.77	63.25	100%
	100%	100%	100%	100%	
Retain 192 Tokens (↓ **66.7**%)					
SparseVLM	82.20	**60.11**	73.94	58.67	97.05%
	95.80%	98.03%	101.61%	92.76%	
FasterVLM	84.70	59.31	**74.64**	59.30	97.94%
	98.72%	96.72%	102.57%	93.75%	
VisionZip	84.70	59.36	73.94	59.12	97.65%
	98.72%	96.80%	101.61%	93.47%	
VisionBee	**86.80**	59.39	74.52	**59.77**	**98.73%**
	101.17%	96.85%	102.40%	94.50%	
Retain 128 Tokens (↓ **77.8**%)					
SparseVLM	80.90	**60.19**	73.35	57.89	96.20%
	94.29%	98.16%	100.80%	91.53%	
FasterVLM	82.20	59.14	73.94	58.13	96.44%
	95.80%	96.44%	101.61%	91.91%	
VisionZip	82.60	58.72	**74.29**	57.87	96.40%
	96.27%	95.76%	102.09%	91.49%	
VisionBee	**86.30**	59.11	74.11	**59.34**	**98.16%**
	100.58%	96.40%	101.84%	93.82%	
Retain 64 Tokens (↓ **88.9**%)					
SparseVLM	63.30	53.94	71.78	50.52	85.06%
	73.78%	87.96%	98.64%	79.87%	
FasterVLM	75.70	57.57	73.94	56.23	93.16%
	88.23%	93.88%	101.61%	88.90%	
VisionZip	75.70	57.44	**74.52**	56.18	93.28%
	88.23%	93.67%	102.40%	88.82%	
VisionBee	**83.70**	**58.01**	73.99	**58.30**	**96.50%**
	97.55%	94.60%	101.68%	92.17%	

attention in the MLLMs' language decoder module, while the last two filter out or merge visual tokens before they reach the language decoder. Additionally, for each LLaVA model, we also present the results corresponding to using all visual tokens, meaning no token selection or pruning, which is the upper bound.

Results In Table 7.1 and 7.2, we summarize the performance of various visual token reduction methods on LLaVA-1.5-7B and LLaVA-1.5-13B, respectively, across four benchmarks. The upper bound corresponds to the baseline performance using all 576 visual tokens, which are extracted from a 336×336 input image. We compare the impact of token summarization (reduction) at three different levels—192 tokens (3× reduction), 128 tokens (4.5× reduction), and 64 tokens (9× reduction)—to assess the efficiency-accuracy trade-off of each method.

Across both model sizes, VisionBee consistently achieves the highest or near-highest accuracy retention across all datasets and reduction ratios. Notably, at the most aggressive setting of 9× context reduction, VisionBee maintains 96.4% and 96.5% of the upper bound accuracy on LLaVA-1.5-7B and LLaVA-1.5-13B, respectively—significantly outperforming other baselines like SparseVLM, FasterVLM, and VisionZip. This highlights the robustness of our submodular selection strategy in preserving essential visual content, even under extreme summarization budgets.

Furthermore, even at lower summarization levels (3× and 4.5× reduction), VisionBee consistently outperforms or matches other methods, demonstrating strong generalization across both hallucination detection (POPE) and reasoning-intensive tasks (SQA, GQA). This performance gain is achieved without requiring any additional training, making VisionBee a plug-and-play and training-free solution for efficient visual token summarization in MLLMs.

These results confirm that VisionBee not only reduces the context length allocated to visual tokens significantly but also maintains high performance across diverse multimodal tasks, ensuring practical scalability for real-world deployment under memory and latency constraints.

FLOPs Reduction in MLLM Language Decoder Module One of the core benefits of VisionBee is its ability to reduce the input sequence length by summarizing visual tokens before they are passed into the LLM decoder, thereby leading to substantial computational savings. The decoder layers in the LLM module incur a substantial FLOPs cost, which grows with the input sequence length n. Specifically, the total FLOPs for a single decoder layer can be broken down into the following components:

$$\text{FLOPs} = 6nd_a^2 + 2n^2d_a + 2n^2d_a + 2nd_a^2 + 6nd_ad_f,$$

where d_a denotes the dimensionality of the multi-headed self-attention, d_f denotes the intermediate dimensionality of the feed-forward network (FFN), and n is the total sequence length (including both visual and textual tokens).

The above expression simplifies to:

$$\text{FLOPs} = 8nd_a^2 + 4n^2d_a + 6nd_ad_f,$$

where the first and third terms are *linear* in n, and the second term is a *quadratic* term arising from the dot-product self-attention.

Let the number of visual tokens be reduced by a summarization (or reduction) ratio r ($r > 1$) so that the new sequence length becomes $n' = n/r$ (assuming that the visual tokens dominate the input sequence fed to the LLM decoder). The FLOPs after visual tokens summarization become:

$$\text{FLOPs}_{\text{reduced}} = \frac{8n}{r}d_a^2 + \frac{4n^2}{r^2}d_a + \frac{6n}{r}d_ad_f.$$

The FLOPs reduction ratio can be written as:

$$\frac{\text{FLOPs}_{\text{reduced}}}{\text{FLOPs}_{\text{original}}} = \frac{\frac{8n}{r}d_a^2 + \frac{4n^2}{r^2}d_a + \frac{6n}{r}d_ad_f}{8nd_a^2 + 4n^2d_a + 6nd_ad_f}.$$

This shows that the **linear terms** are reduced by a factor of $1/r$, while the **quadratic term** is reduced by a factor of $1/r^2$. This quadratic scaling of attention operations makes token summarization especially effective in reducing the most expensive parts of the computation.

For instance, with $r = 9$ (i.e., reducing visual tokens by 9×), we achieve 88.89% reduction n FLOPs for linear terms such as $8nd_a^2$ and $6nd_ad_f$, and 98.77% reduction in quadratic term $4n^2d_a$.

These results highlight that VisionBee can significantly reduce compute and memory costs, particularly in the most expensive attention operations, making it highly suitable for deployment in resource-constrained environments without compromising accuracy.

Qualitative Analysis To better understand the behavior of VisionBee under small summarization budgets, we present qualitative examples of the visual tokens selected by our framework at a 9× context reduction ratio. In Fig. 7.5, we visualize the selected tokens (highlighted with red boxes) for different samples from the studied benchmarks using the LLaVA-1.5-13B model.

As seen in the examples, VisionBee effectively captures the most salient and semantically informative regions of each image despite operating with only 64 out of 576 original tokens. In natural images (top-left), VisionBee focuses on visually critical elements such as human faces, clothing, and interactive objects (e.g., the kite). In structured and cluttered scenes (bottom-left), the algorithm selectively retains object-centric patches that maximize information coverage while eliminating redundant background tokens. Furthermore, in fine-grained illustrations and document-like images (bottom-right), VisionBee preserves tokens corresponding to both visual content and textual elements, demonstrating strong multimodal understanding.

These qualitative results demonstrate that VisionBee achieves a compact yet informative token representation by preserving spatially diverse and task-relevant visual tokens. The observed diversity in selected patches stems from our submodular optimization framework, which jointly considers representativeness through key embedding similarity and importance via [CLS] attention scores. This principled selection strategy allows VisionBee to retain critical semantic information while significantly reducing the computational load for downstream processing in the LLM decoder.

Figure 7.5: Visual tokens selected by the VisionBee algorithm (shown in red boxes) using LLaVA-1.5-13B model under 9× context reduction setting.

7.8 Summary

As visual tokens dominate the input context space in MLLMs, reducing redundant visual tokens is important for improving inference efficiency while addressing associated compute and memory bottlenecks. In this chapter, we introduce **VisionBee**, a training-free visual token summarization approach that employs a principled submodular optimization framework to balance diversity and importance in token selection. By integrating key embedding similarity with [CLS] attention scores, VisionBee selectively retains the most informative and representative visual tokens. We demonstrate that VisionBee achieves significant visual token reduction while maintaining competitive performance across multiple benchmarks, outperforming prior pruning and merging strategies under identical context reduction (or summarization) ratios.

Chapter 8

DIVERSE ADVERSARIAL DATA AUGMENTATION & CONTINUAL LEARNING

8.1 Introduction

Traditional supervised machine learning methods often rely on the assumption that the data is drawn i.i.d. from a stationary probability distribution. This assumption does not hold in many practical scenarios where the learner must continuously learn online and adapt to new tasks without revisiting previous tasks (and those tasks' data). This has motivated research in Continual Learning (also referred to as Lifelong Learning and Incremental Learning), where a machine learning model learns from a stream of data coming from a succession of different tasks [154, 250, 382].

The primary challenge in continual learning (CL) is to alleviate the "catastrophic forgetting" of the previously learned tasks after learning new tasks [93, 223, 276]. This is mainly caused by a shift in the distribution of inputs and labels over time. For example, as a model is updated using new-task gradients, the hidden-layer representations, encoding information about previous tasks, become biased towards these new tasks. This leads to a model confusing the former with current tasks, thus producing incorrect predictions on older tasks. A widely studied strategy to alleviate forgetting is experience replay (ER) [45, 276, 279], which repeatedly trains the model on buffered replay data from previous tasks while learning the current task. However, the buffer in practice is often quite small [336], e.g., in autonomous driving, where the incremental data is large-scale and essentially never-ending. A model can thus overfit to such small buffered data. Although forgetting is mitigated, the goal of retaining broad and accurate knowledge about previous tasks is not achieved, and the problem remains severe.

An ideal strategy to address the above challenge is to focus on the replay of "marginal

samples" easier to forget and confuse with the current task's data, e.g., those near the boundary between the previous tasks' data and the current task's data. However, selecting such samples [7, 301] from a small buffer only brings limited improvement because the buffer (often formed using reservoir sampling [330] and thus is uniform) does not well cover many such marginal samples. Hence, we surmise that the core challenge is how to generate a small and sufficient set of samples, ones that are easier to forget and hence be confused with the current task's data—given the limited buffer size, it is also important for these samples to be diverse and thus efficient.

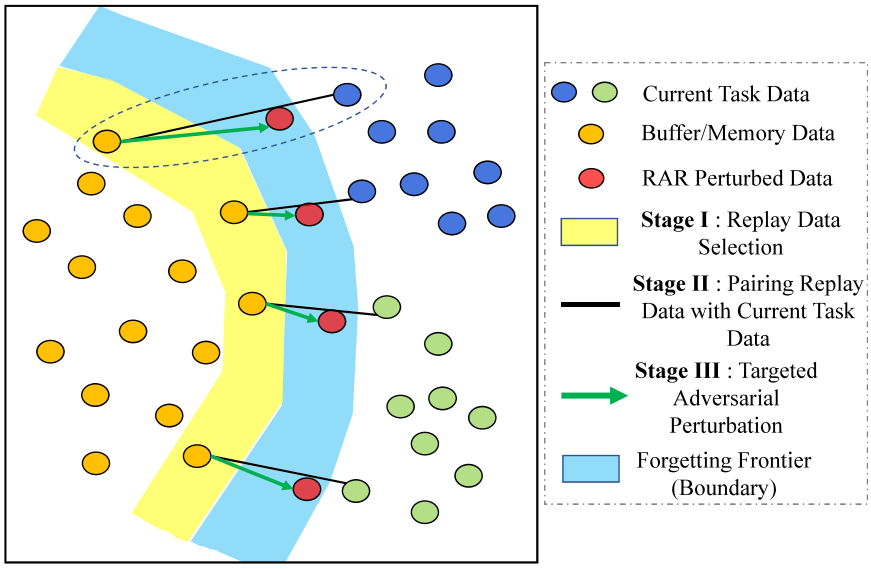


Figure 8.1: Our proposed RAR framework for generating diverse augmentations for experience replay in continual learning

In this chapter, we propose a novel targeted adversarial synthesis-based replay method, Retrospective Adversarial Replay (RAR), that can generate more informative replay instances with richer variations optimized to capture the forgetting frontier of continual learning from a limited buffer. Specifically, given an incoming batch of the current task's data, RAR first identifies the most-likely-to-be-forgotten buffer samples (Stage I in Fig. 8.1) and pairs them with their nearest neighbors within the current task's data (Stage II in Fig. 8.1).

For each pair, we then apply a bounded and targeted adversarial perturbation to the buffer sample, which moves the buffer sample's hidden-layer representation towards that of the paired current task's sample (Stage III in Fig. 8.1). This three-step procedure, as illustrated in Fig. 8.1, generates fine-grained adversarial augmentations of the buffered data that capture "forgetting" at the local boundary to the current task's data. We then effectively reduce the risk of forgetting by replaying these perturbed samples. To improve the diversity and variation of the perturbed samples (thus increasing the learnable information from a limited-size buffer), we investigate the role of MixUp [316, 385, 387], a data augmentation technique applied together with RAR—we find that it brings substantial improvements when there are strict buffer size constraints.

For Stage I replay data selections, we can utilize any existing methods [7, 8, 45, 301, 374] resulting in an effective and data-efficient replay solution in CL, hence making RAR generic and complementary to existing memory-retrieval & update methods. Extensive experiments on RAR and comparisons to SoTA baselines from different CL categories demonstrate the advantages of RAR in mitigating catastrophic forgetting and improving overall accuracy. Moreover, we conduct an ablation study showing that the key components, such as replay samples selection strategy, sample pairing & adversarial perturbation, MixUp, etc, each bring appreciable improvements. Additional sensitivity analysis of the key hyperparameters implies that RAR is stable and consistently achieves these improvements.

Connection to human false memories & their correction: In human psychology, "false memories" [208, 240] refer to a wide variety of human memory errors ranging from the minor misremembering of small details (e.g., distorted recollection) all the way to unmitigated fabrication. The passage of time after experiencing an event increases the likelihood of these phenomena. They are referred to as [240] a "natural by-product of a distributed memory system (DMS)". Rather than a simple search & recollect, DMS [55] helps us in recombining and reconstructing memories via the integration of episodes from the past and present, but it is not always error-free. It is also true that when high-confidence false memories are corrected, people attend more to the provided feedback [89]. Despite a tendency

for belief perseverance [10], the information that breaks through this perseverance is greatly attended to.

The motivation behind RAR is in line with this theory. In RAR, we first combine/pair the replay buffer samples from previously seen tasks with the current task's data. Then, using the pairing, we find existing high-confidence false memories in the form of targeted adversarial perturbations that move the previous data closer to the current data in the model's latent space. Finally, we train the model to produce the correct target for the perturbed input during the replay stage. Unlike our approach, existing ER-based CL methods lack this adversarial interpolation between past and present data and mainly rely on simple retrieval and replay of past data.

This chapter is based on the work published in: **Lilly Kumari**, Shengjie Wang, Tianyi Zhou, and Jeff A. Bilmes. *Retrospective adversarial replay for continual learning. Advances in neural information processing systems, NeurIPS, 2022.*

8.2 Related Work

Existing approaches in CL can be broadly characterized into two groups based on the model's architecture. Methods dealing with *dynamic architecture* maintain separate parameters for each task [183, 290, 375]. These methods require the knowledge of task descriptors, are expensive to train, and often do not scale well when dealing with a long sequence of online tasks. In this work, we focus on the *task-free setting of CL* [327]. *Fixed architecture* approaches use *regularization* techniques to selectively regularize parameters (either weights or outputs) important for old tasks [6, 43, 144, 169, 190, 288, 382], and require task information in general. This category also includes *memory-based* approaches for constrained parameter updates on new samples to minimize interference with previous tasks [44, 209] and for replaying them along with the new task's data [7, 8, 33, 38, 45, 70, 219, 233, 242, 247, 267, 275, 319, 301, 374]. Other ER methods [7, 281, 302, 326] train *generative models* on previous tasks which can be challenging owing to the online (single-pass) learning setting.

Algorithm 7 ER - Experience Replay based Continual Learning

Input: Tasks ($\mathcal{D}_1$, $\mathcal{D}_2$, $\ldots \mathcal{D}_T$), Model θ, Learning Rate η, Memory $\mathcal{M}$, Memory Size m, Replay Budget k
Output: Model θ
for task $i \in [T]$ **do**
  for $(X_i, Y_i) \sim \mathcal{D}_i$ **do**
    $(X'_M, Y'_M) \sim \mathcal{M}$ { // randomly select k samples from memory}
    $\ell_{ER} \leftarrow \ell(X_i, Y_i; \theta) + \ell(X'_M, Y'_M; \theta)$
    $\theta \leftarrow \text{SGD}(\ell_{ER}, \theta, \eta)$
    $\mathcal{M} \leftarrow \text{ReservoirUpdate}(X_i, Y_i, \mathcal{M}, m)$
  end for
end for

Details of Related Work In Sec. 3.2.1, we explore existing approaches for image data augmentation and their role in enhancing data variability for experience replay in continual learning, further discussed in Sec. 3.2.2. Here, we provide a comprehensive overview of prior studies relevant to task-free continual learning. However, among these studies, only GMED [130] has specifically examined the impact of data augmentations in continual learning.

- **Experience Replay (ER)** [45, 276]: ER (described in Algorithm 7) maintains previously seen examples in a limited size constrained memory buffer for future replay. It deals with the problem of memory retrieval and can be used in both task-free as well as task-dependent continual learning settings, but in all experiments in this chapter, we use it as a task-free CL method. As a new incoming batch from the current task comes in, we draw a subset of examples from the buffer such that the size of the drawn subset is the same as the incoming batch size. In all experiments, we fix this subset size (also referred to as the replay budget) as 10.

- **Gradient Episodic Memory (GEM)** [209]: GEM also maintains an episodic memory for each observed task. As a new incoming batch comes in, GEM projects the gradient of the model parameters such that loss incurred on the episodic memory maintained for each task does not increase while learning a new task. This approach, thus, needs access to the task-ids and is computationally intensive owing to solving a QP program at each iteration of training.

- **Averaged Gradient Episodic Memory (A-GEM)** [44]: Averaged-GEM is an improvement over GEM in terms of relaxing the need to require task-ids. Instead of constraining the model parameters using the entire stored buffer, it randomly samples a subset of examples from the memory buffer to regularize the model parameters in order to minimize forgetting on previously seen tasks.

- **Gradient-based Sample Selection (GSS)** [8]: GSS deals with the problem of Memory Update in a task-free CL setting. It aims to maximize sample diversity in the gradient space while selecting examples for memory update. The performance improvement observed is significant when different tasks data are imbalanced.

- **Maximally Interfered Retrieval (MIR)** [7]: MIR is another memory retrieval method that does not need access to task ids while selecting buffer exemplars for replay. It proposes to first perform a virtual update (one-step look-ahead update) only on the incoming batch of the current batch. It then computes the interference on the memory samples in terms of the loss increase observed w.r.t. the virtual updated model parameters and selects those buffer samples for which the loss increase is maximum—it selects the maximum interfered/forgotten samples from the buffer.

- **Adversarial Continual Learning (ACL)** [84]: ACL focuses on learning two kinds of features for task-dependent CL: (1) shared task-invariant (2) task-specific features. The architecture grows as a new task comes by adding a task-specific module. It uti-

lizes adversarial learning to train a shared feature encoder and a discriminator using the classic minimax optimization objective (as used by GANs). The role of the discriminator is to correctly classify the encoded features by their task labels/ids, while the role of the encoder is to generate task-invariant features that can fool the discriminator. RAR, on the other hand, focuses on the static architecture task-free CL and utilizes the idea of adversarial perturbations in the input space to perturb the memory replay samples such that they are located close to the forgetting frontier w.r.t. to the current tasks.

- **Adversarial Shapley Value Experience Replay (ASER)** [301]: ASER proposes a scoring method based on kNN Shapley Value (SV) to select memory samples that are representative of the stored memory samples and adversarially close to the decision boundaries of new classes. On the other hand, RAR's objective is to minimize the worst-case experience replay loss by generating targeted adversarial replay perturbations that are visually similar to the chosen replay samples, but the CL model mistakes them for the nearest class (from the incoming batch) based on their distance from the decision boundaries.

- **Gradient-based Memory Editing (GMED)** [130]: GMED uses gradient updates to edit/perturb the stored replay samples to create more challenging samples for rehearsal in a task-free setting. This work is most relevant to RAR. While their objective aims to perturb samples individually to generate increased losses in the upcoming model updates, our method looks into the local pairwise relationships between the most interfered memory samples and the new task's samples. By utilizing such information, we locally perturb each replay sample in a targeted adversarial manner. The model then confuses them with the new tasks' data, creating more confusing examples close to the forgetting frontier as well as representative of previous data.

8.3 Preliminaries

8.3.1 Discussion of MIR score and its intuition

Maximally Interfered Retrieval (MIR) [7] method finds the buffered data suffering from the most amount of forgetting. The idea of probing to the next-step parameters θ' that we perform in RAR is similar to the Maximally Interfered Retrieval (MIR) method for memory/buffer retrieval, where we compute forgetting scores on the memory samples to identify a subset of memory samples that are negatively impacted by the current-task samples. The forgetting score of MIR for every sample in the memory buffer x_M is defined as follows:

$$\text{MIR}(x_M, \theta, \theta') := \ell(x_M, y(x_M); \theta') - \ell(x_M, y(x_M); \theta) \tag{8.1}$$

Then, a subset of memory samples with the largest MIR scores, i.e., having the greatest loss increase from θ to θ', is selected. We will denote such a subset as X'_M, i.e.

$$X'_M \in \underset{X \subseteq X_M, |X| \leq k}{\text{argmax}} \sum_{x \in X} \text{MIR}(x, \theta, \theta'). \tag{8.2}$$

Intuitively, X'_M contains samples that are most confusing with the incoming ones, and thus using such samples as the anchor samples would result in more successful adversarial perturbations.

8.3.2 Reservoir Sampling

We use traditional reservoir sampling method [330] detailed in Algorithm 8 to update the limited memory buffer when a new batch's data comes in. Reservoir sampling selects a uniform random sample of size m from a data stream of unknown length. It maintains a reservoir of m elements. For the t-th incoming element (where $t > m$), the probability of inclusion in the reservoir is $\frac{m}{t}$. If the element is selected, it replaces an element chosen uniformly at random from the current reservoir. This process ensures that each of the first t elements has an equal probability $\frac{m}{t}$ of being retained in the reservoir at time t.

Algorithm 8 Reservoir Update

Input: Samples (x_t, y_t), Buffer $\mathcal{M}$, reservoir/buffer size m
if $|\mathcal{M}| < m$ **then**
  $\mathcal{M} = \mathcal{M} \cup (x_t, y_t)$
else
  $j = \text{randint}(1, t)$
  if $j \leq m$ **then**
    $\mathcal{M}[j] \leftarrow (x_t, y_t)$
  end if
end if

Prior works based on task-free continual learning [7, 170, 209] have shown promising results when using reservoir sampling as the memory update method. Authors in [141, 209] have proposed modified versions of reservoir sampling, which help populate the memory with diverse samples in a balanced way when dealing with heterogeneous and long-tailed data streams. Since, in this work, our focus is primarily on class-balanced datasets, we use Reservoir Sampling as the memory update method.

8.3.3 Targeted Adversarial Augmentations

Given a model θ, associated layer l of the model, source input-output pair (x_s, y_s) and a target input-output pair (x_t, y_t), the goal of targeted adversarial attack/perturbation is to generate an input z which looks visually similar to the source input x_s, but in the latent space of model's l-th layer, the feature embedding of z is close to the feature embedding of the target input x_t. The objective can be mathematically summarized as follows:

$$\begin{aligned} &\text{minimize} \quad \mathcal{L}_{\theta_l}(z, x_t) \\ &\text{subject to } ||z - x_s||_\infty \leq \epsilon \end{aligned} \tag{8.3}$$

Here, if we use the model's last layer denoted by L, then oftentimes $\mathcal{L}_{\theta_l}(z, x_t)$ can be

replaced with the cross-entropy loss w.r.t. to the target label y_t of the paired sample x_t, i.e., $\ell(z, y_t; \theta)$.

In case we use layer l such that $l < L$, meaning layers lower/shallower than the last layer, one can use the defined distance function in Eq. (8.11) i.e., $d_{\theta_l}(z, x_t) := ||\theta_l(z) - \theta_l(x_t)||_2$ as the minimization objective $\mathcal{L}_{\theta_l}(z, x_t)$ in the above constrained optimization problem.

The Fast Gradient Sign Method (FGSM) proposed in [105] can be used to approximately solve Eq. (8.3) when dealing with infinity norm constraints in the input space. We define the approximate solution $g_{\theta_l}(x_s, x_t)$ as:

$$g_{\theta_l}(x_s, x_t) = x_s - \epsilon \cdot \text{sign}(\nabla_{x_s} \mathcal{L}_{\theta_l}(x_s, x_t)) \tag{8.4}$$

where $\nabla_{x_s} \mathcal{L}_{\theta_l}(x_s, x_t)$ is the gradient of the loss function in Eq. (8.3) w.r.t. x_s.

Iterative-FGSM [161], which performs n iterative updates based on the FGSM method using a smaller step size α (proportional to ϵ/n), can also be used. Here, n denotes the number of iterations used to iteratively optimize Eq. (8.3).

To stabilize the gradient update directions and avoid poor local maxima, Dong et al. [80] proposed Momentum-Iterative FGSM (MI-FGSM), which integrates momentum with a decay factor μ and accumulates the gradients of previous iterations while updating the targeted augmentation at each step.

At step $\tau = 0$, we initialize $x_s^0 = x_s$. Then, at step τ, the accumulated gradient and targeted augmentation are derived as follows (the superscript τ denotes the iteration step, while the subscript t denotes the target example):

$$g^{\tau+1} = \mu \cdot g^{\tau} + \frac{\nabla_{x_s} \mathcal{L}_{\theta_l}(x_s^{\tau}, x_t)}{||\nabla_{x_s} \mathcal{L}_{\theta_l}(x_s^{\tau}, x_t)||_1} \tag{8.5}$$

$$x_s^{\tau+1} = x_s^{\tau} - \alpha \cdot \text{sign}(g^{\tau+1}) \tag{8.6}$$

At the end of each step, we project back into the norm ball $||x_s^{\tau+1} - x_s||_\infty \leq \epsilon$, and we also clip the values back to the input range observed corresponding to the original pixel values when using image data. In this work, we use MI-FGSM with the decay factor μ as 1.0, and the step-size α same as ϵ since we employ only two iterations to generate the RAR

augmented data using buffer replay samples (X'_M, Y'_M) as the source/anchor sample. The paired sample from the incoming batch of current task (X_i, Y_i) is used as the target sample.

8.4 Retrospective Adversarial Replay

The continual learning problem consists of a stream of data distributions $(\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_T)$ that corresponds to T tasks (in this work, the boundaries between tasks are not known). We get data points uniformly sampled from the data distribution of the current task. Our target is to train a machine learning model $f(\cdot; \theta)$ parameterized by θ that minimizes the losses as we learn each task in the given sequential order without increasing the loss on previously learned samples. In other words, the objective is

$$\min_{\theta} \sum_{i=1:T} \mathbb{E}_{(x,y)\sim\mathcal{D}_i} \ell(x, y; \theta) \tag{8.7}$$

Here, ℓ denotes the loss function, and i is an index over the set of tasks. To retain the knowledge of previous tasks and mitigate catastrophic forgetting, similar to [7, 301, 374], we utilize a limited memory buffer to store samples representing previously seen data. Let $\mathcal{M}$ denote the memory/buffer, which is essentially a set of data points and $|\mathcal{M}| \leq m$. Similar to ER (Algorithm 7) , as we train on task i, we get an incoming batch $(X_i, Y_i) \sim D_i$, and we combine it with a subset of memory samples $(X'_M, Y'_M) \subset \mathcal{M}$ to compute the overall loss. We can then update the memory $\mathcal{M}$ by replacing some samples with the ones encountered in the current training step while keeping the size of the buffer unchanged.

In general, ER-based methods typically deal with two problems: (1) *Memory Update* for populating the buffer and (2) *Memory Retrieval* for selecting samples from the buffer for replay. In this work, we focus on synthesizing adversarial memory samples that provide the most conflicting information on the look-ahead probing step. In other words, the model trained only on the incoming task (one-step look-ahead) misclassifies the synthesized samples (anchored around previously seen classes stored in the memory) as the incoming classes. Thus, by learning from such adversarially synthesized memory samples in retrospect, we prevent any conflicts in the probing step. For problem (1), we use reservoir sampling to update

the memory (Sec. 8.3.2) , which can give us uniform samples from the streaming data points and is widely adopted in many continual learning methods [7, 45, 301]. For problem (2), one can utilize any existing memory retrieval methods [7, 8, 45, 301, 374], thus making RAR complimentary and easily integratable with many existing CL methods.

8.4.1 Retrospective Adversarial Perturbation

Existing methods addressing the memory-retrieval problem optimize the model parameters using ℓ_{ER} in Eq. (8.8) where (X'_M, Y'_M) are selected from the buffer $\mathcal{M}$ based on their proposed selection strategies. For ease of notation, we use $y(x)$ to denote the ground-truth label of the sample x.

$$\ell_{ER}(X'_M, X_i, \theta) := \ell(X_i, y(X_i); \theta) + \ell(X'_M, y(X'_M); \theta) \tag{8.8}$$

ER [45] uses naive random sampling to select X'_M, GSS [8] encourages gradient diversity while sampling X'_M, ASER [301] uses kNN Shapley value to select samples which are representative of $\mathcal{M}$ and close to the samples in X_i in the latent space, and MIR [7] selects replay samples which are maximally interfered/forgotten after the look-ahead probing step. In the look-ahead probing step, the model is updated only on the incoming task samples X_i such that $\theta' = \theta - \eta \nabla_\theta \ell(X_i, y(X_i); \theta)$.

Although the methods mentioned above cover crucial aspects (diversity, representativeness, similarity to incoming batch) for selecting X'_M, they do not capture the local pairwise interactions between the memory and current-task samples. Motivated by this, Retrospective Adversarial Replay Loss (ℓ_{RAR}) finds a cardinality-constrained pairing (E) between memory and incoming current-task samples such that the loss on memory samples that are perturbed towards the paired current-task samples is maximized:

$$\ell_{RAR}(X_M \times X_i, \theta, \theta') := \max_{\substack{E \subset X_M \times X_i \\ |E| = c}} \sum_{(x_M, x_i) \in E} \ell(g_{\theta'_l}(x_M, x_i), y(x_M); \theta) \tag{8.9}$$

Here, X_M denotes the entire input samples present in $\mathcal{M}$, and $g_{\theta'_l}(x_M, x_i)$ described mathematically in Eq. (8.10) can be seen as a targeted perturbation of x_M that is visually similar

to x_M but close to the latent embedding of target $x_i \in X_i$ in the l-th layer latent space modeled by θ'. Since the minimization objective in Eq. (8.10) aims to move the perturbed replay sample closer to a current-task sample, it is required that $y(x_M) \neq y(x_i)$ for a given pairing. This helps filter out trivial pairings while optimizing Eq. (8.9).

$$g_{\theta'_l}(x_M, x_i) = \underset{z}{\operatorname{argmin}}\, d_{\theta'_l}(z, x_i) \;\text{ s.t. }\; ||z - x_M||_\infty \leq \epsilon \tag{8.10}$$

$$\text{where } d_{\theta'_l}(z, x_i) := ||\theta'_l(z) - \theta'_l(x_i)||_2 \tag{8.11}$$

Since x_M and x_i have different labels, such perturbed samples that we get from Eq. (8.10) are confusing to the continual learner in terms of distinguishability from the current-task sample x_i while still representing previous tasks based on their visual similarity to $x_M \in \mathcal{M}$. Eq. (8.9) essentially helps in the synthesis of targeted adversarial samples that are close to incoming samples X_i in the latent space, and hence the learner will miss making the correct prediction $y(X_M)$ despite the fact that the adversarial samples are anchored around the original memory samples X_M. Training on such samples can help the model to learn about the boundaries between previous and current tasks and also retain the knowledge of previous tasks [39, 52, 116]. Note that the perturbations are performed based on the look-ahead parameters θ'. In other words, assuming we only learn from the current-task samples X_i and update the parameters to θ', the generated adversarial samples give us many training samples that show conflicting behaviors on θ' and get easily forgotten by the model. In retrospect, we can potentially prevent such forgetting when we are still at θ by using those targeted adversarial samples as training signals, i.e., we minimize the RAR loss (Eq. 8.9) to update the model θ.

Optimizing Eq. (8.9) over all feasible pairs of X_M and X_i is computationally challenging as we effectively need to compute the adversarial perturbations on all such pairs to select the most confusing subset incurring the maximum RAR loss. Moreover, each adversarial sample synthesis (Eq. 8.10) requires multiple gradient steps as we need to run adversarial attack algorithms such as the iterative-FGSM [80, 105] covered in detail in Sec. 8.3.3. Thus, to eliminate this bottleneck, we propose two pruning strategies covered in Sec. 8.4.2 and 8.4.3

to reduce the candidate anchor-target pairs in $X_M \times X_i$ for adversarial perturbations and approximately optimize Eq. (8.9).

8.4.2 Selecting Replay Data from Buffer

To reduce the computational overhead, we only consider a subset of the memory buffer $X'_M \subset \mathcal{M}$ instead of considering all memory input samples X_M. Memory-retrieval methods such as ER [45], GSS [8], ASER [301], and MIR [7] can be used to select a smaller replay set X'_M. This again reiterates that RAR is generic and complementary to existing memory-retrieval methods.

8.4.3 Pairing between Buffer and Current-task Data

To further bring down the computational costs of adversarial perturbations, we perform another pruning over the possible pairs between $X'_M \times X_i$ based on samples' hidden representations. I.e., we first compute the distance $d_{\theta'_l}(x'_M, x_i)$, which is essentially the target value that we seek to minimize for the adversarial perturbation (Eq. 8.10) for every pair $(x'_M, x_i) \in X'_M \times X_i$ with different labels $y(x'_M) \neq y(x_i)$ and then screen out the pairs with larger distance values, maintaining only the smaller-distance ones, in accordance with the min-objective of Eq. (8.10). To utilize the information from the selected memory samples, we pair each $x'_M \in X'_M$ with its nearest neighbor with a different class/label in the incoming batch X_i of the current task. Nearest-neighbor-based matched pairs already represent the most confusing samples, as the samples in a pair belong to different classes. Thus, it is easier for the optimization procedure approximating Eq. (8.10) to further reduce the distances between pairs coming from different classes (shown in Fig. 8.2b), aiding in the synthesis of even more confusing adversarial samples.

Computational overhead for optimizing Eq. (8.9): First, we select memory samples for replay X'_M (Sec. 8.4.2), which has the same costs as the backbone memory-retrieval method. Next, we compute the distances between latent features of X'_M and incoming batch X_i with computational complexity of $\mathcal{O}(|X'_M| * |X_i|)$. This is minimal given the online nature

of CL, where the training batch size is often set to 10. After performing the nearest neighbor matching, we optimize Eq. (8.10) via iterative-FGSM [80], which introduces some additional computational costs, but we keep it marginal by using only two steps of iterative-FGSM. We provide further details in Sec. 8.4.6.

Combining our RAR loss with the loss on selected memory samples and incoming batch samples, we have our overall objective (similar to Eq. 8.8) defined as follows, where β is a hyperparameter to control the trade-offs between the losses on original and adversarially synthesized memory samples.

$$\ell_{all}(X'_M, X_i, \theta) := \ell(X_i, y(X_i); \theta) + \beta \ell_{RAR}(X'_M \times X_i, \theta, \theta') + (1-\beta)\ell(X'_M, y(X'_M); \theta) \quad (8.12)$$

Description of RAR framework: In Algorithm 9, we describe our retrospective adversarial replay algorithm for continual learning. The algorithm essentially consists of three steps: (1) in Lines 5-6, we first perform the look-ahead parameter probing using the incoming samples of the current task to get θ' and select the subset X'_M using existing memory retrieval methods, such as ER, MIR, ASER, etc. (2) in Lines 7-11, we prune down the pairings between X'_M and X_i based on distances between hidden representations computed using Eq. (8.11), and (3) in Line 12, we define the RAR loss by performing adversarial perturbations on the selected pairs S. Note that since we use the selected pairings S of size k as an approximation of the true maximum-valued pairs, we can just sum over entries in S instead of picking the maximum subset as done in Eq. (8.9). In Line 15, we update the memory buffer $\mathcal{M}$ based on incoming samples (X_i, Y_i) using reservoir sampling [330] and keep the memory limit m.

8.4.4 MixUp based Retrospective Adversarial Augmentation

We also propose another method *mix-RAR* utilizing the *mixup* technique [385] to bring in more diversity to the pool of replay examples selected from the buffer. After creating virtual examples $(\tilde{x}_M, \tilde{y}_M)$ using $x_i, x_j \in X'_M$ and $\lambda \in [0, 1]$ such that $\tilde{x}_M = \lambda x_i + (1-\lambda)x_j$ and $\tilde{y}_M = \lambda y_i + (1-\lambda)y_j$, we follow the original RAR algorithm to achieve pairing and subsequent

Algorithm 9 RAR - Retrospective Adversarial Replay for Continual Learning

Input: Tasks ($\mathcal{D}_1$, $\mathcal{D}_2$, $\ldots \mathcal{D}_T$), Model θ, Learning Rate η, Memory $\mathcal{M}$, Memory Size m, Replay Budget k, Trade-off Parameter β, Perturbation Strength ϵ & number of gradient steps n

Output: Model θ

for task $i \in [T]$ **do**

  for $(X_i, Y_i) \sim \mathcal{D}_i$ **do**

    $\theta^{'} \leftarrow \text{SGD}(\ell(X_i, Y_i; \theta), \theta, \eta)$ { // Look-ahead update of model parameters}

    Select (X'_M, Y'_M) from $\mathcal{M}$ s.t. $|X'_M| = k$ { // using existing memory-retrieval methods}

    Let $S := \emptyset$

    for $x'_M \in X'_M$ **do**

      $x'_i \in \text{argmin}_{x_i \in X_i, y(x'_M) \neq y(x_i)} \, d_{\theta'_l}(x'_M, x_i)$

      $S \leftarrow S \cup \{(x'_M, x'_i)\}$

    end for

    $\ell_{RAR}(S, \theta, \theta') \leftarrow \sum_{(x'_M, x_i) \in S} \ell(g_{\theta'_l}(x'_M, x_i), y(x'_M); \theta)$ { // perform the targeted adversarial perturbations (Sec. 8.3.3) and define the corresponding RAR loss}

    $\ell_{all} \leftarrow \ell(X_i, Y_i; \theta) + \beta \, \ell_{RAR}(S, \theta, \theta') + (1 - \beta) \, \ell(X'_M, Y'_M; \theta)$

    $\theta \leftarrow \text{SGD}(\ell_{all}, \theta, \eta)$

    $\mathcal{M} \leftarrow \text{ReservoirUpdate}(X_i, Y_i, \mathcal{M}, m)$

  end for

end for

targeted adversarial perturbation as shown in Algorithm 12. The proposed MixUp strategy is different from [112, 227] as we apply mixup among the replay samples and then generate RAR perturbed samples anchored around them. This is consistent with RAR's objective. Given a current task's sample from class (c), training on a RAR perturbation applied to a mixup between two buffered samples from different past classes (a,b) can potentially help in capturing the forgetting frontier between three pairs of classes (a,c), (b,c) and (a,b), thus minimizing forgetting on both past classes.

8.4.5 RAR variants

ER-RAR In the ER-RAR method, we simply modify the replay sample selection step (line 6) of Algorithm 9. As shown in line 6 of Algorithm 10, we rely on random sampling to get a subset of the buffer for replay.

MIR-RAR In Algorithm 11, we change line 6 of Algorithm 9 to use the MIR score to select the replay subset X'_M.

mix-RAR In the mix-RAR method detailed in Algorithm 12, after selecting the replay samples X'_M (using ER, MIR, ASER, etc.) from memory $\mathcal{M}$, we simply apply MixUp [385] to perform perturbations/editing on the replay examples. Mixup generates virtual training samples $\tilde{x}_M, \tilde{y}_M$ by taking convex combinations of pairs of inputs and their labels.

$$\tilde{x}_M = \lambda x_i + (1-\lambda)x_j \qquad \tilde{y}_M = \lambda y_i + (1-\lambda)y_j \tag{8.13}$$

where $x_i, x_j \in X'_M$ and $y_i, y_j \in Y'_M$ and $\lambda \in [0, 1]$ is drawn from a Beta distribution $\lambda \sim \text{Beta}(\alpha, \alpha)$, where α is a hyperparameter.

We then pair up these virtual samples with the current's task samples based on the distance function defined in Eq. (8.11) and use Eq. (8.10) to perturb them such that in the representation space modeled by θ'_l, the augmentation of the virtual sample is closer to the paired sample from the current task, than the source/anchor samples x_i and x_j.

Algorithm 10 ER-RAR - Retrospective Adversarial Replay for Continual Learning using vanilla ER as memory-retrieval method

Input: Tasks ($\mathcal{D}_1$, $\mathcal{D}_2$, ... $\mathcal{D}_T$), Model θ, Learning Rate η, Memory $\mathcal{M}$, Memory Size m, Replay Budget k, Trade-off Parameter β, Perturbation Strength ϵ & number of gradient steps n

Output: Model θ

for task $i \in [T]$ **do**

  for $(X_i, Y_i) \sim \mathcal{D}_i$ **do**

    $\theta' \leftarrow \text{SGD}(\ell(X_i, Y_i; \theta), \theta, \eta)$ { // Look-ahead update of model parameters}

    $(X'_M, Y'_M) \sim \mathcal{M}$ { // randomly select k samples from memory}

    Let $S := \emptyset$

    for $x'_M \in X'_M$ **do**

      $x'_i \in \text{argmin}_{x_i \in X_i, y(x'_M) \neq y(x_i)} \, d_{\theta'_l}(x'_M, x_i)$.

      $S \leftarrow S \cup \{(x'_M, x'_i)\}$.

    end for

    $\ell_{RAR}(S, \theta, \theta') \leftarrow \sum_{(x'_M, x_i) \in S} \ell(g_{\theta'_l}(x'_M, x_i), y(x'_M); \theta)$ { // perform the targeted adversarial perturbations (Sec. 8.3.3) and define the corresponding loss}

    $\ell_{all} \leftarrow \ell(X_i, Y_i; \theta) + \beta \, \ell_{RAR}(S, \theta, \theta') + (1 - \beta) \, \ell(X'_M, Y'_M; \theta)$

    $\theta \leftarrow \text{SGD}(\ell_{all}, \theta, \eta)$

    $\mathcal{M} \leftarrow \text{ReservoirUpdate}(X_i, Y_i, \mathcal{M}, m)$

  end for

end for

Algorithm 11 MIR-RAR - Retrospective Adversarial Replay for Continual Learning using MIR as memory-retrieval method

Input: Tasks ($\mathcal{D}_1$, $\mathcal{D}_2$, $\ldots \mathcal{D}_T$), Model θ, Learning Rate η, Memory $\mathcal{M}$, Memory Size m, Replay Budget k, Subset Size c, Trade-off Parameter β, Perturbation Strength ϵ & number of gradient steps n
Output: Model θ
for task $i \in [T]$ **do**
  for $(X_i, Y_i) \sim \mathcal{D}_i$ **do**
    $\theta' \leftarrow \text{SGD}(\ell(X_i, Y_i; \theta), \theta, \eta)$ { // Look-ahead update of model parameters}
    $(X_M, Y_M) \sim \mathcal{M}$ { // randomly select c samples from memory}
    Compute $\text{MIR}(x_M, \theta, \theta')$ for $x_M \in X_M$ and select the subset X'_M based on Eq. (8.2) s.t. $|X'_M| = k$
    Let $S := \emptyset$.
    for $x'_M \in X'_M$ **do**
      $x'_i \in \text{argmin}_{x_i \in X_i, y(x'_M) \neq y(x_i)} d_{\theta'_l}(x'_M, x_i)$.
      $S \leftarrow S \cup \{(x'_M, x'_i)\}$.
    end for
    $\ell_{RAR}(S, \theta, \theta') \leftarrow \sum_{(x'_M, x_i) \in S} \ell(g_{\theta'_l}(x'_M, x_i), y(x'_M); \theta)$ { // perform the targeted adversarial perturbations (Sec. 8.3.3) and define the corresponding loss}
    $\ell_{all} \leftarrow \ell(X_i, Y_i; \theta) + \beta\, \ell_{RAR}(S, \theta, \theta') + (1 - \beta)\, \ell(X'_M, Y'_M; \theta)$
    $\theta \leftarrow \text{SGD}(\ell_{all}, \theta, \eta)$
    $\mathcal{M} \leftarrow \text{ReservoirUpdate}(X_i, Y_i, \mathcal{M}, m)$
  end for
end for

Algorithm 12 mix-RAR - Retrospective Adversarial Replay for Continual Learning using MixUp

Input: Tasks ($\mathcal{D}_1$, $\mathcal{D}_2$, $\ldots \mathcal{D}_T$), Model θ, Learning Rate η, Memory $\mathcal{M}$, Memory Size m, Replay Budget k, Trade-off Parameter β, Perturbation Strength ϵ & number of gradient steps n
Output: Model θ
for task $i \in [T]$ **do**
  for $(X_i, Y_i) \sim \mathcal{D}_i$ **do**
    $\theta' \leftarrow \text{SGD}(\ell(X_i, Y_i; \theta), \theta, \eta)$ { // Look-ahead update of model parameters}
    Select (X'_M, Y'_M) from $\mathcal{M}$ s.t. $|X'_M| = k$ { // using existing memory-retrieval methods}
    $\tilde{X}'_M, \tilde{Y}'_M = \text{mixup}(X'_M, Y'_M, \alpha)$ { // perform mixup on replay samples}
    Let $S := \emptyset$
    for $\tilde{x}_M \in \tilde{X}'_M$ **do**
      $x'_i \in \text{argmin}_{x_i \in X_i, y(\tilde{x}_M) \neq y(x_i)} \, d_{\theta'_l}(\tilde{x}_M, x_i)$.
      $S \leftarrow S \cup \{(\tilde{x}_M, x'_i)\}$.
    end for
    $\ell_{RAR}(S, \theta, \theta') \leftarrow \sum_{(\tilde{x}_M, x_i) \in S} \ell(g_{\theta'_l}(\tilde{x}_M, x_i), y(\tilde{x}_M); \theta)$ { // perform the adversarial perturbations (Sec. 8.3.3) and define the corresponding loss}
    $\ell_{all} \leftarrow \ell(X_i, Y_i; \theta) + \beta \, \ell_{RAR}(S, \theta, \theta') + (1 - \beta) \, \ell(X'_M, Y'_M; \theta)$
    $\theta \leftarrow \text{SGD}(\ell_{all}, \theta, \eta)$
    $\mathcal{M} \leftarrow \text{ReservoirUpdate}(X_i, Y_i, \mathcal{M}, m)$
  end for
end for

8.4.6 Complexity Analysis

We compare the replay-based methods in terms of forward and backward computations used, besides the forward features computed on the incoming batch X_i and backward pass used for the final parameters updates:

1. **ER** (Alg. 7): 1 extra forward pass on the replay samples, i.e., $f(X'_M; \theta)$
2. **ER-RAR** (Alg. 9): 1 backward pass for the virtual model update to θ', n forward and backward passes to generate the targeted augmentations of X'_M using θ' where n denotes the number of iterative steps used to generate the RAR data, 1 forward pass on the replay examples and the generated RAR data using θ.
3. **MIR**: 1 backward pass for the virtual model update to θ', 2 forward passes on the larger memory set, i.e., $f(X_M; \theta), f(X_M; \theta')$, 1 forward pass on the smaller set of replay samples, i.e., $f(X'_M; \theta)$.
4. **MIR-RAR** (Alg. 11): 1 backward pass for the virtual model update to θ', 2 forward passes on the larger memory set, i.e. $f(X_M; \theta), f(X_M; \theta')$, n forward and backward passes to generate the targeted augmentations of X'_M using θ' where n denotes the number of iterative steps used to generate the RAR data, 1 forward pass on the smaller set of original replay samples and the RAR augmented data using θ.

Thus, comparing ER-RAR w.r.t ER, ER-RAR needs n extra forward passes and $(n+1)$ extra backward passes on top of ER. Similarly, MIR-RAR adds n additional forward and backward passes on top of MIR. Setting $n = 2$ for all datasets (except Split-MNIST with 1000 training samples per task where $n = 5$), i.e., using only two gradient steps to optimize the RAR augmentation objective defined in Eq. (8.10) in the input space, we limit the additional computational costs introduced by RAR.

Computational costs reduction based on optimizations proposed in Sec. 8.4.2 and 8.4.3: Instead of using all $|X_M|$ examples in the memory buffer $\mathcal{M}$, replay data selection reduces the potential candidates to be considered for adversarial data generation to $|X'_M|$. For example, given a buffer size of 500 and a replay budget of $k = 10$, the potential buffer

samples considered for generating RAR adversarial augmentations is only 10 instead of 500.

Having reduced the size of the replay set to $|X'_M|$, one would still need to consider every potential pairing between X'_M and X_i (current task samples) to find the most confusing subset of cardinality k, which incurs maximum RAR loss defined in Eq. (8.9). Here, k is essentially the replay budget. Generating targeted adversarial samples for each pair can be computationally prohibitive. Our nearest neighbor-based pairing brings down the number of adversarial samples to be generated (by optimizing Eq. 8.9) to k from $|X'_M| \times |X_i|$. In the above example, for a current batch size of 10, we only need to generate $k = 10$ targeted adversarial samples to approximately optimize Eq. (8.9) instead of generating 10×10 adversarial samples and then choosing k samples with the largest RAR loss.

8.5 Experiments

Datasets We evaluate RAR on four supervised image classification benchmarks for task-free CL. Based on the taxonomy described in [327], the datasets that we studied fall under the *class-incremental* (CI) category. The dataset details are as follows:

Split-MNIST is a variant of the MNIST dataset of handwritten digits [167] split into five disjoint tasks based on the labels. Each task consists of two classes, and we use only 1k examples from each task for training and report results on the complete test set. The goal is to classify all ten classes at the end of the last task.

Split-CIFAR10 is a variant of the CIFAR-10 dataset [152] split into 5 disjoint tasks based on labels. Each task consists of two classes and 10k training examples.

Split-CIFAR100 is a variant of the CIFAR-100 dataset comprised of 20 disjoint tasks, with each task dealing with 5 classes. This is a 100-way classification problem since we do not use any task information during training/testing.

Split-miniImageNet splits the mini-ImageNet dataset [71] into 20 disjoint tasks based on labels. Each task comprises five classes and 2.5k training examples.

In Table 8.1, we present the statistics of all four datasets. Training samples used for each task are 95% of the mentioned numbers as 5% is held out as the validation set. For

fine-tuning the hyperparameters associated with the RAR framework, we use the held-out validation set. In the case of the Split-MNIST dataset, following the setting of previous methods such as [7, 130], we use only 1000 training examples per task.

Table 8.1: Class-incremental Continual Learning Datasets Statistics

	Split-MNIST	**Split-CIFAR10**	**Split-CIFAR100**	**Split-miniImageNet**
# of tasks	5	5	20	20
# of classes per task	2	2	5	5
# of training samples per task	1000	10000	2500	2500
# of test samples per task	2000	2000	500	500
# of overall classes	10	10	100	100

Evaluation Metrics Following [43, 209], we use two standard metrics used in CL to evaluate RAR's performance. *Final average accuracy (A_T)* assesses the overall performance of the model whereas *Final average forgetting (F_T)* measures how much the model has forgotten each task once the online learning has completed. Lower forgetting is better. Below, $a_{i,j}$ denotes the accuracy evaluated on the test set of task j after the model has experienced all tasks up to i.

$$A_T = \frac{1}{T}\sum_{j=1}^{T} a_{T,j}, \qquad F_T = \frac{1}{T-1}\sum_{j=1}^{T-1} \max_{l\in\{1,\ldots T-1\}} (a_{l,j} - a_{T,j}) \tag{8.14}$$

Models and Settings We follow the same setup as [7] for deciding model architectures for all four datasets. For Split-MNIST, we use an MLP classifier with two hidden layers, each with 400 units with ReLU activation, followed by a linear classifier layer with 10 units. For Split CIFAR-10, Split CIFAR-100, and Split mini-ImageNet, we use a reduced ResNet-18 classifier [114]. The replay budget k is the same as the mini-batch size (fixed to 10) irrespective of the buffer size m. For hyperparameters tuning on each dataset, we hold out 5% of the training samples for each task and use it as a validation set.

In all experiments based on experience replay methods, we use SGD optimizer with a learning rate of 0.1 for all datasets. For GMED [130] and ASER [301] based methods, we present the results using the set of hyperparameters reported in the papers. Both mini-batch size and replay budget are set to 10 following the above-mentioned methods to make the comparisons fair. For MIR-based methods, we set the subset size c to 50 for all datasets following [7]. c denotes the size of the larger set of buffer examples on which the forgetting/interference score is computed.

Hyperparameters Tuning We tune the hyperparameters associated with the RAR framework $\{n,\ \beta,\ \epsilon\}$ on the held-out validation set across all datasets. For all datasets, we analyze (as shown in Fig. 8.5) the sensitivity of the performance of RAR in terms of the number of gradient steps n used to perform the constrained targeted optimization in the input space, as shown in Eq. (8.10). Specifically, we study values of $n \in \{2, 5, 10\}$. Based on the trade-off between the performance and the computational costs associated with the forward and backward passes employed to generate the targeted adversarial augmentation, we set $n = 2$ for all datasets except for Split-MNIST, where we set $n = 5$ based on the average forgetting score as shown in Fig. 8.5a. Since the MNIST dataset is relatively easier to train on, more gradient steps are needed to generate difficult samples that are indiscernible for the model in the learned representation space after the look-ahead update on the new task's data.

For the weight assigned to the ℓ_{RAR} computed using Eq. (8.9) in the final loss objective (denoted by β), we search over $\{0.05, 0.075, 0.1, 0.2, 0.3, 0.4, 0.5\}$ and fine-tune them separately for different datasets across various buffer sizes that we study. For Split-MNIST, β equal to $0.3, 0.3$, and 0.4 lead to the best-reported performance across buffer sizes $200, 500$, and 1000, respectively. For Split-CIFAR10, these values are $0.5, 0.1$, and 0.075 for buffer sizes $200, 500$, and 1000 respectively. For Split-CIFAR100 and Split-miniImageNet, we get the best setting as $\beta = 0.4$.

We search for the best perturbation tolerance ϵ hyperparameter in $\{0.15, 0.3\}$ for Split-

MNIST, $\{0.0314, 0.07\}$ for Split-CIFAR10 and Split-CIFAR100, and $\{0.01568, 0.0314\}$ for Split-miniImageNet. The best setting ϵ values are 0.3 for Split-MNIST and 0.0314 for the remaining three datasets.

Prior works such as [265] show that the deeper/higher layers of neural networks are disproportionately responsible for catastrophic forgetting in line with the observation that the shallower/lower layers learn general features of and higher layers learn more task-specific features. Therefore, we use the penultimate layer (the layer just before the final classification/softmax layer) as layer l in Eq. (8.10) and in the representation space of θ'_l, we minimize the augmentation objective such that the penultimate feature embedding of the RAR augmented sample and its paired current task sample are as close as possible, but visually still similar to the original buffer sample.

8.5.1 Results

We report the average accuracy and forgetting in Table 8.2 and 8.3, respectively, for experiments on all four datasets under various buffer size constraints. We use three different memory-retrieval methods, namely *ER* [45], *MIR* [7], and *ASER* [301] and compare *RAR* and *mix-RAR* with their respective baselines which do not use the RAR loss (Eq. 8.9). For a fair comparison, we compare *mix-RAR* with baseline retrieval methods using mixup between replay samples. We also compare against standard baseline methods: (a) *Fine-tuning*, which trains the model continuously on the stream of tasks without employing any strategy to avoid forgetting, (b) *Averaged Gradient Episodic Memory (AGEM)* [44] (c) *iid-online*, which trains a model on the iid-data sampled from entire dataset $\mathcal{D} = \cup_{i=1}^{T} \mathcal{D}_i$ for a single epoch, (d) *iid-offline*, which trains a model similar to the iid-online setting but for multiple epochs (three epochs considered in the reported results). We compare against another memory sample perturbation method called *GMED* [130] in Table 8.5 to avoid repetition.

We provide the pseudo-code for ER (Alg. 7), ER-RAR (Alg. 10), MIR-RAR (Alg. 11), and mix-RAR (Alg. 12) to describe how our proposed framework can be unified with existing CL methods. As shown in Table 8.2, both *RAR* and *mix-RAR* using different memory-

Table 8.2: Average Accuracy (↑) on different task-free CL datasets across different buffer sizes (m). We averaged results of over 15 runs for all except for Split-MNIST, where we averaged over 20 runs.

	Split-MNIST		**Split-CIFAR10**			**Split-CIFAR100**	**Split-miniImageNet**
Method	m=500	m=1000	m=200	m=500	m=1000	m=10,000	m=10,000
Fine-tuning	19.0 ± 0.2	19.0 ± 0.2	18.4 ± 0.3	18.4 ± 0.3	18.4 ± 0.3	3.06 ± 0.2	2.84 ± 0.4
AGEM	29.02 ± 5.3	-	22.7 ± 1.8	22.7 ± 1.9	22.6 ± 0.7	2.40 ± 0.2	2.92 ± 0.3
ER	80.7 ± 2.1	83.3 ± 1.4	25.7 ± 1.3	31.9 ± 0.9	39.5 ± 1.7	26.0 ± 0.3	23.7 ± 0.5
ER-RAR (ours)	$\mathbf{84.2 \pm 1.8}$	$\mathbf{86.9 \pm 1.1}$	$\mathbf{33.2 \pm 1.4}$	$\mathbf{40.4 \pm 1.7}$	$\mathbf{44.0 \pm 1.4}$	$\mathbf{29.8 \pm 0.4}$	$\mathbf{27.9 \pm 0.5}$
ER-mix	80.9 ± 1.3	81.7 ± 1.1	24.9 ± 1.8	33.0 ± 2.3	37.5 ± 1.6	26.6 ± 0.8	23.0 ± 1.1
ER-mix-RAR (ours)	$\mathbf{86.1 \pm 1.1}$	$\mathbf{86.9 \pm 1.0}$	$\mathbf{37.2 \pm 1.0}$	$\mathbf{41.6 \pm 1.0}$	$\mathbf{44.4 \pm 0.8}$	$\mathbf{31.5 \pm 0.6}$	$\mathbf{28.8 \pm 0.5}$
MIR	84.8 ± 1.2	86.3 ± 1.4	28.0 ± 1.5	37.5 ± 1.6	45.3 ± 1.2	26.7 ± 0.4	24.3 ± 0.6
MIR-RAR (ours)	$\mathbf{87.9 \pm 1.4}$	$\mathbf{89.4 \pm 1.1}$	$\mathbf{33.3 \pm 1.1}$	$\mathbf{43.1 \pm 1.1}$	$\mathbf{45.3 \pm 1.5}$	$\mathbf{29.3 \pm 0.4}$	$\mathbf{26.7 \pm 0.7}$
MIR-mix	85.7 ± 0.8	86.6 ± 0.7	35.2 ± 1.2	40.7 ± 0.8	42.5 ± 1.5	27.6 ± 0.5	23.3 ± 0.8
MIR-mix-RAR (ours)	$\mathbf{89.0 \pm 0.7}$	$\mathbf{89.1 \pm 1.0}$	$\mathbf{37.6 \pm 1.0}$	$\mathbf{42.9 \pm 1.1}$	$\mathbf{44.7 \pm 1.3}$	$\mathbf{29.9 \pm 0.4}$	$\mathbf{27.7 \pm 0.5}$
ASER	79.6 ± 2.5	79.5 ± 1.5	24.2 ± 1.2	32.2 ± 1.5	37.0 ± 2.3	26.5 ± 0.6	25.0 ± 0.8
ASER-RAR (ours)	$\mathbf{81.3 \pm 1.5}$	$\mathbf{83.2 \pm 1.1}$	$\mathbf{34.5 \pm 1.2}$	$\mathbf{40.4 \pm 1.0}$	$\mathbf{42.7 \pm 1.3}$	$\mathbf{29.5 \pm 0.4}$	$\mathbf{26.9 \pm 0.7}$
ASER-mix	80.2 ± 1.8	79.6 ± 1.7	25.9 ± 1.2	29.5 ± 1.8	34.0 ± 1.7	26.8 ± 0.6	22.1 ± 0.9
ASER-mix-RAR (ours)	$\mathbf{82.0 \pm 1.6}$	$\mathbf{84.0 \pm 1.2}$	$\mathbf{35.6 \pm 0.9}$	$\mathbf{39.2 \pm 1.4}$	$\mathbf{41.5 \pm 1.6}$	$\mathbf{29.9 \pm 0.3}$	$\mathbf{27.5 \pm 0.5}$
iid-online	86.8 ± 1.1	86.8 ± 1.1	60.8 ± 1.0	60.8 ± 1.0	60.8 ± 1.0	18.13 ± 0.8	17.53 ± 1.6
iid-offline	92.3 ± 0.5	92.3 ± 0.5	79.2 ± 0.4	79.2 ± 0.4	79.2 ± 0.4	42.0 ± 0.9	37.46 ± 1.3

Table 8.3: Average Forgetting (↓) on different task-free CL datasets across different buffer sizes (m). We averaged results of over 15 runs for all except for Split-MNIST, where we averaged over 20 runs.

	Split-MNIST		**Split-CIFAR10**			**Split-CIFAR100**	**Split-miniImageNet**
Method	m=500	m=1000	m=200	m=500	m=1000	m=10,000	m=10,000
ER	17.1 ± 2.6	13.1 ± 1.8	51.3 ± 4.1	39.0 ± 2.1	26.3 ± 2.4	17.1 ± 0.5	28.3 ± 0.6
ER-RAR (ours)	$\mathbf{11.0 \pm 2.5}$	$\mathbf{7.4 \pm 1.1}$	$\mathbf{37.6 \pm 2.1}$	$\mathbf{25.1 \pm 2.5}$	$\mathbf{18.1 \pm 2.0}$	$\mathbf{10.8 \pm 0.4}$	$\mathbf{16.4 \pm 0.8}$
ER-mix	17.3 ± 1.6	15.8 ± 1.5	54.9 ± 5.2	37.2 ± 3.5	25.8 ± 4.0	21.9 ± 1.1	33.9 ± 1.5
ER-mix-RAR (ours)	$\mathbf{9.6 \pm 1.4}$	$\mathbf{8.4 \pm 1.5}$	$\mathbf{27.4 \pm 2.3}$	$\mathbf{20.1 \pm 1.6}$	$\mathbf{15.4 \pm 1.1}$	$\mathbf{10.7 \pm 0.4}$	$\mathbf{16.5 \pm 1.1}$
MIR	10.7 ± 1.6	7.5 ± 1.8	48.1 ± 2.9	32.2 ± 2.1	19.1 ± 1.8	13.8 ± 0.4	21.4 ± 0.6
MIR-RAR (ours)	$\mathbf{6.1 \pm 1.2}$	$\mathbf{3.1 \pm 0.7}$	$\mathbf{33.5 \pm 2.2}$	$\mathbf{21.1 \pm 2.2}$	$\mathbf{14.1 \pm 1.6}$	$\mathbf{10.9 \pm 0.5}$	$\mathbf{15.5 \pm 0.8}$
MIR-mix	8.7 ± 0.9	7.8 ± 1.1	$\mathbf{24.7 \pm 2.6}$	$\mathbf{16.1 \pm 2.2}$	$\mathbf{12.9 \pm 2.7}$	16.4 ± 0.6	30.3 ± 1.2
MIR-mix-RAR (ours)	$\mathbf{4.6 \pm 1.0}$	$\mathbf{4.1 \pm 1.0}$	26.6 ± 1.9	17.4 ± 1.7	13.1 ± 2.0	$\mathbf{11.3 \pm 0.5}$	$\mathbf{13.7 \pm 1.0}$
ASER	19.1 ± 3.0	17.8 ± 2.4	64.3 ± 3.3	45.5 ± 3.3	29.3 ± 3.4	16.8 ± 0.6	27.0 ± 1.0
ASER-RAR (ours)	$\mathbf{15.4 \pm 2.2}$	$\mathbf{12.0 \pm 1.9}$	$\mathbf{35.4 \pm 2.0}$	$\mathbf{21.4 \pm 2.0}$	$\mathbf{14.5 \pm 1.6}$	$\mathbf{8.9 \pm 0.4}$	$\mathbf{15.9 \pm 1.0}$
ASER-mix	18.6 ± 2.2	18.9 ± 2.6	54.7 ± 3.8	50.1 ± 4.7	38.1 ± 3.3	21.7 ± 0.8	33.7 ± 1.0
ASER-mix-RAR (ours)	$\mathbf{16.1 \pm 2.3}$	$\mathbf{12.0 \pm 1.7}$	$\mathbf{30.3 \pm 1.8}$	$\mathbf{21.5 \pm 3.3}$	$\mathbf{18.7 \pm 3.2}$	$\mathbf{11.0 \pm 0.5}$	$\mathbf{12.4 \pm 1.0}$

Table 8.4: Comparison of RAR with other non-RAR baselines having similar computational costs. Numbers within brackets denote the overall buffer size used.

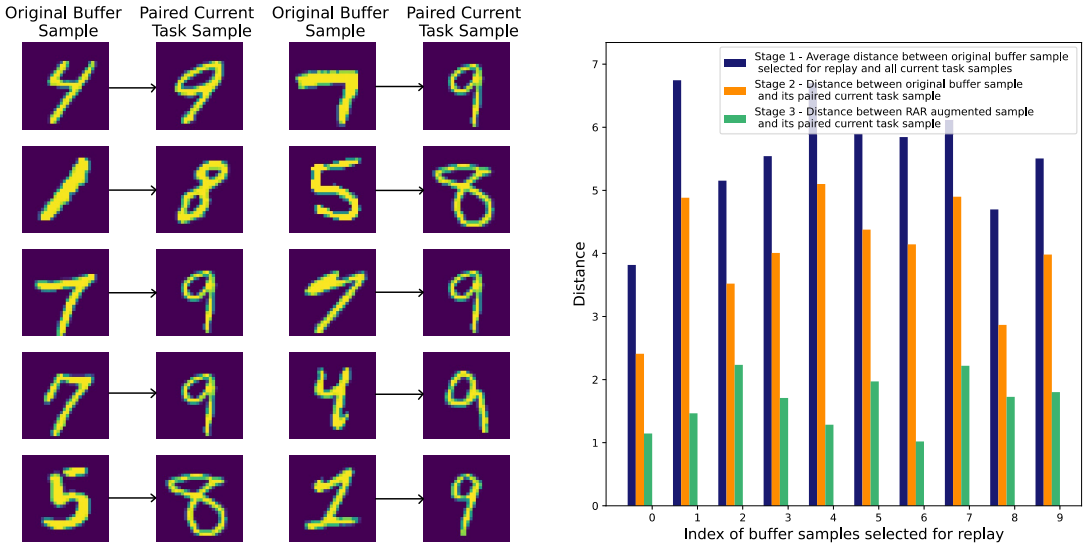
	Split-MNIST (200)		**Split-CIFAR10 (200)**		**Split-CIFAR10 (500)**		**Split-CIFAR10 (1000)**	
Method	Accuracy (↑)	Forgetting (↓)	Accuracy	Forgetting	Accuracy	Forgetting	Accuracy	Forgetting
ER-3-iters	82.3 ± 0.8	17.6 ± 0.8	28.5 ± 1.3	52.5 ± 3.7	36.1 ± 1.5	39.5 ± 2.1	41.5 ± 1.2	30.8 ± 1.9
ER-GMED	82.5 ± 1.4	15.0 ± 1.7	30.4 ± 0.9	39.0 ± 1.3	38.3 ± 1.1	27.7 ± 1.3	42.8 ± 1.5	20.5 ± 1.2
ER-RAR (ours)	$\mathbf{84.0 \pm 1.1}$	$\mathbf{11.8 \pm 1.5}$	$\mathbf{33.2 \pm 1.4}$	$\mathbf{37.6 \pm 2.1}$	$\mathbf{40.4 \pm 1.7}$	$\mathbf{25.1 \pm 2.5}$	$\mathbf{44.0 \pm 1.4}$	$\mathbf{18.1 \pm 2.0}$
MIR-3-iters	83.1 ± 0.8	15.3 ± 1.4	28.4 ± 1.0	53.4 ± 2.1	41.1 ± 0.8	33.3 ± 1.5	$\mathbf{48.9 \pm 1.1}$	22.4 ± 1.9
MIR-GMED	84.2 ± 1.2	12.2 ± 1.0	27.5 ± 1.3	45.5 ± 1.6	37.2 ± 1.1	28.6 ± 1.9	43.0 ± 1.5	17.6 ± 2.1
MIR-RAR (ours)	$\mathbf{86.2 \pm 0.6}$	$\mathbf{9.1 \pm 0.8}$	$\mathbf{33.3 \pm 1.1}$	$\mathbf{33.5 \pm 2.2}$	$\mathbf{43.1 \pm 1.1}$	$\mathbf{21.1 \pm 2.2}$	45.3 ± 1.5	$\mathbf{14.1 \pm 1.6}$

retrieval methods consistently outperform their respective non-RAR baselines (without and with mixup) across all four datasets. From Table 8.3, we observe that the improvements obtained by integrating *RAR* with respective replay selection methods {*ER, MIR & ASER*} (both without and with mixup) are significant. In the case of Split-CIFAR10, *MIR-mix* tends to perform slightly better than *MIR-mix-RAR* in terms of forgetting, but based on accuracy improvement that *MIR-mix-RAR* brings, the relative difference is negligible since forgetting is dependent on the maximum accuracy that a continual learner achieves across different tasks.

In the case of Split-miniImageNet, the ER-mix-RAR method achieves the best average accuracy of 28.8%, outperforming other MIR & ASER-based methods, and has a similar trend on the Split-CIFAR100 dataset. We hypothesize that for large buffer sizes such as 10k (100 examples per class), the ER-based selection aids in sampling a diverse set of classes from the memory buffer compared to MIR & ASER, thus improving the overall accuracy. However, since MIR & ASER consider samples suffering interference from the new task's data, MIR-mix-RAR & ASER-mix-RAR help minimize the forgetting of the old tasks.

The improved performance of ER-RAR, MIR-RAR, and ASER-RAR compared to ER, RAR, and ASER, respectively, demonstrate that RAR is able to alleviate catastrophic forgetting on old tasks by generating diverse and confusing samples close to the forgetting frontier. To ensure fair comparison in terms of computational costs, in Table 8.4, we also compare RAR to GMED [130], ER-3-iters, and MIR-3-iters. 3-iters means that the model is trained for multiple iterations on the incoming batch and buffer samples as a new batch comes in.

Case study: Visualizing the Three Stages of RAR For the Split-MNIST dataset, we visually analyze the three stages of the RAR framework for a buffer size of 500 using MIR as the memory-retrieval method. Fig. 8.2a shows the samples selected for replay using the MIR score (Eq. 8.1) and their paired samples from the current task's batch based on distances computed using Eq. (8.11). Fig. 8.2b shows how the distance between a replaying sample and the current task's data changes over the three stages: (1) for each sample selected from



(a) (b)

Figure 8.2: Split-MNIST - (a) Buffer samples selected for replay using MIR strategy along with their paired current-task sample. (b) Visualization of how the distance between the buffered replay samples and its nearest neighbor in the current tasks' batch evolves across different stages of RAR

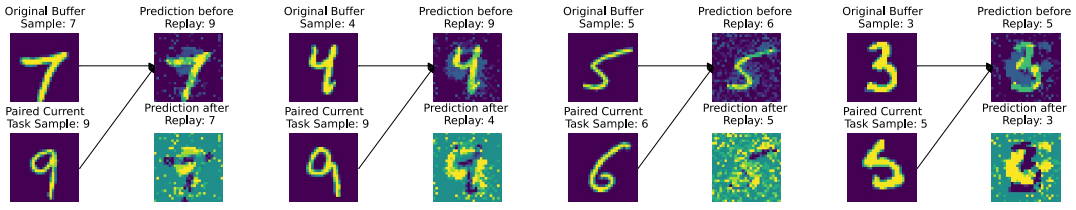


Figure 8.3: Retrospective Adversarial Perturbations (top-right of each sub-figure) for different replay samples shown alongside original replay samples (top-left) and paired current-task sample (bottom-left). The bottom-right image shows the scaled perturbation that was added to the original buffered sample to generate the RAR perturbed data.

the buffer, the blue bar represents its average distance to all samples in the current-task batch; (2) after pairing each buffered replay sample with its nearest neighbor as described in Sec. 8.4.3, its distance to the paired sample (orange bar) becomes smaller than the blue bar; (3) after applying adversarial perturbations to each buffered sample as described in Sec. 8.4.1, the distance is further reduced.

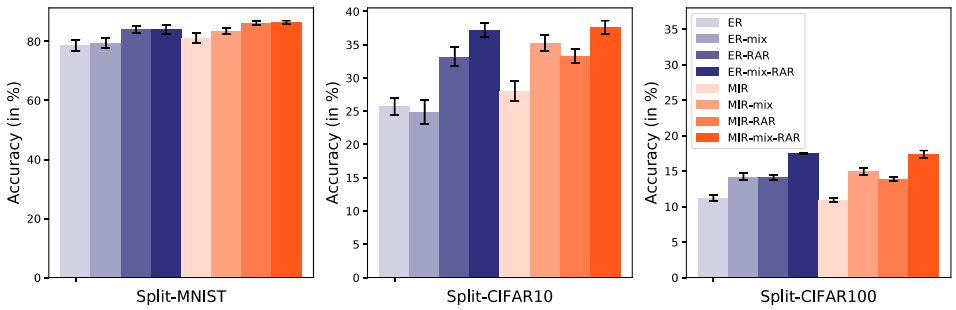
The decrease in distance indicates that the pairing and adversarial perturbation in RAR are effective in generating samples closer to the forgetting frontier than the original buffer data selected for replay, meaning the RAR perturbations are similar to X_M but very close to the paired samples in X_i (current task's batch) in the *l-th* layer latent representation space of model θ. In Fig. 8.3, we show four examples of RAR perturbed samples generated via Eq. (8.10) from four buffer samples and their paired samples from the current task. For most of the buffer samples, a single gradient step of θ replaying them and their RAR perturbed counterparts suffices to correct their wrong predictions (e.g., the current task's classes), hence effectively alleviating the model's forgetting on previous tasks.

8.5.2 Advantage of MixUp in Small-Buffer Cases

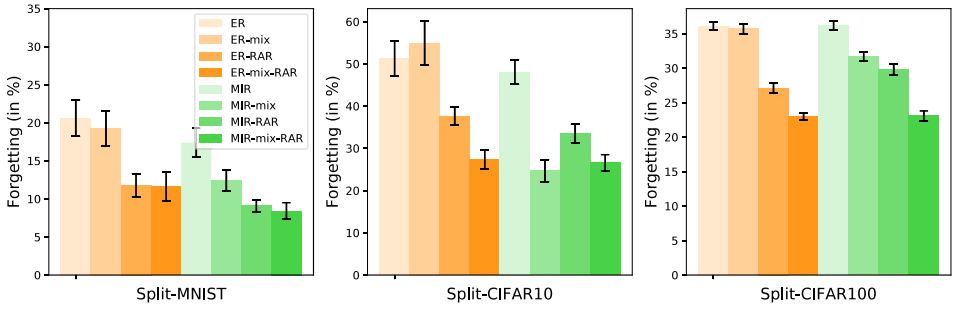
On each dataset, as reported in Table 8.3, all mix-RAR methods perform better compared to RAR methods in terms of reducing forgetting on previous tasks with slight improvement in average accuracy. The proposed use of MixUp among the replay samples before RAR is able to increase the variation of RAR perturbed samples, which is essential to CL with a small buffer. To verify its effectiveness, we conduct experiments that store only 20 examples per class. Fig. 8.4 shows that ER-mix-RAR & MIR-mix-RAR consistently outperform other ER & MIR methods respectively. We can achieve even more significant improvements in forgetting, particularly for more challenging tasks such as Split-CIFAR100.

8.5.3 Ablation Results

In Table 8.5, we compare RAR methods to two baseline methods: (a) *Random* which add a perturbation of the same tolerance as the RAR method to the replay samples before the



(a) Accuracy (↑)



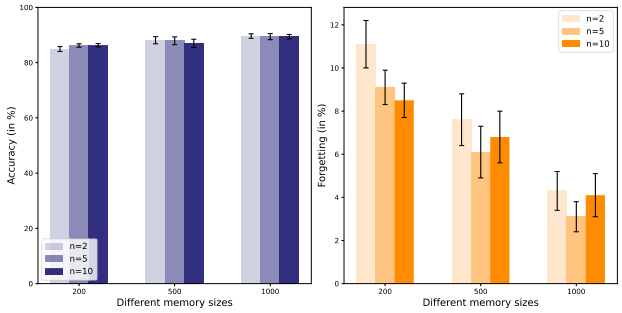
(b) Forgetting (↓)

Figure 8.4: Results under small buffer constraints, 20 examples per class for each dataset

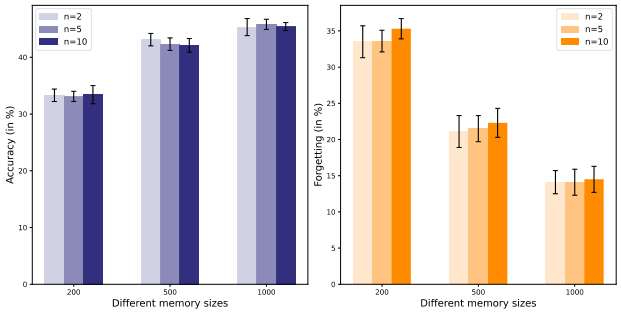
Table 8.5: Comparison of RAR to other replay data perturbation methods: (a) Random edit using same perturbation tolerance as RAR (b) GMED. Numbers within brackets denote the overall buffer size used.

	Split-MNIST (200)		**Split-CIFAR10 (200)**		**Split-CIFAR10 (500)**		**Split-CIFAR100 (10k)**	
Method	Accuracy	Forgetting	Accuracy	Forgetting	Accuracy	Forgetting	Accuracy	Forgetting
ER-Random	80.2 ± 2.3	18.7 ± 3.1	28.5 ± 1.6	41.9 ± 2.5	38.3 ± 1.1	30.2 ± 1.6	26.3 ± 0.5	14.4 ± 0.5
ER-GMED	82.5 ± 1.4	15.0 ± 1.7	30.4 ± 0.9	39.0 ± 1.3	38.3 ± 1.1	27.7 ± 1.3	25.5 ± 0.5	13.5 ± 0.5
ER-RAR (ours)	$\mathbf{84.0 \pm 1.1}$	$\mathbf{11.8 \pm 1.5}$	$\mathbf{33.2 \pm 1.4}$	$\mathbf{37.6 \pm 2.1}$	$\mathbf{40.4 \pm 1.7}$	$\mathbf{25.1 \pm 2.5}$	$\mathbf{29.8 \pm 0.4}$	$\mathbf{10.8 \pm 0.4}$
MIR-Random	82.3 ± 1.6	15.9 ± 1.7	29.1 ± 1.2	45.3 ± 2.4	39.2 ± 1.3	29.0 ± 2.1	26.5 ± 0.3	13.6 ± 0.5
MIR-GMED	84.2 ± 1.2	12.2 ± 1.0	27.5 ± 1.3	45.5 ± 1.6	37.2 ± 1.1	28.6 ± 1.9	25.1 ± 0.3	11.7 ± 0.4
MIR-RAR (ours)	$\mathbf{86.2 \pm 0.6}$	$\mathbf{9.1 \pm 0.8}$	$\mathbf{33.3 \pm 1.1}$	$\mathbf{33.5 \pm 2.2}$	$\mathbf{43.1 \pm 1.1}$	$\mathbf{21.1 \pm 2.2}$	$\mathbf{29.3 \pm 0.4}$	$\mathbf{10.9 \pm 0.5}$

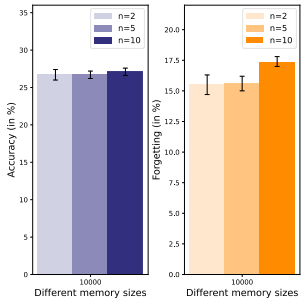
experience replay stage, (b) *GMED* [130] which performs smart gradient updates in the input space by treating each replay sample individually (no local pairing considered between buffer and current task's data). As can be seen in Table 8.5, ER-RAR and MIR-RAR consistently outperform other methods, confirming RAR's superior ability to tackle catastrophic forgetting via generating targeted perturbations influenced by the local interference between different task data.



(a) Split-MNIST



(b) Split-CIFAR10

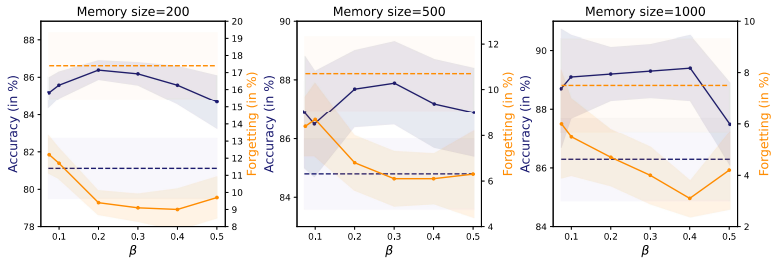


(c) Split-miniImg

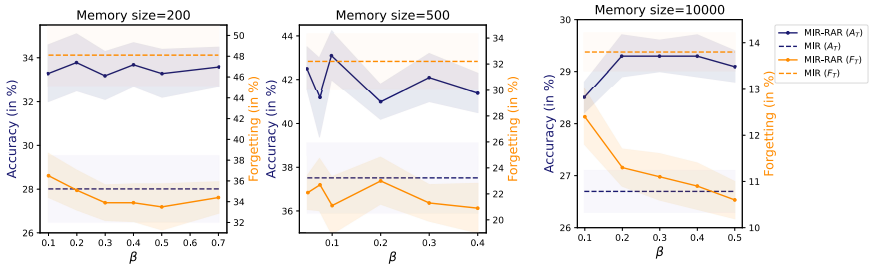
Figure 8.5: Sensitivity of the performance of RAR (using MIR) to the number of perturbation steps (n) used to generate the RAR perturbed samples on different datasets under various buffer constraints.

8.5.4 Sensitivity Analysis

Number of gradient steps used for computing the adversarial perturbation In Fig. 8.5, we plot the sensitivity of RAR's performance with respect to the number of gradient steps n used to generate the retrospective adversarial samples via optimizing Eq. (8.10) in the input space. For Split-CIFAR10 and Split-miniImageNet (where we report results using two gradient steps), we observe that for a larger number of steps $\{5, 10\}$, the performance stays comparable. This suggests that nearest-neighbor-based pairing using distance computations (covered in Sec 8.4.3) helps achieve anchor-target pairs that are already optimized and, hence, do not require numerous gradient updates to get classified as the paired target class from the current task.



(a) Split-MNIST



(b) Split-CIFAR10 (c) Split-CIFAR100

Figure 8.6: Sensitivity of the performance of RAR (using MIR) to the trade-off coefficient (β) dedicated to ℓ_{RAR} on different datasets under various buffer constraints.

Sensitivity to β coefficient Next, we study the sensitivity of RAR corresponding to β, which denotes the trade-off coefficient associated with ℓ_{RAR} in the final loss objective Eq. (8.12)—we use only one coefficient to reduce hyperparameter search costs. Fig. 8.6 shows that RAR (using MIR) outperforms vanilla MIR in terms of both accuracy (solid blue line is always above the dashed blue line) and forgetting (solid orange line always stays below the dashed orange line) for various non-zero values of β across the studied datasets.

Virtual update for RAR samples generation When training sequentially on a set of tasks, catastrophic forgetting of previous tasks occurs as the network weights become biased to meet the objectives of the new task. Performing a single-step virtual update of the model by exposing it only to the current task's data helps us explore which buffer samples are most likely to be forgotten. We use θ', the virtually updated model, to generate the adversarial samples using Eq. (8.10) as such perturbed samples are likely to be forgotten in future updates and confused with the current task's classes, leading to increased loss on them. In retrospect, using the generated perturbed sample (visually close to the original sample x_M in the input space and hence in-distribution) with its correct label $y(x_M)$ and replaying it while we are still at θ helps in alleviating forgetting. As we combine the losses, we may also consider the procedure as first doing an update on the incoming task to get θ' and then performing an adversarial training step based on θ' using the RAR augmented samples.

One can argue that the step size used to perform this virtual update (η_{virtual}) could be different from the step size (η) used to update the continual learner such that the adversarial samples generated using θ' are not too easy or too hard to classify by the current model θ. We perform experiments on the Split-CIFAR10 dataset with a buffer size of 500 and an SGD optimizer with a learning rate (η) of 0.1. We find that using the same learning rate as the SGD optimizer to perform the virtual update results in the best performance on MIR-RAR, as shown in Table 8.6.

Table 8.6: Performance of MIR-RAR for different values of η_{virtual} when $\eta = 0.1$

η_{virtual}	Accuracy (↑)	Forgetting (↓)
No virtual update	41.7 ± 1.3	21.8 ± 1.4
0.02	42.1 ± 1.6	22.1 ± 2.8
0.05	42.6 ± 1.1	20.6 ± 3.7
0.1	43.1 ± 1.1	21.1 ± 2.2
0.15	43.1 ± 1.6	22.1 ± 3.5
0.2	41.9 ± 1.1	23.1 ± 3.6

Ratio between incoming mini-batch size and replay budget We select the same number of replay samples from the buffer as the mini-batch size of the incoming new task following prior works [7, 130, 301]. This is set to 10, irrespective of the buffer size. Here, we perform additional experiments where we vary the ratio between the incoming mini-batch size and replay budget. As expected, when the combined batch is dominated primarily by the incoming task samples (mini-batch size/replay budget = 15/5), we observe a significant drop in the overall performance with respect to both the average accuracy and forgetting as shown in Table 8.7. However, when replaying more samples from the buffer (mini-batch size/replay budget = 5/15) in each training iteration, which over-compensates for forgetting, we do not observe any improvement in reducing the forgetting. This suggests that to achieve a good plasticity-stability trade-off, the ratio between incoming mini-batch size and replay budget needs to be close to one.

8.5.5 Extended Comparisons

Comparison to knowledge distillation based methods On the Split-CIFAR10 dataset for two different buffer sizes, we augment RAR with DER [38], which utilizes knowledge distillation in the logits space. DER, which imposes regularization in the logits space, can be seen as a complimentary method to RAR, and we observe that combining them together

Table 8.7: Performance of replay-based continual learning methods when using different ratios between incoming mini-batch size and replay budget. The dataset used is Split-CIFAR10 with a buffer size of 500.

Ratio	**10/10**		**5/15**		**15/5**	
Method	Accuracy (↑)	Forgetting (↓)	Accuracy (↑)	Forgetting (↓)	Accuracy (↑)	Forgetting (↓)
ER	31.9 ± 0.9	39.0 ± 2.1	30.5 ± 2.7	53.0 ± 4.4	20.9 ± 2.1	55.7 ± 6.1
ER-RAR	40.4 ± 1.7	25.1 ± 2.5	40.2 ± 2.2	26.0 ± 3.2	26.0 ± 3.6	31.4 ± 3.5
MIR	37.5 ± 1.6	32.2 ± 2.1	37.4 ± 1.4	35.1 ± 2.0	16.5 ± 1.5	68.5 ± 3.3
MIR-RAR	43.1 ± 1.1	21.1 ± 2.2	41.0 ± 1.4	22.9 ± 3.2	18.6 ± 2.6	58.0 ± 5.1

leads to further improvements in Table 8.8. ER-DER is essentially DER that uses random sampling to select the replay samples and was proposed in the DER paper. In MIR-DER, we use MIR scores to select the replay batch for DER. In Table 8.8, *aug* means that we apply standard augmentations such as random cropping and rotations on the replay samples.

Comparison to other augmentation techniques In Table 8.5, we compare RAR to another perturbation method *random*, which adds random noise of the same perturbation magnitude as RAR, and show that RAR leads to significant improvement in reducing forgetting as well as increasing the average accuracy on different CL benchmark datasets. In Table 8.9, we compare to other standard augmentations techniques, such as random cropping and rotations applied to the replay samples. Utilizing these standard augmentation strategies along with RAR leads to impressive improvements in terms of the forgetting metric, as shown in Table 8.9.

When using standard augmentations along with RAR, ER-RAR-aug, and MIR-RAR-aug achieve similar performance irrespective of the memory retrieval/selection method used. This indicates that RAR, when combined with standard augmentations such as random cropping and rotations, is more robust to replay data selection and can generate high-quality replay

Table 8.8: Comparison of RAR with DER (with and without augmentation) to non-RAR baselines. Numbers within brackets denote the overall buffer size used.

	Split-CIFAR10 (200)		**Split-CIFAR10 (500)**	
Method	Accuracy (↑)	Forgetting (↓)	Accuracy (↑)	Forgetting (↓)
ER-DER	28.0 ± 1.3	47.2 ± 7.2	32.9 ± 1.9	28.4 ± 3.3
ER-RAR-DER	36.0 ± 1.6	21.2 ± 2.6	39.8 ± 0.5	15.5 ± 2.2
ER-DER-aug	34.0 ± 3.3	37.4 ± 7.6	39.6 ± 1.5	18.2 ± 4.1
ER-RAR-DER-aug	36.7 ± 1.5	15.8 ± 2.3	40.3 ± 1.1	10.3 ± 1.9
MIR-DER	36.6 ± 2.3	20.0 ± 1.5	38.6 ± 1.7	10.8 ± 1.9
MIR-RAR-DER	35.9 ± 1.9	20.1 ± 3.3	38.5 ± 0.8	10.8 ± 1.8
MIR-DER-aug	36.5 ± 2.3	17.0 ± 1.8	39.5 ± 1.0	15.0 ± 1.7
MIR-RAR-DER-aug	37.2 ± 1.8	13.4 ± 1.1	40.1 ± 0.8	9.5 ± 2.3

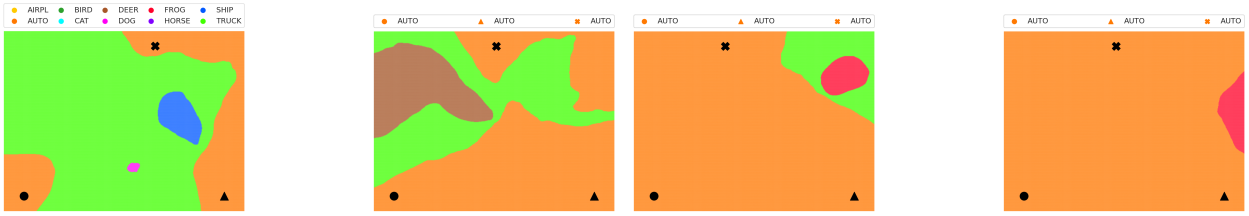
Table 8.9: Comparison of RAR to standard augmentation techniques. Numbers within brackets denote the overall buffer size used.

	Split-CIFAR10 (200)		**Split-CIFAR10 (500)**	
Method	Accuracy (↑)	Forgetting (↓)	Accuracy (↑)	Forgetting (↓)
ER	25.7 ± 1.3	51.3 ± 4.1	31.9 ± 0.9	39.0 ± 2.1
ER-aug	30.0 ± 4.1	43.2 ± 6.9	39.4 ± 2.8	26.0 ± 6.3
ER-RAR	33.2 ± 1.4	37.6 ± 2.1	40.4 ± 1.7	25.1 ± 2.5
ER-RAR-aug	39.3 ± 1.6	21.6 ± 3.4	42.2 ± 1.9	13.6 ± 2.5
MIR	28.0 ± 1.5	48.1 ± 2.9	37.5 ± 1.6	32.2 ± 2.1
MIR-aug	36.1 ± 2.6	23.7 ± 3.6	40.5 ± 1.2	18.4 ± 3.3
MIR-RAR	33.3 ± 1.1	33.5 ± 2.2	43.1 ± 1.1	21.1 ± 2.2
MIR-RAR-aug	41.6 ± 1.2	21.0 ± 2.8	42.2 ± 2.3	12.5 ± 2.6

samples (diverse, confusing, and representative of previously seen tasks).

However, such augmentation strategies rely on strong human priors, and manually selecting such augmentation operators requires domain expertise and prior knowledge of the dataset of interest [398]. They may perform poorly in scenarios when human knowledge is weak for the targeted domain. For example, for medical image analysis, such simple transformations are shown to be insufficient in capturing many of the subtle variations present in the data [394]. Moreover, they are static and pre-defined before the training so they cannot capture the training dynamics during the continual learning process for the purpose of reducing the forgetting. RAR, on the other hand, automatically generates augmentations that are adaptive to the forgetting dynamics of the model and thus focuses on reducing the local interference near the forgetting frontier between different tasks.

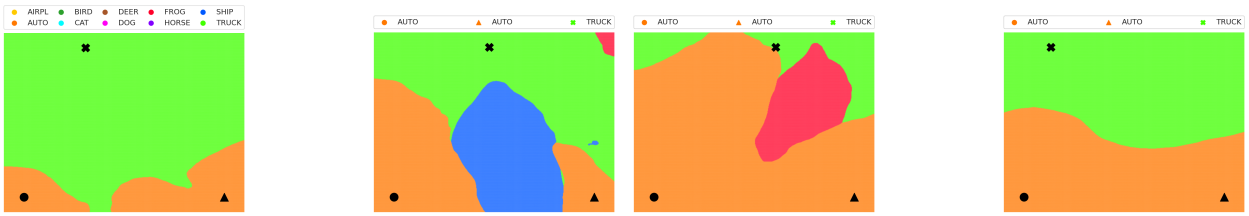
8.5.6 Visualizations of Learned Decision Regions



(a) ER (b) MIR (c) MIR-RAR (d) MIR-mix-RAR

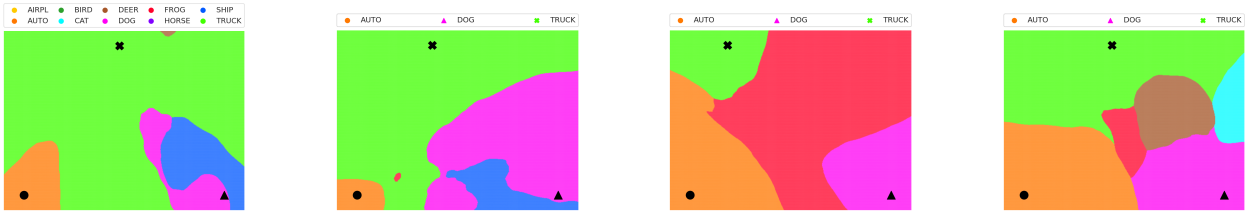
Figure 8.7: Decision regions are generated using a triplet of training samples on which the model makes the right predictions. All samples are from the same class *Auto (Task 1)*. Legend in (a) applies to all and describes the color used to denote different classes. The remaining legends denote the classes corresponding to the chosen triplet samples.

Inspired by recent works in decision boundaries visualization [307], we try to visualize the decision regions learned by the continual learner along the data manifold. To plot the decision regions, we use the plotting technique from [307]. Using a sampled triplet of data



(a) ER (b) MIR (c) MIR-RAR (d) MIR-mix-RAR

Figure 8.8: Decision regions are generated using a triplet of training samples on which the model makes the right predictions. Two samples belong to class *Auto (Task 1)*, while the third sample is from class *Truck (Task 5)*. Legend in (a) applies to all and describes the color used to denote different classes. The remaining legends denote the classes corresponding to the chosen triplet samples.



(a) ER (b) MIR (c) MIR-RAR (d) MIR-mix-RAR

Figure 8.9: Decision regions are generated using a triplet of training samples on which the model makes the right predictions. Samples are from the following classes: *Auto (Task 1)*, *Dog (Task 3)*, *Truck (Task 5)*. Legend in (a) applies to all and describes the color used to denote different classes. The remaining legends denote the classes corresponding to the chosen triplet samples.

points (x_1, x_2, x_3), we construct a plane spanned by vectors $\overrightarrow{v_1} = x_2 - x_1$ and $\overrightarrow{v_2} = x_3 - x_1$ and plot decision regions in this plane.

Specifically, for our use case, once the training on all tasks (5 in the case of Split-CIFAR10) is finished, we use the trained model to make predictions on the trained data. We, then, select three training set instances (x_1, x_2, x_3) on which the model makes the correct predictions in the following ways: (1) all samples are from the same class, (2) two samples share the same class label while the third belongs to a different class, and (3) all three samples are from different classes.

We analyze the decision regions learned by the ResNet-18 model using ER [45], MIR [7], MIR-RAR, and MIR-mix-RAR methods on the Split-CIFAR10 dataset. Task to classes mapping is as follows: ***Task 1***: {*AIRPL, AUTO*}, ***Task 2***: {*BIRD, CAT*}, ***Task 3***: {*DEER, DOG*}, ***Task 4***: {*FROG, HORSE*}, ***Task 5***: {*SHIP, TRUCK*}.

For case (1), we select all samples from class *Auto* and plot the regions in Fig 8.7. It is expected that in case of worse forgetting, the decision regions would be dominated by the last task classes even when we use the first task's samples to model the visualization plane. In Fig. 8.7, the plane is mostly dominated by green (and blue) color in the case of ER. In the case of MIR-mix-RAR, we see the plane mostly covering the class used to construct the visualization plane, even when the class selected (*Auto*) belongs to the first task the model encountered.

Similarly, in Fig. 8.8 (case 2) & 8.9 (case 3), as we go from left to right, we see the decision regions being more structured around the classes covered in the triplet set and less dominated by the color of the classes from the last task—this demonstrates that forgetting is minimized when using RAR and mix-RAR.

8.6 Summary

In this chapter, we study how to perturb limited buffer replay data to mitigate catastrophic forgetting in continual learning. We propose "Retrospective Adversarial Replay (RAR)", which focuses on data from previous tasks near the forgetting boundary of the current task

in the model's representation space. RAR first allocates the most confusing pairs of a buffered sample and a current task's sample and then applies a bounded targeted adversarial perturbation to the former, which further moves it towards the latter in the model's representation space and creates a perturbed sample close to the boundary. Moreover, we study the role of MixUp in increasing the variation of replay augmentations, which significantly improves CL in the small buffer regime. When integrated with existing replay methods, RAR consistently outperforms previous CL methods across benchmark datasets under different settings.

Chapter 9

CONCLUSION & FUTURE WORK

9.1 Summary

This dissertation addresses a core challenge of resource efficiency in modern machine learning (ML) by leveraging two complementary strategies: submodular data selection and data augmentation. These approaches are designed to reduce computational, data, and memory costs across different stages of the ML lifecycle—annotation, training, and inference—while maintaining or improving model performance. The increasing complexity of LLMs, ViTs, and MLLMs has led to significant demands on training data, inference latency, and memory requirements, which make their deployment in real-world applications challenging. The proposed approaches introduce principled submodular optimization methods to optimize data and context selection at various stages of the ML pipeline, leading to more efficient learning in resource-constrained and data-scarce environments.

9.1.1 Submodular Query-Focused Summarizaton

A core contribution of this dissertation is the development of the Submodular Span Summarization (S3) framework for query-focused summarization (Chapter 4). S3 introduces a two-stage submodular approach: it first filters out non-relevant items by minimizing conditional gain (Stage 1) and then applies submodular summarization (Stage 2) to reduce redundancy within the query-relevant subset. This two-stage design helps model the properties of query relevance and diversity in the final summary set, making S3 particularly effective for tasks such as active learning, targeted data selection, and efficient query-guided input-context selection.

9.1.2 Submodular Input-Context Selection in LLMs and MLLMs

Building upon S3, this dissertation proposes Div-S3, an end-to-end submodular optimization framework for In-Context Learning (ICL) using LLMs (Chapter 5). Div-S3 enables efficient exemplar annotation and retrieval, improving performance under annotation budget constraints. This contributes to improving test-time input context selection, a key challenge in large-scale inference, where selecting only the most relevant and diverse context items is critical for minimizing compute and memory overhead.

Another significant contribution is the introduction of BumbleBee, a submodular optimization based approach for KV cache summarization in LLMs (Chapter 6). As the context window of autoregressive LLMs scales, their growing KV cache size becomes a major memory bottleneck for inference efficiency. BumbleBee dynamically selects the most informative key-value pairs using a submodular mixture function capturing token representativeness and importance, enabling efficient infinite-context retention without architectural modifications or additional fine-tuning. This approach reduces memory footprint and latency, making it suitable for scalable deployment of LLMs in resource-constrained settings.

To further generalize this approach to multimodal inference, this dissertation introduces VisionBee, a vision counterpart of BumbleBee (Chapter 7). VisionBee applies submodular optimization to perform visual token summarization in MLLMs, selectively retaining informative and diverse visual tokens (or patches), thereby reducing computational and memory costs for image understanding tasks.

9.1.3 Data Augmentation

Beyond data and input context selection, this dissertation also explores data augmentation to mitigate forgetting and improve generalization while continually learning an ML model under low-resource settings. In particular, the proposed Retrospective Adversarial Replay (RAR) framework (Chapter 8) generates informative replay instances near the forgetting frontier using adversarial perturbations and MixUp-based augmentation. Replaying such

samples enables the model to retain performance across all previously learned tasks without requiring access to the full original data from those tasks.

Together, these contributions offer a unified framework for resource-efficient learning across a wide range of ML paradigms and data modalities. The proposed techniques improve data utilization, reduce memory and compute requirements, and enhance scalability—making ML systems more accessible, robust, and deployable in real-world environments.

9.2 Future Directions

Adaptive Summarization Budgets & Hierarchical Token Pruning in BumbleBee Current KV cache summarization techniques apply static token selection thresholds, which may not be optimal across different model sizes and sequence lengths. Future work can explore adaptive summarization budgets, where the summarization budget is dynamically adjusted based on attention patterns, task complexity, and available memory. Additionally, instead of applying summarization uniformly across all layers of the LLM decoder, a promising direction would be to prune more aggressively at lower layers while preserving richer context representations at higher layers. This hierarchical pruning approach is motivated by the observation that lower-layer representations typically capture more generic features, whereas higher layers encode more task-specific information. Dynamically pruning tokens earlier in the decoder could, therefore, lead to improved inference efficiency in large-scale models.

Long Video Understanding using Submodular Summarization An interesting direction for future work is extending BumbleBee to the domain of long video understanding by incorporating both spatial and temporal summarization of visual tokens. At the spatial level, VisionBee can be used to identify the most important and diverse set of visual tokens within each frame, reducing the number of visual tokens passed to the language model. At the temporal level, the online BumbleBee algorithm can be employed in the LLM decoder module of the MLLM to summarize and retain only the most informative visual tokens across

frames, thereby summarizing the overall video context. Together, this dual summarization strategy can significantly reduce the computational and memory overhead in MLLMs while maintaining the key information needed to perform downstream video tasks accurately over extended video inputs.

Memory Management in LLM Agents via Submodular Summarization An effective future direction involves integrating submodular summarization techniques into memory modules for LLM-based agents. Extractive vanilla or query-focused textual summarization using submodular frameworks such as S3 can be used to distill essential information from long interaction histories or retrieved documents, ensuring that only the most diverse and informative content is retained. These extractive summaries can then be optionally further summarized using an LLM for abstractive refinement, further reducing the memory footprint. Complementing this, BumbleBee's online KV cache summarization can manage internal memory during inference by pruning redundant key-value attention states, enabling scalable reasoning in long-running agent workflows.

BIBLIOGRAPHY

[1] Muhammad Adnan, Akhil Arunkumar, Gaurav Jain, Prashant J. Nair, Ilya Soloveychik, and Purushotham Kamath. Keyformer: Kv cache reduction through key tokens selection for efficient generative inference, 2024.

[2] Ishika Agarwal, Krishna Killamsetty, Lucian Popa, and Marina Danilevksy. Delift: Data efficient language model instruction fine tuning. *arXiv preprint arXiv:2411.04425*, 2024.

[3] Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. Gqa: Training generalized multi-query transformer models from multi-head checkpoints, 2023.

[4] Alon Albalak, Yanai Elazar, Sang Michael Xie, Shayne Longpre, Nathan Lambert, Xinyi Wang, Niklas Muennighoff, Bairu Hou, Liangming Pan, Haewon Jeong, et al. A survey on data selection for language models. *arXiv preprint arXiv:2402.16827*, 2024.

[5] Keivan Alizadeh, Iman Mirzadeh, Dmitry Belenko, Karen Khatamifard, Minsik Cho, Carlo C Del Mundo, Mohammad Rastegari, and Mehrdad Farajtabar. Llm in a flash: Efficient large language model inference with limited memory. *arXiv preprint arXiv:2312.11514*, 2023.

[6] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 139–154, 2018.

[7] Rahaf Aljundi, Lucas Caccia, Eugene Belilovsky, Massimo Caccia, Min Lin, Laurent Charlin, and Tinne Tuytelaars. Online continual learning with maximally interfered retrieval. *arXiv preprint arXiv:1908.04742*, 2019.

[8] Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. Gradient based sample selection for online continual learning. *arXiv preprint arXiv:1903.08671*, 2019.

[9] Aida Amini, Saadia Gabriel, Shanchuan Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. Mathqa: Towards interpretable math word problem solving with operation-based formalisms. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2357–2367, 2019.

[10] Craig A Anderson, Mark R Lepper, and Lee Ross. Perseverance of social theories: The role of explanation in the persistence of discredited information. *Journal of personality and social psychology*, 39(6):1037, 1980.

[11] Anthopic. Claude 2.1. https://www.anthropic.com/news/claude-2-1. Accessed: 2024-03-25.

[12] Anthropic. Long context window tips — docs.anthropic.com. https://docs.anthropic.com/claude/docs/long-context-window-tips. [Accessed 29-03-2024].

[13] Relja Arandjelovic and Andrew Zisserman. Multiple queries for large scale specific object retrieval. In *BMVC*, pages 1–11, 2012.

[14] Sanjeev Arora, Yingyu Liang, and Tengyu Ma. A simple but tough-to-beat baseline for sentence embeddings. *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

[15] Jordan T Ash, Chicheng Zhang, Akshay Krishnamurthy, John Langford, and Alekh Agarwal. Deep batch active learning by diverse, uncertain gradient lower bounds. *arXiv preprint arXiv:1906.03671*, 2019.

[16] Jimmy Lei Ba. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[17] Ashwinkumar Badanidiyuru and Jan Vondrák. Fast algorithms for maximizing submodular functions. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 1497–1514. SIAM, 2014.

[18] Guangji Bai, Zheng Chai, Chen Ling, Shiyu Wang, Jiaying Lu, Nan Zhang, Tingwei Shi, Ziyang Yu, Mengdan Zhu, Yifei Zhang, et al. Beyond efficiency: A systematic survey of resource-efficient large language models. *arXiv preprint arXiv:2401.00625*, 2024.

[19] Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, et al. Longbench: A bilingual, multitask benchmark for long context understanding. *arXiv preprint arXiv:2308.14508*, 2023.

[20] Ravikumar Balakrishnan, Tian Li, Tianyi Zhou, Nageen Himayat, Virginia Smith, and Jeff Bilmes. Diverse client selection for federated learning via submodular maximization. In *International Conference on Learning Representations*, 2022.

[21] Jihwan Bang, Heesu Kim, YoungJoon Yoo, Jung-Woo Ha, and Jonghyun Choi. Rainbow memory: Continual learning with a memory of diverse samples. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8218–8227, 2021.

[22] Brian R Bartoldson, Bhavya Kailkhura, and Davis Blalock. Compute-efficient deep learning: Algorithmic trends and opportunities. *Journal of Machine Learning Research*, 24(122):1–77, 2023.

[23] Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854, 2019.

[24] Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

[25] Luisa Bentivogli, Bernardo Magnini, Ido Dagan, Hoa Trang Dang, and Danilo Giampiccolo. The fifth PASCAL recognizing textual entailment challenge. In *Proceedings of the Second Text Analysis Conference, TAC 2009, Gaithersburg, Maryland, USA, November 16-17, 2009*. NIST, 2009.

[26] Jeff Bilmes. Submodularity in machine learning and artificial intelligence. *arXiv preprint arXiv:2202.00132*, 2022.

[27] Jeff Bilmes. Submarine: SUBModularity for ARtificial INtelligencE and machine learning. Online Software System, 2025. https://submarine.page.

[28] Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7432–7439, 2020.

[29] Sid Black, Leo Gao, Phil Wang, Connor Leahy, and Stella Biderman. Gpt-neo: Large scale autoregressive language modeling with mesh-tensorflow. *If you use this software, please cite it using these metadata*, 58:2, 2021.

[30] Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your vit but faster. In *The Eleventh International Conference on Learning Representations*, 2023.

[31] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

[32] Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, et al. Improving language models by retrieving from trillions of tokens. In *International conference on machine learning*, pages 2206–2240. PMLR, 2022.

[33] Zalán Borsos, Mojmir Mutny, and Andreas Krause. Coresets via bilevel optimization for continual learning and streaming. *Advances in Neural Information Processing Systems*, 33:14879–14890, 2020.

[34] Damian Borth, Tao Chen, Rongrong Ji, and Shih-Fu Chang. Sentibank: large-scale ontology and classifiers for detecting sentiment and emotions in visual content. In *Proceedings of the 21st ACM international conference on Multimedia*, pages 459–460, 2013.

[35] Christopher Bowles, Liang Chen, Ricardo Guerrero, Paul Bentley, Roger Gunn, Alexander Hammers, David Alexander Dickie, Maria Valdés Hernández, Joanna Wardlaw, and Daniel Rueckert. Gan augmentation: Augmenting training data using generative adversarial networks. *arXiv preprint arXiv:1810.10863*, 2018.

[36] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[37] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*, 2023.

[38] Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline. *Advances in neural information processing systems*, 33:15920–15930, 2020.

[39] Xiaoyu Cao and Neil Zhenqiang Gong. Mitigating evasion attacks to deep neural networks via region-based classification. In *Proceedings of the 33rd Annual Computer Security Applications Conference*, pages 278–287, 2017.

[40] Jaime Carbonell and Jade Goldstein. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 335–336, 1998.

[41] Stephanie Chan, Adam Santoro, Andrew Lampinen, Jane Wang, Aaditya Singh, Pierre Richemond, James McClelland, and Felix Hill. Data distributional properties drive emergent in-context learning in transformers. *Advances in Neural Information Processing Systems*, 35:18878–18891, 2022.

[42] Ting-Yun Chang and Robin Jia. Data curation alone can stabilize in-context learning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8123–8144, 2023.

[43] Arslan Chaudhry, Puneet K Dokania, Thalaiyasingam Ajanthan, and Philip HS Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 532–547, 2018.

[44] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. *arXiv preprint arXiv:1812.00420*, 2018.

[45] Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K Dokania, Philip HS Torr, and Marc'Aurelio Ranzato. On tiny episodic memories in continual learning. *arXiv preprint arXiv:1902.10486*, 2019.

[46] Hung-Ting Chen and Eunsol Choi. Open-world evaluation for retrieving diverse perspectives. *arXiv preprint arXiv:2409.18110*, 2024.

[47] Jieneng Chen, Luoxin Ye, Ju He, Zhao-Yang Wang, Daniel Khashabi, and Alan Yuille. Llavolta: Efficient multi-modal models via stage-wise visual context compression. *arXiv preprint arXiv:2406.20092*, 2024.

[48] Liang Chen, Haozhe Zhao, Tianyu Liu, Shuai Bai, Junyang Lin, Chang Zhou, and Baobao Chang. An image is worth 1/2 tokens after layer 2: Plug-and-play inference acceleration for large vision-language models. In *European Conference on Computer Vision*, pages 19–35. Springer, 2024.

[49] Tao Chen, Damian Borth, Trevor Darrell, and Shih-Fu Chang. Deepsentibank: Visual sentiment concept classification with deep convolutional neural networks. *arXiv preprint arXiv:1410.8586*, 2014.

[50] Yang Chen, Hexiang Hu, Yi Luan, Haitian Sun, Soravit Changpinyo, Alan Ritter, and Ming-Wei Chang. Can pre-trained vision and language models answer visual information-seeking questions? In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 14948–14968, 2023.

[51] Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality, March 2023.

[52] Kanghyun Choi, Deokki Hong, Noseong Park, Youngsok Kim, and Jinho Lee. Qimera: Data-free quantization with synthetic boundary supporting samples. *Advances in Neural Information Processing Systems*, 34, 2021.

[53] Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*, 2020.

[54] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

[55] Thomas B Christophel, P Christiaan Klink, Bernhard Spitzer, Pieter R Roelfsema, and John-Dylan Haynes. The distributed nature of working memory. *Trends in cognitive sciences*, 21(2):111–124, 2017.

[56] Marvin M Chun and Nicholas B Turk-Browne. Interactions between attention and memory. *Current opinion in neurobiology*, 17(2):177–184, 2007.

[57] Cody Coleman, Christopher Yeh, Stephen Mussmann, Baharan Mirzasoleiman, Peter Bailis, Percy Liang, Jure Leskovec, and Matei Zaharia. Selection via proxy: Efficient data selection for deep learning. In *International Conference on Learning Representations*, 2020.

[58] Gerard Cornuejols, Marshall Fisher, and George L Nemhauser. On the uncapacitated location problem. In *Annals of Discrete Mathematics*, volume 1, pages 163–177. Elsevier, 1977.

[59] Thomas M Cover. *Elements of information theory*. John Wiley & Sons, 1999.

[60] Antonia Creswell, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A Bharath. Generative adversarial networks: An overview. *IEEE signal processing magazine*, 35(1):53–65, 2018.

[61] Florinel-Alin Croitoru, Vlad Hondru, Radu Tudor Ionescu, and Mubarak Shah. Diffusion models in vision: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

[62] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501*, 2018.

[63] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 702–703, 2020.

[64] W. H. Cunningham. Decomposition of submodular functions. *Combinatorica*, 3(1):53–68, 1983.

[65] Damai Dai, Yutao Sun, Li Dong, Yaru Hao, Shuming Ma, Zhifang Sui, and Furu Wei. Why can gpt learn in-context? language models secretly perform gradient descent as meta-optimizers. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 4005–4019, 2023.

[66] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019.

[67] Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems*, 2022.

[68] Arnav M Das, Gantavya Bhatt, Lilly Kumari, Sahil Verma, and Jeff Bilmes. Cobra: Combinatorial retrieval augmentation for few-shot learning. *arXiv preprint arXiv:2412.17684*, 2024.

[69] Pradeep Dasigi, Kyle Lo, Iz Beltagy, Arman Cohan, Noah A Smith, and Matt Gardner. A dataset of information-seeking questions and answers anchored in research papers. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4599–4610, 2021.

[70] Matthias De Lange and Tinne Tuytelaars. Continual prototype evolution: Learning online from non-stationary data streams. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8250–8259, 2021.

[71] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[72] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[73] Alessio Devoto, Yu Zhao, Simone Scardapane, and Pasquale Minervini. A simple and effective l_2 norm-based strategy for kv cache compression. *arXiv preprint arXiv:2406.11430*, 2024.

[74] Terrance DeVries. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.

[75] William B. Dolan and Chris Brockett. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*, 2005.

[76] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. *arXiv preprint arXiv:1605.09782*, 2016.

[77] Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Baobao Chang, et al. A survey on in-context learning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 1107–1128, 2024.

[78] Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, and Zhifang Sui. A survey for in-context learning. *arXiv preprint arXiv:2301.00234*, 2022.

[79] Shichen Dong, Wen Cheng, Jiayu Qin, and Wei Wang. Qaq: Quality adaptive quantization for llm kv cache, 2024.

[80] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. Boosting adversarial attacks with momentum. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9185–9193, 2018.

[81] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[82] Haodong Duan, Jueqi Wei, Chonghua Wang, Hongwei Liu, Yixiao Fang, Songyang Zhang, Dahua Lin, and Kai Chen. Botchat: Evaluating llms' capabilities of having multi-turn dialogues. *arXiv preprint arXiv:2310.13650*, 2023.

[83] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[84] Sayna Ebrahimi, Franziska Meier, Roberto Calandra, Trevor Darrell, and Marcus Rohrbach. Adversarial continual learning. In *European Conference on Computer Vision*, pages 386–402. Springer, 2020.

[85] Pablo A Estévez, Michel Tesmer, Claudio A Perez, and Jacek M Zurada. Normalized mutual information feature selection. *IEEE Transactions on neural networks*, 20(2):189–201, 2009.

[86] Kawin Ethayarajh. How contextual are contextualized word representations? comparing the geometry of bert, elmo, and gpt-2 embeddings. *arXiv preprint arXiv:1909.00512*, 2019.

[87] Lijie Fan, Kaifeng Chen, Dilip Krishnan, Dina Katabi, Phillip Isola, and Yonglong Tian. Scaling laws of synthetic images for model training... for now. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7382–7392, 2024.

[88] Wenqi Fan, Yujuan Ding, Liangbo Ning, Shijie Wang, Hengyun Li, Dawei Yin, Tat-Seng Chua, and Qing Li. A survey on rag meeting llms: Towards retrieval-augmented large language models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 6491–6501, 2024.

[89] Lisa K Fazio and Elizabeth J Marsh. Correcting false memories. *Psychological science*, 21(6):801–803, 2010.

[90] Guy Feigenblat, Haggai Roitman, Odellia Boni, and David Konopnicki. Unsupervised query-focused multi-document summarization using the cross entropy method. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 961–964, 2017.

[91] François Fleuret. Fast binary feature selection with conditional mutual information. *Journal of Machine learning research*, 5(9), 2004.

[92] Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. *arXiv preprint arXiv:2010.01412*, 2020.

[93] Robert M French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135, 1999.

[94] Satoru Fujishige. *Submodular functions and optimization*. Elsevier, 2005.

[95] Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

[96] Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation, 12 2023.

[97] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2023.

[98] Leon A Gatys. A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576*, 2015.

[99] Adam Gazzaley and Anna C Nobre. Top-down modulation: bridging selective attention and working memory. *Trends in cognitive sciences*, 16(2):129–135, 2012.

[100] Suyu Ge, Yunan Zhang, Liyuan Liu, Minjia Zhang, Jiawei Han, and Jianfeng Gao. Model tells you what to discard: Adaptive kv cache compression for llms, 2023.

[101] Dan Gillick and Benoit Favre. A scalable global model for summarization. In *Proceedings of the Workshop on Integer Linear Programming for Natural Language Processing*, pages 10–18, 2009.

[102] In Gim, Guojun Chen, Seung-seob Lee, Nikhil Sarda, Anurag Khandelwal, and Lin Zhong. Prompt cache: Modular attention reuse for low-latency inference. *arXiv preprint arXiv:2311.04934*, 2023.

[103] Hongyu Gong, Yelong Shen, Dian Yu, Jianshu Chen, and Dong Yu. Recurrent chunking mechanisms for long-text machine reading comprehension. *arXiv preprint arXiv:2005.08056*, 2020.

[104] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[105] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[106] Andrew Guillory and Jeff Bilmes. Active semi-supervised learning using submodular functions. In *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence*, pages 274–282, 2011.

[107] Andrew Guillory and Jeff A Bilmes. Online submodular set cover, ranking, and repeated active learning. *Advances in neural information processing systems*, 24, 2011.

[108] Zonghao Guo, Ruyi Xu, Yuan Yao, Junbo Cui, Zanlin Ni, Chunjiang Ge, Tat-Seng Chua, Zhiyuan Liu, and Gao Huang. Llava-uhd: an lmm perceiving any aspect ratio and high-resolution images. In *European Conference on Computer Vision*, pages 390–406. Springer, 2025.

[109] Shivanshu Gupta, Matt Gardner, and Sameer Singh. Coverage-based example selection for in-context learning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 13924–13950, 2023.

[110] Michael Gygli, Helmut Grabner, and Luc Van Gool. Video summarization by learning submodular mixtures of objectives. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3090–3098, 2015.

[111] Chi Han, Ziqi Wang, Han Zhao, and Heng Ji. In-context learning of large language models explained as kernel regression. *arXiv preprint arXiv:2305.12766*, 2023.

[112] Xuejun Han and Yuhong Guo. Continual learning with dual regularizations. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 619–634. Springer, 2021.

[113] Bo He, Hengduo Li, Young Kyun Jang, Menglin Jia, Xuefei Cao, Ashish Shah, Abhinav Shrivastava, and Ser-Nam Lim. Ma-lmm: Memory-augmented large multimodal model for long-term video understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13504–13514, 2024.

[114] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[115] Zhanying He, Chun Chen, Jiajun Bu, Can Wang, Lijun Zhang, Deng Cai, and Xiaofei He. Document summarization based on data reconstruction. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

[116] Byeongho Heo, Minsik Lee, Sangdoo Yun, and Jin Young Choi. Knowledge distillation with adversarial samples supporting decision boundary. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3771–3778, 2019.

[117] Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6609–6625, 2020.

[118] Rein Houthooft, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. Vime: Variational information maximizing exploration. *Advances in neural information processing systems*, 29, 2016.

[119] Drew A Hudson and Christopher D Manning. Gqa: A new dataset for real-world visual reasoning and compositional question answering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6700–6709, 2019.

[120] Chip Huyen. *Designing machine learning systems.* " O'Reilly Media, Inc.", 2022.

[121] Sergey Ioffe. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[122] Hamish Ivison, Noah A Smith, Hannaneh Hajishirzi, and Pradeep Dasigi. Data-efficient finetuning using cross-task nearest neighbors. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 9036–9061, 2023.

[123] Rishabh Iyer and Jeff Bilmes. Submodular optimization with submodular cover and submodular knapsack constraints. In *Neural Information Processing Society (NeurIPS, formerly NIPS)*, Lake Tahoe, CA, December 2013.

[124] Rishabh Iyer, Stefanie Jegelka, and Jeff Bilmes. Fast semidifferential-based submodular function optimization: Extended version. In *ICML*, 2013.

[125] Rishabh Iyer, Ninad Khargoankar, Jeff Bilmes, and Himanshu Asanani. Submodular combinatorial information measures with applications in machine learning. In *Algorithmic Learning Theory*, pages 722–754. PMLR, 2021.

[126] Gautier Izacard and Édouard Grave. Leveraging passage retrieval with generative models for open domain question answering. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 874–880, 2021.

[127] Joel Jang, Seonghyeon Ye, and Minjoon Seo. Can large language models truly understand prompts? a case study with negated prompts. In *Transfer Learning for Natural Language Processing Workshop*, pages 52–62. PMLR, 2023.

[128] Huiqiang Jiang, Qianhui Wu, Xufang Luo, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. Longllmlingua: Accelerating and enhancing llms in long context scenarios via prompt compression. *arXiv preprint arXiv:2310.06839*, 2023.

[129] Pin Jiang and Yahong Han. Hierarchical variational network for user-diversified & query-focused video summarization. In *Proceedings of the 2019 on International Conference on Multimedia Retrieval*, pages 202–206, 2019.

[130] Xisen Jin, Arka Sadhu, Junyi Du, and Xiang Ren. Gradient-based editing of memory examples for online task-free continual learning. *Advances in Neural Information Processing Systems*, 34, 2021.

[131] Roy Jonker and Anton Volgenant. A shortest augmenting path algorithm for dense and sparse linear assignment problems. *Computing*, 38(4):325–340, 1987.

[132] Mikael Kågebäck, Olof Mogren, Nina Tahmasebi, and Devdatt Dubhashi. Extractive summarization using continuous vector space models. In *Proceedings of the 2nd Workshop on Continuous Vector Space Models and their Compositionality (CVSC)*, pages 31–39, 2014.

[133] Hao Kang, Qingru Zhang, Souvik Kundu, Geonhwa Jeong, Zaoxing Liu, Tushar Krishna, and Tuo Zhao. Gear: An efficient kv cache compression recipe for near-lossless generative inference of llm, 2024.

[134] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International Conference on Machine Learning*, pages 5156–5165. PMLR, 2020.

[135] Amirhossein Kazemnejad, Inkit Padhi, Karthikeyan Natesan Ramamurthy, Payel Das, and Siva Reddy. The impact of positional encoding on length generalization in transformers. *Advances in Neural Information Processing Systems*, 36, 2024.

[136] Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186, 2019.

[137] Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. Generalization through memorization: Nearest neighbor language models. *arXiv preprint arXiv:1911.00172*, 2019.

[138] Samir Khuller, Anna Moss, and Joseph Seffi Naor. The budgeted maximum coverage problem. *Information processing letters*, 70(1):39–45, 1999.

[139] Krishnateja Killamsetty, Sivasubramanian Durga, Ganesh Ramakrishnan, Abir De, and Rishabh Iyer. Grad-match: Gradient matching based data subset selection for efficient deep model training. In *International Conference on Machine Learning*, pages 5464–5474. PMLR, 2021.

[140] Krishnateja Killamsetty, Durga Sivasubramanian, Ganesh Ramakrishnan, and Rishabh Iyer. Glister: Generalization based data subset selection for efficient and robust learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 8110–8118, 2021.

[141] Chris Dongjoo Kim, Jinseo Jeong, and Gunhee Kim. Imbalanced continual learning with partitioning reservoir sampling. In *European Conference on Computer Vision*, pages 411–428. Springer, 2020.

[142] Brendan King and Jeffrey Flanigan. Diverse retrieval-augmented in-context learning for dialogue state tracking. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 5570–5585, 2023.

[143] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.

[144] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[145] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020.

[146] Hayato Kobayashi, Masaki Noguchi, and Taichi Yatsuka. Summarization based on embedding distributions. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 1984–1989, 2015.

[147] Abdullatif Köksal, Timo Schick, and Hinrich Schütze. Meal: Stable and active learning for few-shot prompting. *arXiv preprint arXiv:2211.08358*, 2022.

[148] Dan Kondratyuk, Lijun Yu, Xiuye Gu, José Lezama, Jonathan Huang, Grant Schindler, Rachel Hornung, Vighnesh Birodkar, Jimmy Yan, Ming-Chang Chiu, et al. Videopoet: A large language model for zero-shot video generation. *arXiv preprint arXiv:2312.14125*, 2023.

[149] Jannik Kossen, Tom Rainforth, and Yarin Gal. In-context learning in large language models learns label relationships but is not conventional learning. *arXiv preprint arXiv:2307.12375*, 2023.

[150] Suraj Kothawade, Anmol Mekala, D Chandra Sekhara Hetha Havya, Mayank Kothyari, Rishabh Iyer, Ganesh Ramakrishnan, and Preethi Jyothi. Ditto: Data-efficient and fair targeted subset selection for asr accent adaptation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5810–5822, 2023.

[151] Andreas Krause and Carlos Guestrin. *A note on the budgeted maximization of submodular functions*. Citeseer, 2005.

[152] Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009.

[153] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[154] Abhishek Kumar and Hal Daumé. Learning task grouping and overlap in multi-task learning. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, ICML'12, page 1723–1730, 2012.

[155] Lilly Kumari and Jeff Bilmes. Submodular span, with applications to conditional data summarization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 12344–12352, 2021.

[156] Lilly Kumari, Usama Bin Shafqat, and Nikhil Sarda. Retrieval augmented generation for dialog modeling. In *NeurIPS ENLSP Workshop*, 2023.

[157] Lilly Kumari, Shengjie Wang, Arnav Das, Tianyi Zhou, and Jeff Bilmes. An end-to-end submodular framework for data-efficient in-context learning. In *Findings of the Association for Computational Linguistics: NAACL 2022*, 2024.

[158] Lilly Kumari, Shengjie Wang, Tianyi Zhou, and Jeff A Bilmes. Retrospective adversarial replay for continual learning. *Advances in Neural Information Processing Systems*, 35:28530–28544, 2022.

[159] Lilly Kumari, Shengjie Wang, Tianyi Zhou, Nikhil Sarda, Anthony Rowe, and Jeff Bilmes. Bumblebee: Dynamic kv-cache streaming submodular summarization for infinite-context transformers. In *First Conference on Language Modeling*, 2024.

[160] Po-Nien Kung and Nanyun Peng. Do models really learn to follow instructions? an empirical study of instruction tuning. *arXiv preprint arXiv:2305.11383*, 2023.

[161] Alexey Kurakin, Ian Goodfellow, Samy Bengio, et al. Adversarial examples in the physical world, 2016.

[162] Litton J Kurisinkel and Nancy F. Chen. Llm based multi-document summarization exploiting main-event biased monotone submodular content extraction, 2023.

[163] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, SOSP '23, page 611–626, New York, NY, USA, 2023. Association for Computing Machinery.

[164] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

[165] Hugo Laurençon, Lucile Saulnier, Thomas Wang, Christopher Akiki, Albert Villanova del Moral, Teven Le Scao, Leandro Von Werra, Chenghao Mou, Eduardo González Ponferrada, Huu Nguyen, et al. The bigscience roots corpus: A 1.6 tb composite multilingual dataset. *Advances in Neural Information Processing Systems*, 35:31809–31826, 2022.

[166] Chandrashekhar Lavania, Kai Wei, Rishabh Iyer, and Jeff Bilmes. A practical online framework for extracting running video summaries under a fixed memory budget. In *Proceedings of the 2021 SIAM International Conference on Data Mining (SDM)*, pages 226–234. SIAM, 2021.

[167] Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

[168] Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. Deduplicating training data makes language models better. *arXiv preprint arXiv:2107.06499*, 2021.

[169] Sang-Woo Lee, Jin-Hwa Kim, Jaehyun Jun, Jung-Woo Ha, and Byoung-Tak Zhang. Overcoming catastrophic forgetting by incremental moment matching. *Advances in neural information processing systems*, 30, 2017.

[170] Soochan Lee, Junsoo Ha, Dongsu Zhang, and Gunhee Kim. A neural dirichlet process mixture model for task-free continual learning. *arXiv preprint arXiv:2001.00689*, 2020.

[171] Yong Jae Lee, Joydeep Ghosh, and Kristen Grauman. Discovering important people and objects for egocentric video summarization. In *2012 IEEE conference on computer vision and pattern recognition*, pages 1346–1353. IEEE, 2012.

[172] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, et al. Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic web*, 6(2):167–195, 2015.

[173] Yaniv Leviathan, Matan Kalman, and Yossi Matias. Selective attention improves transformer. *arXiv preprint arXiv:2410.02703*, 2024.

[174] Itay Levy, Ben Bogin, and Jonathan Berant. Diverse demonstrations improve in-context compositional generalization. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1401–1422, 2023.

[175] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*, 2019.

[176] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474, 2020.

[177] Canjia Li, Andrew Yates, Sean MacAvaney, Ben He, and Yingfei Sun. Parade: Passage representation aggregation for document reranking. *arXiv preprint arXiv:2008.09093*, 2020.

[178] Dacheng Li, Rulin Shao, Anze Xie, Ying Sheng, Lianmin Zheng, Joseph Gonzalez, Ion Stoica, Xuezhe Ma, and Hao Zhang. How long can context length of open-source LLMs truly promise? In *NeurIPS 2023 Workshop on Instruction Tuning and Instruction Following*, 2023.

[179] Huayang Li, Yixuan Su, Deng Cai, Yan Wang, and Lemao Liu. A survey on retrieval-augmented text generation. *arXiv preprint arXiv:2202.01110*, 2022.

[180] Jiatao Li, Xinyu Hu, and Xiaojun Wan. Smart-rag: Selection using determinantal matrices for augmented retrieval. *arXiv preprint arXiv:2409.13992*, 2024.

[181] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International conference on machine learning*, pages 19730–19742. PMLR, 2023.

[182] Xiaonan Li and Xipeng Qiu. Finding supporting examples for in-context learning. *arXiv preprint arXiv:2302.13539*, 2023.

[183] Xilai Li, Yingbo Zhou, Tianfu Wu, Richard Socher, and Caiming Xiong. Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting. In *International Conference on Machine Learning*, pages 3925–3934. PMLR, 2019.

[184] Xin Li and Dan Roth. Learning question classifiers. In *COLING 2002: The 19th International Conference on Computational Linguistics*, 2002.

[185] Xin Li and Dan Roth. Learning question classifiers. In *COLING 2002: The 19th International Conference on Computational Linguistics*, 2002.

[186] Xin Li, Yunfei Wu, Xinghua Jiang, Zhihao Guo, Mingming Gong, Haoyu Cao, Yinsong Liu, Deqiang Jiang, and Xing Sun. Enhancing visual document understanding with contrastive learning in large visual-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15546–15555, 2024.

[187] Yifan Li, Yifan Du, Kun Zhou, Jinpeng Wang, Wayne Xin Zhao, and Ji-Rong Wen. Evaluating object hallucination in large vision-language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 292–305, 2023.

[188] Yingcong Li, Muhammed Emrullah Ildiz, Dimitris Papailiopoulos, and Samet Oymak. Transformers as algorithms: Generalization and stability in in-context learning. In *International Conference on Machine Learning*, pages 19565–19594. PMLR, 2023.

[189] Yuanzhi Li, Sébastien Bubeck, Ronen Eldan, Allie Del Giorno, Suriya Gunasekar, and Yin Tat Lee. Textbooks are all you need ii: phi-1.5 technical report. *arXiv preprint arXiv:2309.05463*, 2023.

[190] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.

[191] Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, et al. Holistic evaluation of language models. *arXiv preprint arXiv:2211.09110*, 2022.

[192] Bin Lin, Yang Ye, Bin Zhu, Jiaxi Cui, Munan Ning, Peng Jin, and Li Yuan. Video-LLaVA: Learning united visual representation by alignment before projection. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 5971–5984, Miami, Florida, USA, November 2024. Association for Computational Linguistics.

[193] Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[194] Hui Lin and Jeff Bilmes. Multi-document summarization via budgeted maximization of submodular functions. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 912–920, 2010.

[195] Hui Lin and Jeff Bilmes. A class of submodular functions for document summarization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 510–520. Association for Computational Linguistics, 2011.

[196] Hui Lin and Jeff Bilmes. Learning mixtures of submodular shells with application to document summarization. In *Proceedings of the Twenty-Eighth Conference on Uncertainty in Artificial Intelligence*, pages 479–490, 2012.

[197] Hao Liu, Matei Zaharia, and Pieter Abbeel. Ringattention with blockwise transformers for near-infinite context. In *The Twelfth International Conference on Learning Representations*, 2024.

[198] Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 26296–26306, 2024.

[199] Haotian Liu, Chunyuan Li, Yuheng Li, Bo Li, Yuanhan Zhang, Sheng Shen, and Yong Jae Lee. Llava-next: Improved reasoning, ocr, and world knowledge, 2024.

[200] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36, 2024.

[201] Jiachang Liu, Dinghan Shen, Yizhe Zhang, William B Dolan, Lawrence Carin, and Weizhu Chen. What makes good in-context examples for gpt-3? In *Proceedings of Deep Learning Inside Out (DeeLIO 2022): The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 100–114, 2022.

[202] Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *arXiv preprint arXiv:2307.03172*, 2023.

[203] Peng Liu, Lizhe Wang, Rajiv Ranjan, Guojin He, and Lei Zhao. A survey on active deep learning: from model driven to data driven. *ACM Computing Surveys (CSUR)*, 54(10s):1–34, 2022.

[204] Ruibo Liu, Jerry Wei, Fangyu Liu, Chenglei Si, Yanzhe Zhang, Jinmeng Rao, Steven Zheng, Daiyi Peng, Diyi Yang, Denny Zhou, et al. Best practices and lessons learned on synthetic data for language models. *arXiv preprint arXiv:2404.07503*, 2024.

[205] Yuliang Liu, Biao Yang, Qiang Liu, Zhang Li, Zhiyin Ma, Shuo Zhang, and Xiang Bai. Textmonkey: An ocr-free large multimodal model for understanding document. *arXiv preprint arXiv:2403.04473*, 2024.

[206] Yuzong Liu, Kai Wei, Katrin Kirchhoff, Yisong Song, and Jeff Bilmes. Submodular feature selection for high-dimensional acoustic score spaces. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 7184–7188. IEEE, 2013.

[207] Zichang Liu, Aditya Desai, Fangshuo Liao, Weitao Wang, Victor Xie, Zhaozhuo Xu, Anastasios Kyrillidis, and Anshumali Shrivastava. Scissorhands: Exploiting the persistence of importance hypothesis for llm kv cache compression at test time. *arXiv preprint arXiv:2305.17118*, 2023.

[208] Elizabeth F Loftus and Jacqueline E Pickrell. The formation of false memories, 1995.

[209] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30:6467–6476, 2017.

[210] I Loshchilov. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[211] Pan Lu, Swaroop Mishra, Tanglin Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. Learn to explain: Multimodal reasoning via thought chains for science question answering. *Advances in Neural Information Processing Systems*, 35:2507–2521, 2022.

[212] Sheng Lu, Irina Bigoulaeva, Rachneet Sachdeva, Harish Tayyar Madabushi, and Iryna Gurevych. Are emergent abilities in large language models just in-context learning? *arXiv preprint arXiv:2309.01809*, 2023.

[213] Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8086–8098, 2022.

[214] Yingzhou Lu, Minjie Shen, Huazheng Wang, Xiao Wang, Capucine van Rechem, and Wenqi Wei. Machine learning for synthetic data generation: a review. *arXiv preprint arXiv:2302.04062*, 2023.

[215] Gen Luo, Yiyi Zhou, Yuxin Zhang, Xiawu Zheng, Xiaoshuai Sun, and Rongrong Ji. Feast your eyes: Mixture-of-resolution adaptation for multimodal large language models. *arXiv preprint arXiv:2403.03003*, 2024.

[216] Shulei Ma, Zhi-Hong Deng, and Yunlun Yang. An unsupervised multi-document summarization framework based on neural document model. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1514–1523, 2016.

[217] Muhammad Maaz, Hanoona Rasheed, Salman Khan, and Fahad Shahbaz Khan. Video-chatgpt: Towards detailed video understanding via large vision and language models. *arXiv preprint arXiv:2306.05424*, 2023.

[218] Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating very long-term conversational memory of llm agents. *arXiv preprint arXiv:2402.17753*, 2024.

[219] Zheda Mai, Ruiwen Li, Hyunwoo Kim, and Scott Sanner. Supervised contrastive replay: Revisiting the nearest class mean classifier in online class-incremental continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3589–3599, 2021.

[220] Katerina Margatina, Timo Schick, Nikolaos Aletras, and Jane Dwivedi-Yu. Active learning principles for in-context learning with large language models. *arXiv preprint arXiv:2305.14264*, 2023.

[221] Max Marion, Ahmet Üstün, Luiza Pozzobon, Alex Wang, Marzieh Fadaee, and Sara Hooker. When less is more: Investigating data pruning for pretraining llms at scale. *arXiv preprint arXiv:2309.04564*, 2023.

[222] Pedro Henrique Martins, Zita Marinho, and André FT Martins. ∞-former: Infinite memory transformer. *arXiv preprint arXiv:2109.00301*, 2021.

[223] Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier, 1989.

[224] Ryan McDonald. A study of global inference algorithms in multi-document summarization. In *European Conference on Information Retrieval*, pages 557–564. Springer, 2007.

[225] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models, 2016.

[226] Thomas Merth, Qichen Fu, Mohammad Rastegari, and Mahyar Najibi. Superposition prompting: Improving and accelerating retrieval-augmented generation. In *Forty-first International Conference on Machine Learning*, 2024.

[227] Fei Mi, Lingjing Kong, Tao Lin, Kaicheng Yu, and Boi Faltings. Generalized class incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 240–241, 2020.

[228] Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391, 2018.

[229] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[230] Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. Rethinking the role of demonstrations: What makes in-context learning work? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11048–11064, 2022.

[231] Michel Minoux. Accelerated greedy algorithms for maximizing submodular set functions. In *Optimization Techniques: Proceedings of the 8th IFIP Conference on Optimization Techniques Würzburg, September 5–9, 1977*, pages 234–243. Springer, 1978.

[232] Pitu B Mirchandani and Richard L Francis. *Discrete location theory*. Wiley, 1990.

[233] Seyed Iman Mirzadeh, Mehrdad Farajtabar, Razvan Pascanu, and Hassan Ghasemzadeh. Understanding the role of training regimes in continual learning. *Advances in Neural Information Processing Systems*, 33:7308–7320, 2020.

[234] Baharan Mirzasoleiman, Ashwinkumar Badanidiyuru, Amin Karbasi, Jan Vondrák, and Andreas Krause. Lazier than lazy greedy. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015.

[235] Baharan Mirzasoleiman, Jeff Bilmes, and Jure Leskovec. Coresets for data-efficient training of machine learning models. In *International Conference on Machine Learning*, pages 6950–6960. PMLR, 2020.

[236] Jesse Mu, Xiang Lisa Li, and Noah Goodman. Learning to compress prompts with gist tokens, 2024.

[237] Preetum Nakkiran, Gal Kaplun, Yamini Bansal, Tristan Yang, Boaz Barak, and Ilya Sutskever. Deep double descent: Where bigger models and more data hurt. *Journal of Statistical Mechanics: Theory and Experiment*, 2021(12):124003, 2021.

[238] Shashi Narayan, Shay B Cohen, and Mirella Lapata. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807, 2018.

[239] George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. An analysis of approximations for maximizing submodular set functions—i. *Mathematical programming*, 14(1):265–294, 1978.

[240] Eryn J Newman and D Stephen Lindsay. False memories: What the hell are they for? *Applied Cognitive Psychology: The Official Journal of the Society for Applied Research in Memory and Cognition*, 23(8):1105–1121, 2009.

[241] Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nati Srebro. Exploring generalization in deep learning. *Advances in neural information processing systems*, 30, 2017.

[242] Cuong V. Nguyen, Yingzhen Li, Thang D. Bui, and Richard E. Turner. Variational continual learning. In *International Conference on Learning Representations*, 2018.

[243] Curtis G Northcutt, Anish Athalye, and Jonas Mueller. Pervasive label errors in test sets destabilize machine learning benchmarks. *arXiv preprint arXiv:2103.14749*, 2021.

[244] Team OLMo. 2 olmo 2 furious, 2024.

[245] OpenAI. Gpt-4 technical report, 2023.

[246] J.G. Oxley. *Matroid Theory: Second Edition*. Oxford University Press, 2011.

[247] Pingbo Pan, Siddharth Swaroop, Alexander Immer, Runa Eschenhagen, Richard Turner, and Mohammad Emtiyaz E Khan. Continual deep learning by functional regularisation of memorable past. *Advances in Neural Information Processing Systems*, 33:4453–4464, 2020.

[248] Guilherme Penedo, Hynek Kydlíček, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, Thomas Wolf, et al. The fineweb datasets: Decanting the web for the finest text data at scale. *arXiv preprint arXiv:2406.17557*, 2024.

[249] Hanchuan Peng, Fuhui Long, and Chris Ding. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on pattern analysis and machine intelligence*, 27(8):1226–1238, 2005.

[250] Anastasia Pentina and Christoph H Lampert. Lifelong learning with non-iid tasks. *Advances in Neural Information Processing Systems*, 28:1540–1548, 2015.

[251] Matthew Peters, Sebastian Ruder, and Noah A Smith. To tune or not to tune? adapting pretrained representations to diverse tasks. *arXiv preprint arXiv:1903.05987*, 2019.

[252] Marc Pickett, Jeremy Hartman, Ayan Kumar Bhowmick, Raquib-ul Alam, and Aditya Vempaty. Better rag using relevant information gain. *arXiv preprint arXiv:2407.12101*, 2024.

[253] Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. Efficiently scaling transformer inference. *Proceedings of Machine Learning and Systems*, 5, 2023.

[254] Abhilash Potluri, Fangyuan Xu, and Eunsol Choi. Concise answers to complex questions: Summarization of long-form answers. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9709–9728, 2023.

[255] Samira Pouyanfar, Saad Sadiq, Yilin Yan, Haiman Tian, Yudong Tao, Maria Presa Reyes, Mei-Ling Shyu, Shu-Ching Chen, and Sundaraja S Iyengar. A survey on deep learning: Algorithms, techniques, and applications. *ACM computing surveys (CSUR)*, 51(5):1–36, 2018.

[256] Hang Qi, Evan R. Sparks, and Ameet Talwalkar. Paleo: A performance model for deep neural networks. In *International Conference on Learning Representations*, 2017.

[257] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[258] Alec Radford and Karthik Narasimhan. Improving language understanding by generative pre-training. 2018.

[259] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[260] Jack W Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, et al. Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint arXiv:2112.11446*, 2021.

[261] Jack W Rae, Anna Potapenko, Siddhant M Jayakumar, and Timothy P Lillicrap. Compressive transformers for long-range sequence modelling. *arXiv preprint arXiv:1911.05507*, 2019.

[262] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(1), jan 2020.

[263] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

[264] Ori Ram, Yoav Levine, Itay Dalmedigos, Dor Muhlgay, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. In-context retrieval-augmented language models. *arXiv preprint arXiv:2302.00083*, 2023.

[265] Vinay V Ramasesh, Ethan Dyer, and Maithra Raghu. Anatomy of catastrophic forgetting: Hidden representations and task semantics. *arXiv preprint arXiv:2007.07400*, 2020.

[266] Alexander Ratner, Stephen H Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré. Snorkel: Rapid training data creation with weak supervision. In *Proceedings of the VLDB endowment. International conference on very large data bases*, volume 11, page 269. NIH Public Access, 2017.

[267] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2001–2010, 2017.

[268] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do imagenet classifiers generalize to imagenet? In *International conference on machine learning*, pages 5389–5400. PMLR, 2019.

[269] Machel Reid, Nikolay Savinov, Denis Teplyashin, Dmitry Lepikhin, Timothy Lillicrap, Jean-baptiste Alayrac, Radu Soricut, Angeliki Lazaridou, Orhan Firat, Julian Schrittwieser, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024.

[270] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, 2019.

[271] Pengjie Ren, Zhumin Chen, Zhaochun Ren, Furu Wei, Liqiang Nie, Jun Ma, and Maarten De Rijke. Sentence relations for extractive summarization with deep neural networks. *ACM Transactions on Information Systems (TOIS)*, 36(4):1–32, 2018.

[272] H S V N S Kowndinya Renduchintala, Sumit Bhatia, and Ganesh Ramakrishnan. Smart: Submodular data mixture strategy for instruction tuning, 2024.

[273] H S V N S Kowndinya Renduchintala, Krishnateja Killamsetty, Sumit Bhatia, Milan Aggarwal, Ganesh Ramakrishnan, Rishabh Iyer, and Balaji Krishnamurthy. Ingenious: Using informative data subsets for efficient pre-training of language models, 2023.

[274] Luka Ribar, Ivan Chelombiev, Luke Hudlass-Galley, Charlie Blake, Carlo Luschi, and Douglas Orr. Sparq attention: Bandwidth-efficient llm inference, 2024.

[275] Matthew Riemer, Ignacio Cases, Robert Ajemian, Miao Liu, Irina Rish, Yuhai Tu, , and Gerald Tesauro. Learning to learn without forgetting by maximizing transfer and minimizing interference. In *International Conference on Learning Representations*, 2019.

[276] Anthony Robins. Catastrophic forgetting, rehearsal and pseudorehearsal. *Connection Science*, 7(2):123–146, 1995.

[277] Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S Gordon. Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In *2011 AAAI Spring Symposium Series*, 2011.

[278] Haggai Roitman, Guy Feigenblat, Doron Cohen, Odellia Boni, and David Konopnicki. Unsupervised dual-cascade learning with pseudo-feedback distillation for query-focused extractive summarization. In *Proceedings of The Web Conference 2020*, pages 2577–2584, 2020.

[279] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience replay for continual learning. *Advances in Neural Information Processing Systems*, 32, 2019.

[280] Andrew Rosenberg and Julia Hirschberg. V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*, pages 410–420, 2007.

[281] Mohammad Rostami, Soheil Kolouri, and Praveen K Pilly. Complementary learning for overcoming catastrophic forgetting using experience replay. *arXiv preprint arXiv:1903.04566*, 2019.

[282] Ohad Rubin, Jonathan Herzig, and Jonathan Berant. Learning to retrieve prompts for in-context learning. *arXiv preprint arXiv:2112.08633*, 2021.

[283] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

[284] Siddharth Samsi, Dan Zhao, Joseph McDonald, Baolin Li, Adam Michaleas, Michael Jones, William Bergeron, Jeremy Kepner, Devesh Tiwari, and Vijay Gadepally. From words to watts: Benchmarking the energy costs of large language model inference. In *2023 IEEE High Performance Extreme Computing Conference (HPEC)*, pages 1–9. IEEE, 2023.

[285] Nikhil Sardana, Jacob Portes, Sasha Doubov, and Jonathan Frankle. Beyond chinchilla-optimal: Accounting for inference in language model scaling laws. *arXiv preprint arXiv:2401.00448*, 2023.

[286] Lars Schmarje, Vasco Grossmann, Claudius Zelenka, Sabine Dippel, Rainer Kiko, Mariusz Oszust, Matti Pastell, Jenny Stracke, Anna Valros, Nina Volkmann, et al. Is one annotation enough?-a data-centric image classification benchmark for noisy and ambiguous label estimation. *Advances in Neural Information Processing Systems*, 35:33215–33232, 2022.

[287] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. Laion-5b: An open large-scale dataset for training next generation image-text models. *Advances in Neural Information Processing Systems*, 35:25278–25294, 2022.

[288] Jonathan Schwarz, Wojciech Czarnecki, Jelena Luketina, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress & compress: A scalable framework for continual learning. In *International Conference on Machine Learning*, pages 4528–4537. PMLR, 2018.

[289] Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. *arXiv preprint arXiv:1708.00489*, 2017.

[290] Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In *International Conference on Machine Learning*, pages 4548–4557. PMLR, 2018.

[291] Burr Settles. Active learning literature survey. *Machine Learning*, 15(2):201–221, 1994.

[292] Burr Settles. From theories to queries: Active learning in practice. In *Active learning and experimental design workshop in conjunction with AISTATS 2010*, pages 1–18. JMLR Workshop and Conference Proceedings, 2011.

[293] Burr Settles, Mark Craven, and Lewis Friedland. Active learning with real annotation costs. In *Proceedings of the NIPS workshop on cost-sensitive learning*, volume 1. Vancouver, CA:, 2008.

[294] Dhruv Shah, Błażej Osiński, Sergey Levine, et al. Lm-nav: Robotic navigation with large pre-trained models of language, vision, and action. In *Conference on robot learning*, pages 492–504. PMLR, 2023.

[295] Aidean Sharghi, Boqing Gong, and Mubarak Shah. Query-focused extractive video summarization. In *European Conference on Computer Vision*, pages 3–19. Springer, 2016.

[296] Aidean Sharghi, Jacob S Laurel, and Boqing Gong. Query-focused video summarization: Dataset, evaluation, and a memory network based approach. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4788–4797, 2017.

[297] Noam Shazeer. Fast transformer decoding: One write-head is all you need, 2019.

[298] Ying Sheng, Lianmin Zheng, Binhang Yuan, Zhuohan Li, Max Ryabinin, Beidi Chen, Percy Liang, Christopher Re, Ion Stoica, and Ce Zhang. Flexgen: High-throughput generative inference of large language models with a single gpu. *ICML*, 2023.

[299] Dachuan Shi, Chaofan Tao, Ying Jin, Zhendong Yang, Chun Yuan, and Jiaqi Wang. Upop: Unified and progressive pruning for compressing vision-language transformers. In *International Conference on Machine Learning*, pages 31292–31311. PMLR, 2023.

[300] Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed H Chi, Nathanael Schärli, and Denny Zhou. Large language models can be easily distracted by irrelevant context. In *International Conference on Machine Learning*, pages 31210–31227. PMLR, 2023.

[301] Dongsub Shim, Zheda Mai, Jihwan Jeong, Scott Sanner, Hyunwoo Kim, and Jongseong Jang. Online class-incremental continual learning with adversarial shapley value. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 9630–9638, 2021.

[302] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. *arXiv preprint arXiv:1705.08690*, 2017.

[303] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of big data*, 6(1):1–48, 2019.

[304] Kurt Shuster, Spencer Poff, Moya Chen, Douwe Kiela, and Jason Weston. Retrieval augmentation reduces hallucination in conversation. *arXiv preprint arXiv:2104.07567*, 2021.

[305] Amanpreet Singh, Vivek Natarajan, Meet Shah, Yu Jiang, Xinlei Chen, Dhruv Batra, Devi Parikh, and Marcus Rohrbach. Towards vqa models that can read. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8317–8326, 2019.

[306] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642, 2013.

[307] Gowthami Somepalli, Liam Fowl, Arpit Bansal, Ping Yeh-Chiang, Yehuda Dar, Richard Baraniuk, Micah Goldblum, and Tom Goldstein. Can neural nets learn the same model twice? investigating reproducibility and double descent from the decision boundary perspective. *arXiv preprint arXiv:2203.08124*, 2022.

[308] Enxin Song, Wenhao Chai, Guanhong Wang, Yucheng Zhang, Haoyang Zhou, Feiyang Wu, Haozhe Chi, Xun Guo, Tian Ye, Yanting Zhang, et al. Moviechat: From dense token to sparse memory for long video understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18221–18232, 2024.

[309] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[310] Susanne Still and Doina Precup. An information-theoretic approach to curiosity-driven reinforcement learning. *Theory in Biosciences*, 131:139–148, 2012.

[311] Hongjin Su, Jungo Kasai, Chen Henry Wu, Weijia Shi, Tianlu Wang, Jiayi Xin, Rui Zhang, Mari Ostendorf, Luke Zettlemoyer, Noah A Smith, et al. Selective annotation makes language models better few-shot learners. *arXiv preprint arXiv:2209.01975*, 2022.

[312] Z. Svitkina and L. Fleischer. Submodular approximation: Sampling-based algorithms and lower bounds. In *FOCS*, pages 697–706, 2008.

[313] Zoya Svitkina and Lisa Fleischer. Submodular approximation: Sampling-based algorithms and lower bounds. *SIAM Journal on Computing*, 40(6):1715–1737, 2011.

[314] Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024.

[315] Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, et al. Lamda: Language models for dialog applications. *arXiv preprint arXiv:2201.08239*, 2022.

[316] Sunil Thulasidasan, Gopinath Chennupati, Jeff A Bilmes, Tanmoy Bhattacharya, and Sarah Michalak. On mixup training: Improved calibration and predictive uncertainty for deep neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

[317] Kushal Tirumala, Daniel Simig, Armen Aghajanyan, and Ari Morcos. D4: Improving llm pretraining via document de-duplication and diversification. *Advances in Neural Information Processing Systems*, 36:53983–53995, 2023.

[318] Naftali Tishby and Daniel Polani. Information theory of decisions and actions. In *Perception-action cycle: Models, architectures, and hardware*, pages 601–636. Springer, 2010.

[319] Michalis K. Titsias, Jonathan Schwarz, Alexander G. de G. Matthews, Razvan Pascanu, and Yee Whye Teh. Functional regularisation for continual learning with gaussian processes. In *International Conference on Learning Representations*, 2020.

[320] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023.

[321] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[322] Ivor W Tsang, James T Kwok, Pak-Ming Cheung, and Nello Cristianini. Core vector machines: Fast svm training on very large data sets. *Journal of Machine Learning Research*, 6(4), 2005.

[323] Sebastian Tschiatschek, Rishabh K Iyer, Haochen Wei, and Jeff A Bilmes. Learning mixtures of submodular functions for image collection summarization. In *Advances in neural information processing systems*, pages 1413–1421, 2014.

[324] Melina R Uncapher and Michael D Rugg. Selecting for memory? the influence of selective attention on the mnemonic binding of contextual information. *Journal of Neuroscience*, 29(25):8270–8279, 2009.

[325] Guillermo Valle-Perez, Chico Q Camargo, and Ard A Louis. Deep learning generalizes because the parameter-function map is biased towards simple functions. In *International Conference on Learning Representations*, 2019.

[326] Gido M Van de Ven and Andreas S Tolias. Generative replay with feedback connections as a general strategy for continual learning. *arXiv preprint arXiv:1809.10635*, 2018.

[327] Gido M Van de Ven and Andreas S Tolias. Three scenarios for continual learning. *arXiv preprint arXiv:1904.07734*, 2019.

[328] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[329] Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, David Lopez-Paz, and Yoshua Bengio. Manifold mixup: Better representations by interpolating hidden states. In *International conference on machine learning*, pages 6438–6447. PMLR, 2019.

[330] Jeffrey S Vitter. Random sampling with a reservoir. *ACM Transactions on Mathematical Software (TOMS)*, 11(1):37–57, 1985.

[331] Johannes Von Oswald, Eyvind Niklasson, Ettore Randazzo, João Sacramento, Alexander Mordvintsev, Andrey Zhmoginov, and Max Vladymyrov. Transformers learn in-context by gradient descent. In *International Conference on Machine Learning*, pages 35151–35174. PMLR, 2023.

[332] Xiaojun Wan and Jianmin Zhang. Ctsum: extracting more certain summaries for news articles. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, pages 787–796, 2014.

[333] Zhongwei Wan, Xin Wang, Che Liu, Samiul Alam, Yu Zheng, Jiachen Liu, Zhongnan Qu, Shen Yan, Yi Zhu, Quanlu Zhang, et al. Efficient large language models: A survey. *Transactions on Machine Learning Research*, 2024.

[334] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, 2018.

[335] Ben Wang and Aran Komatsuzaki. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. https://github.com/kingoflolz/mesh-transformer-jax, May 2021.

[336] Liyuan Wang, Xingxing Zhang, Kuo Yang, Longhui Yu, Chongxuan Li, Lanqing HONG, Shifeng Zhang, Zhenguo Li, Yi Zhong, and Jun Zhu. Memory replay with

data compression for continual learning. In *International Conference on Learning Representations*, 2022.

[337] Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Yang Fan, Kai Dang, Mengfei Du, Xuancheng Ren, Rui Men, Dayiheng Liu, Chang Zhou, Jingren Zhou, and Junyang Lin. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*, 2024.

[338] Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

[339] Yumeng Wang and Zhenyang Xiao. Loma: Lossless compressed memory attention, 2024.

[340] Zaitian Wang, Pengfei Wang, Kunpeng Liu, Pengyang Wang, Yanjie Fu, Chang-Tien Lu, Charu C Aggarwal, Jian Pei, and Yuanchun Zhou. A comprehensive survey on data augmentation. *arXiv preprint arXiv:2405.09591*, 2024.

[341] Zhicheng Wang, Yufang Liu, Tao Ji, Xiaoling Wang, Yuanbin Wu, Congcong Jiang, Ye Chao, Zhencong Han, Ling Wang, Xu Shao, et al. Rehearsal-free continual language learning via efficient parameter isolation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10933–10946, 2023.

[342] Zhiruo Wang, Jun Araki, Zhengbao Jiang, Md Rizwan Parvez, and Graham Neubig. Learning to filter context for retrieval-augmented generation. *arXiv preprint arXiv:2311.08377*, 2023.

[343] Albert Webson and Ellie Pavlick. Do prompt-based models really understand the meaning of their prompts? In Marine Carpuat, Marie-Catherine de Marneffe, and Ivan Vladimir Meza Ruiz, editors, *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2300–2344, Seattle, United States, July 2022. Association for Computational Linguistics.

[344] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. Emergent abilities of large language models. *Transactions on Machine Learning Research*, 2022.

[345] Jerry Wei, Jason Wei, Yi Tay, Dustin Tran, Albert Webson, Yifeng Lu, Xinyun Chen, Hanxiao Liu, Da Huang, Denny Zhou, et al. Larger language models do in-context learning differently. *arXiv preprint arXiv:2303.03846*, 2023.

[346] Kai Wei, Rishabh Iyer, and Jeff Bilmes. Submodularity in data subset selection and active learning. In *International conference on machine learning*, pages 1954–1963. PMLR, 2015.

[347] Kai Wei, Yuzong Liu, Katrin Kirchhoff, Chris Bartels, and Jeff Bilmes. Submodular subset selection for large-scale speech training data. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3311–3315. IEEE, 2014.

[348] Kai Wei, Yuzong Liu, Katrin Kirchhoff, and Jeff Bilmes. Unsupervised submodular subset selection for speech data. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4107–4111. IEEE, 2014.

[349] Yuxin Wen, Qingqing Cao, Qichen Fu, Sachin Mehta, and Mahyar Najibi. Efficient vision-language models by summarizing visual tokens into compact registers. *arXiv preprint arXiv:2410.14072*, 2024.

[350] Likang Wu, Zhi Zheng, Zhaopeng Qiu, Hao Wang, Hongchao Gu, Tingjia Shen, Chuan Qin, Chen Zhu, Hengshu Zhu, Qi Liu, Hui Xiong, and Enhong Chen. A survey on large language models for recommendation, 2023.

[351] Yuhuai Wu, Markus N Rabe, DeLesley Hutchins, and Christian Szegedy. Memorizing transformers. *arXiv preprint arXiv:2203.08913*, 2022.

[352] Zhiyong Wu, Yaoxiang Wang, Jiacheng Ye, and Lingpeng Kong. Self-adaptive in-context learning: An information compression perspective for in-context example selection and ordering, 2023.

[353] Mengzhou Xia, Sadhika Malladi, Suchin Gururangan, Sanjeev Arora, and Danqi Chen. Less: Selecting influential data for targeted instruction tuning. *arXiv preprint arXiv:2402.04333*, 2024.

[354] Guangxuan Xiao, Jiaming Tang, Jingwei Zuo, junxian guo, Shang Yang, Haotian Tang, Yao Fu, and Song Han. Duoattention: Efficient long-context LLM inference with retrieval and streaming heads. In *The Thirteenth International Conference on Learning Representations*, 2025.

[355] Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. *arXiv preprint arXiv:2309.17453*, 2023.

[356] Shuwen Xiao, Zhou Zhao, Zijian Zhang, Ziyu Guan, and Deng Cai. Query-biased self-attentive network for query-focused video summarization. *IEEE Transactions on Image Processing*, 29:5889–5899, 2020.

[357] Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. An explanation of in-context learning as implicit bayesian inference. In *International Conference on Learning Representations*, 2021.

[358] Jing Xu, Arthur Szlam, and Jason Weston. Beyond goldfish memory: Long-term open-domain conversation. *arXiv preprint arXiv:2107.07567*, 2021.

[359] Yilun Xu, Peng Cao, Yuqing Kong, and Yizhou Wang. L_dmi: A novel information-theoretic loss function for training deep nets robust to label noise. *Advances in neural information processing systems*, 32, 2019.

[360] Jingfeng Yang, Hongye Jin, Ruixiang Tang, Xiaotian Han, Qizhang Feng, Haoming Jiang, Shaochen Zhong, Bing Yin, and Xia Hu. Harnessing the power of llms in practice: A survey on chatgpt and beyond. *ACM Transactions on Knowledge Discovery from Data*, 18(6):1–32, 2024.

[361] Senqiao Yang, Yukang Chen, Zhuotao Tian, Chengyao Wang, Jingyao Li, Bei Yu, and Jiaya Jia. Visionzip: Longer is better but not necessary in vision language models. *arXiv preprint arXiv:2412.04467*, 2024.

[362] Shuo Yang, Zeke Xie, Hanyu Peng, Min Xu, Mingming Sun, and Ping Li. Dataset pruning: Reducing training data by examining generalization influence. In *The Eleventh International Conference on Learning Representations*, 2023.

[363] Yu Yang, Hao Kang, and Baharan Mirzasoleiman. Towards sustainable learning: Coresets for data-efficient deep learning. In *International Conference on Machine Learning*, pages 39314–39330. PMLR, 2023.

[364] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, 2018.

[365] Jin-ge Yao, Xiaojun Wan, and Jianguo Xiao. Compressive document summarization via sparse optimization. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[366] Jiacheng Ye, Zhiyong Wu, Jiangtao Feng, Tao Yu, and Lingpeng Kong. Compositional exemplars for in-context learning. *arXiv preprint arXiv:2302.05698*, 2023.

[367] Lu Ye, Ze Tao, Yong Huang, and Yang Li. Chunkattention: Efficient self-attention with prefix-aware kv cache and two-phase partition, 2024.

[368] Tianzhu Ye, Li Dong, Yuqing Xia, Yutao Sun, Yi Zhu, Gao Huang, and Furu Wei. Differential transformer. *arXiv preprint arXiv:2410.05258*, 2024.

[369] Serena Yeung, Alireza Fathi, and Li Fei-Fei. Videoset: Video summary evaluation through text. *arXiv preprint arXiv:1406.5824*, 2014.

[370] Fan Yin, Jesse Vig, Philippe Laban, Shafiq Joty, Caiming Xiong, and Chien-Sheng Jason Wu. Did you read the instructions? rethinking the effectiveness of task definitions in instruction learning. *arXiv preprint arXiv:2306.01150*, 2023.

[371] Shukang Yin, Chaoyou Fu, Sirui Zhao, Ke Li, Xing Sun, Tong Xu, and Enhong Chen. A survey on multimodal large language models. *arXiv preprint arXiv:2306.13549*, 2023.

[372] Wangsong Yin, Mengwei Xu, Yuanchun Li, and Xuanzhe Liu. Llm as a system service on mobile devices. *arXiv preprint arXiv:2403.11805*, 2024.

[373] Chanwoong Yoon, Taewhoo Lee, Hyeon Hwang, Minbyul Jeong, and Jaewoo Kang. Compact: Compressing retrieved documents actively for question answering. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 21424–21439, 2024.

[374] Jaehong Yoon, Divyam Madaan, Eunho Yang, and Sung Ju Hwang. Online coreset selection for rehearsal-based continual learning. In *International Conference on Learning Representations*, 2022.

[375] Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. *arXiv preprint arXiv:1708.01547*, 2017.

[376] Yuxuan Yue, Zhihang Yuan, Haojie Duanmu, Sifan Zhou, Jianlong Wu, and Liqiang Nie. Wkvquant: Quantizing weight and key/value cache for large language models gains more, 2024.

[377] Zhenrui Yue, Honglei Zhuang, Aijun Bai, Kai Hui, Rolf Jagerman, Hansi Zeng, Zhen Qin, Dong Wang, Xuanhui Wang, and Michael Bendersky. Inference scaling for long-context retrieval augmented generation. In *The Thirteenth International Conference on Learning Representations*, 2025.

[378] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6023–6032, 2019.

[379] Sangdoo Yun, Seong Joon Oh, Byeongho Heo, Dongyoon Han, Junsuk Choe, and Sanghyuk Chun. Re-labeling imagenet: from single to multi-labels, from global to localized labels. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2340–2350, 2021.

[380] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, 2019.

[381] Yury Zemlyanskiy, Joshua Ainslie, Michiel de Jong, Philip Pham, Ilya Eckstein, and Fei Sha. Readtwice: Reading very large documents with memories. *arXiv preprint arXiv:2105.04241*, 2021.

[382] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *International Conference on Machine Learning*, pages 3987–3995. PMLR, 2017.

[383] Daochen Zha, Zaid Pervaiz Bhat, Kwei-Herng Lai, Fan Yang, and Xia Hu. Data-centric ai: Perspectives and challenges. In *Proceedings of the 2023 SIAM International Conference on Data Mining (SDM)*, pages 945–948. SIAM, 2023.

[384] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations*, 2022.

[385] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

[386] Jingyi Zhang, Jiaxing Huang, Sheng Jin, and Shijian Lu. Vision-language models for vision tasks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[387] Linjun Zhang, Zhun Deng, Kenji Kawaguchi, Amirata Ghorbani, and James Zou. How does mixup help with robustness and generalization? In *International Conference on Learning Representations*, 2021.

[388] Qizhe Zhang, Aosong Cheng, Ming Lu, Zhiyong Zhuo, Minqi Wang, Jiajun Cao, Shaobo Guo, Qi She, and Shanghang Zhang. [cls] attention is all you need for training-free visual token pruning: Make vlm inference faster. *arXiv preprint arXiv:2412.01818*, 2024.

[389] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

[390] Yaqian Zhang, Bernhard Pfahringer, Eibe Frank, Albert Bifet, Nick Jin Sean Lim, and Yunzhe Jia. A simple but strong baseline for online continual learning: Repeated augmented rehearsal. *Advances in Neural Information Processing Systems*, 35:14771–14783, 2022.

[391] Yiming Zhang, Shi Feng, and Chenhao Tan. Active example selection for in-context learning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 9134–9148, 2022.

[392] Yuan Zhang, Chun-Kai Fan, Junpeng Ma, Wenzhao Zheng, Tao Huang, Kuan Cheng, Denis Gudovskiy, Tomoyuki Okuno, Yohei Nakata, Kurt Keutzer, et al. Sparsevlm: Visual token sparsification for efficient vision-language model inference. *arXiv preprint arXiv:2410.04417*, 2024.

[393] Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang Wang, and Beidi Chen. H_2o: Heavy-hitter oracle for efficient generative inference of large language models, 2023.

[394] Amy Zhao, Guha Balakrishnan, Fredo Durand, John V Guttag, and Adrian V Dalca. Data augmentation using learned transformations for one-shot medical image segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8543–8553, 2019.

[395] Yang Zhao, Hao Zhang, and Xiuyuan Hu. Penalizing gradient norm for efficiently improving generalization in deep learning. In *International Conference on Machine Learning*, pages 26982–26992. PMLR, 2022.

[396] Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. Calibrate before use: Improving few-shot performance of language models. In *International Conference on Machine Learning*, pages 12697–12706. PMLR, 2021.

[397] Jingjing Zheng, Zhuolin Jiang, Rama Chellappa, and Jonathon P Phillips. Submodular attribute selection for action recognition in video. *Advances in Neural Information Processing Systems*, 27, 2014.

[398] Yu Zheng, Zhi Zhang, Shen Yan, and Mi Zhang. Deep autoaugment. *arXiv preprint arXiv:2203.06172*, 2022.

[399] Ming Zhong, Da Yin, Tao Yu, Ahmad Zaidi, Mutethia Mutuma, Rahul Jha, Ahmed Hassan, Asli Celikyilmaz, Yang Liu, Xipeng Qiu, et al. Qmsum: A new benchmark for query-based multi-domain meeting summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5905–5921, 2021.

[400] Zexuan Zhong, Tao Lei, and Danqi Chen. Training language models with memory augmentation. *arXiv preprint arXiv:2205.12674*, 2022.

[401] Zhisheng Zhong, Jiequan Cui, Shu Liu, and Jiaya Jia. Improving calibration for long-tailed recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16489–16498, 2021.

[402] Tianyi Zhou and Jeff Bilmes. Minimax curriculum learning: Machine teaching with desirable difficulties and scheduled diversity. In *International Conference on Learning Representations*, 2018.

[403] Tianyi Zhou, Shengjie Wang, and Jeffrey Bilmes. Curriculum learning by dynamic instance hardness. *Advances in Neural Information Processing Systems*, 33:8602–8613, 2020.

[404] Haichao Zhu, Li Dong, Furu Wei, Bing Qin, and Ting Liu. Transforming wikipedia into augmented data for query-focused summarization. *arXiv preprint arXiv:1911.03324*, 2019.