Towards Unsupervised Learning of Submodular Functions for Summarization

Chandrashekhar Lavania

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

Doctor of Philosophy

University of Washington

2020

Reading Committee:

Jeffrey Bilmes, Chair

Sreeram Kannan

Hannaneh Hajishirzi

Program Authorized to Offer Degree:
Electrical and Computer Engineering

University of Washington

Abstract

Towards Unsupervised Learning of Submodular Functions for Summarization

Chandrashekhar Lavania

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

In the information age, vast volumes of data are generated daily. There exist a plethora of data sources, including text, videos, and sensor networks. The large size of data can make it difficult to process. Furthermore, the generated data often has considerable redundancy. Therefore, extracting meaningful information from the data can make it easier to process for downstream tasks. Summarization is one way to extract this information.

In the past, submodular functions have been successfully used for summarizing data. These functions can be defined based on domain knowledge or can be learned from the data itself. However, supervised learning of submodular functions faces an obstacle as there is often a lack of known good summaries of the data for training. Therefore, it would be beneficial if the functions can be learned in an unsupervised manner.

This work proposes an approach towards learning a mixture of submodular functions in an unsupervised manner. It is achieved through a two-part process. First, an autoencoder neural network is trained in a constrained manner. The aim is to produce features such that a larger feature value implies that the input data sample has a larger amount of the corresponding learned property. It is analogous to bag-of-words features, with the "words" learned automatically.

Next, a mixture of submodular functions is instantiated using the learned features. Each component of the mixture consists of a concave composed with a modular function. The mixture weights are learned through an approach that does not directly utilize supervised summary information. It optimizes a set of meta-objectives, each of which corresponds to a likely necessary condition on what constitutes a good summarization objective. Empirical results on different data modalities show that the proposed two-part process produces functions that perform significantly better than a variety of baseline methods.

This work also explores other applications of the proposed features. The focus application is the summarization of video streams on the fly under a memory budget. The aim is to produce running summaries (within a budget) of incoming video streams at each time step. Any video snippet that is not part of a summary at a given time step is dropped to abide by the memory constraints. To this end, this work proposes two algorithms, one each for single-stream and multi-stream summarization. Empirical evaluations demonstrate that the algorithms, instantiated with the proposed features, can outperform the baselines.

In addition to these explorations, it is also shown that the proposed constrained training can be used with different flavors of autoencoder architectures and losses. The influence of these different setups is also demonstrated for the task of summarization.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

GLOSSARY

FEATURE: a representation, often numerical, of an object

AUTOENCODER FEATURES: output of the encoder in the autoencoder

MODEL: learned parameters of a framework

DNN: deep neural network

AUTOENCODER ARCHITECTURE: arrangement and structure of the layers in an autoencoder

RUNNING SUMMARY: summary of the entire history of a data stream produced on the fly and a each time step.

ACKNOWLEDGMENTS

I want to first thank my Ph.D. advisor, Jeffrey Bilmes, for accepting me into MELODI Lab and introducing me to submodularity. When we first met in person, he told me that the journey of doing a doctorate is an endeavor towards not just exploring a particular subject but also towards becoming a scholar. I find them to be very wise words, and they will be a guiding principle throughout my life as I continue to strive towards scholarship. I want to thank him for broadening my thought process, allowing me to imbibe novel ideas that were very useful during my doctoral studies. I also want to thank him for never allowing a lack of resources to be a hurdle for my research and funding me throughout my Ph.D. studies. I am extremely thankful for his guidance and support through research advice and life lessons that will likely influence my life.

I also want to thank my Ph.D. committee members: Sreeram Kannan, Hannaneh Hajishirzi and Archis Ghate for their advice and feedback.

My labmates also played a vital role during my doctoral studies. I want to thank my past and present labmates from the MELODI Lab: Richard Rogers, Galen Andrew, John Halloran, Bethany Herwaldt, Rishabh Iyer, Kai Wei, Yuzong Liu, Junyuan Xie, Sunil Thulasidasan, Tianyi Zhou, Shengjie Wang, Wenruo Bai, Jennifer Gillenwater, Rahul Kidambi, Neeraja Abhyankar, Lilly Kumari, Narendra Shivaraman, Arnav Das and Gantavya Bhatt. Thank you, everyone, for the great discussions and also for making my Ph.D. life much more rewarding and fun than it would have otherwise been.

I want to thank Brenda Larson, Jennifer Huberman and Alma Royeca for all their help

with the administrative procedures during my Ph.D.

I am thankful to grants by Intel (through ISTC-PC center) and the Semiconductor Research Corporation (through the TERRASWARM and CONIX centers).

I am also thankful to Shrisha Rao from IIIT-Bangalore for being a teacher, a mentor, introducing me to research, and inspiring me to pursue a doctorate. It was with his encouragement that I was able to decide and commit myself for the long haul of a Ph.D.

I want to thank my grandparents, Sarvati Devi, Yerramilli Pushpavalli, Ram Prasad Lavania, and Yerramilli Sreerama Murty, for their outlook on life and inculcating it in the family. They encouraged post-graduate education in the family and instilled an attitude towards becoming a scholar.

I also want to thank my parents Annapurna Saraswat, Seshu Lavania, Kripa Shanker Saraswat, and Umesh Chandra Lavania, for their supportive outlook towards my decision to pursue a doctorate. I am thankful for their advice and encouragement during my studies. They helped me in developing my ideals and thought process that supported me during my doctoral studies.

I am thankful to my sister Koshika Sharma, my brother-in-law Anurag Sharma, and their kids Tusharika and Divaskar for their encouragement and thoughtful support during my doctorate.

My quest for a doctorate was perhaps most taxing on Swasti Lavania (MDB). I am incredibly thankful for her support, understanding, and patience during this time. Her positive attitude during my studies was a pillar of support for me and motivated me to complete my doctoral studies.

DEDICATION

To My Grand Parents, Parents and MDB

Chapter 1
INTRODUCTION

Extensive sources of audio, video, speech, text documents and sensor data have become commonplace and are expected to become larger and more prevalent in the future [181, 32]. Such data sources are often redundant (e.g., surveillance video may consists of long stretches of almost static scenes with occasional motion-based interruption). Therefore, processing them is inefficient and requires more compute and memory resources than necessary. At times the size of a data set alone can make it infeasible to process entirely. The size especially becomes an issue for expensive computational tasks such as multi-epoch deep neural network training.

The extraction of meaningful information from the data can make it more manageable, and summarization is one way to extract information. Generating a summary necessitates taking a large set or stream of data and reducing it down to a small subset that represents the whole in some measurable way. It is starting to become an essential and useful facet of data science.

In this context, a summarization procedure involves at least two parts:

- **Defining a summarization objective:** Summarization often requires optimizing an objective function that evaluates the quality and representativeness of any subset of the data. The responsibility of the objective function is to capture the information of interest.

- **Optimizing the given objective:** Given such an objective, summarization focuses

on finding a subset (under constraints such as size) that produces a high objective function value.

Design and optimization of the objective can be a challenge, and a diverse range of methods are explored in literature for this purpose [56, 42, 136, 92, 155, 7]. Monotone, non-decreasing submodular functions have emerged as a promising class of functions for defining the summarization objective. These are set functions that naturally model notions of diversity, dispersion, and coverage. Such functions have shown their merit in several summarization scenarios in different domains [93, 115, 180, 154, 89, 177, 167, 176]. Let $V = \{v_1, v_2, \ldots, v_n\}$, denote a ground set. Then a set function $f : 2^V \rightarrow \mathbb{R}$ is *submodular* if $f(a|A) \geq f(a|B) \quad \forall A \subseteq B \subseteq V,\ a \in V \setminus B$, where $f(a|A) \triangleq f(a \cup A) - f(A)$ is the *gain* of adding element a to A. A set function is *monotone* if $f(A) \leq f(B)$ whenever $A \subseteq B$. A function f is *supermodular* if $-f$ is submodular, *modular* if it is both, and *normalized* if $f(\emptyset) = 0$.

A summarization procedure attempts to maximize the submodular objective to produce the summary where the maximization is usually performed under relevant constraints (such as the summary budget). The problem of constrained maximization of submodular functions is NP-Hard [124]. There has, however, been extensive research on efficient procedures to produce approximate solutions [124, 110, 172, 14, 113, 116, 5, 12, 112].

On the other hand, the problem of defining the optimum submodular objective has seen little exploration. These functions are often determined based on domain knowledge [89, 90, 180, 16], and an extensive understanding of the data is required in this approach. Such knowledge is not common and can prove a significant impediment in using submodular functions for summarization.

A different approach to defining the submodular function is to learn it based on the data. In a supervised setting, it was explored with promising results [185, 19, 52, 94]. The most

significant benefit of learning the submodular objective function is that it requires minimal knowledge of the structures in the data. The learning procedure builds upon the initially limited knowledge and produces a powerful objective function. However, one major limitation of these learning procedures is that they require labeled data where the labels are ground truth summaries. The labeling procedure requires manual annotation and summarization of potentially substantial data sets. One example is massive scale image summarization, where V is a large (e.g., $n > 1M$) collection of images, and the aim is to produce summarizes of size 1000. Another example arises from environmental sensors (e.g., video, audio, temperature, humidity, pressure, luminous intensity, pollution, EM radiation), and smart networks. An escalating variety and volume of such sensors are used in modern smart cities. The result of such usage is a collection of data that is likely highly redundant. This redundant cache of data can benefit from summarization, but the collection's size poses labeling issues that often make the task of acquiring labeled data for summarization very challenging. An unsupervised approach is beneficial to mitigate this issue. The focus of this thesis is to provide advancements towards unsupervised learning of submodular functions.

The set of monotone submodular functions is still vast. Therefore, the task of learning the submodular function is non trivial. One technique is to restrict the choice still further. A sub-class of monotone submodular functions called "feature based" functions [70] is a promising option. Feature based submodular functions are defined as:

$$F(A) = \sum_{u \in U} w_u \phi_u(m_u(A)), \tag{1.1}$$

where U is a set of features, $w \in \mathbb{R}_+^U$ (for $u \in U$, $w_u \geq 0$ is a feature weight), ϕ_u is a monotone non-decreasing concave function, and $m_u : V \to \mathbb{R}_+$ is a non-negative normalized modular function specific to feature u. For data item $v \in V$ and feature $u \in U$, $m_u(v)$

measures the degree of "u-ness" of v, and for $A \subseteq V$, $m_u(A) = \sum_{a \in A} m_u(a)$ is an additive measure of u-ness of the data items within A. The usefulness of feature based functions has been successfully displayed in various summarization tasks [127, 70, 176, 167]. These functions assume that data items are represented as a non-negative vector in feature space U. This representation is useful since the evaluation $F(A)$ of $A \subseteq V$ does not require access to all of V. Thus, these functions have a lower computational cost compared to functions whose evaluation requires access to the whole data (such as the $O(n^2)$ cost required to use a simiarity matrix [95]).

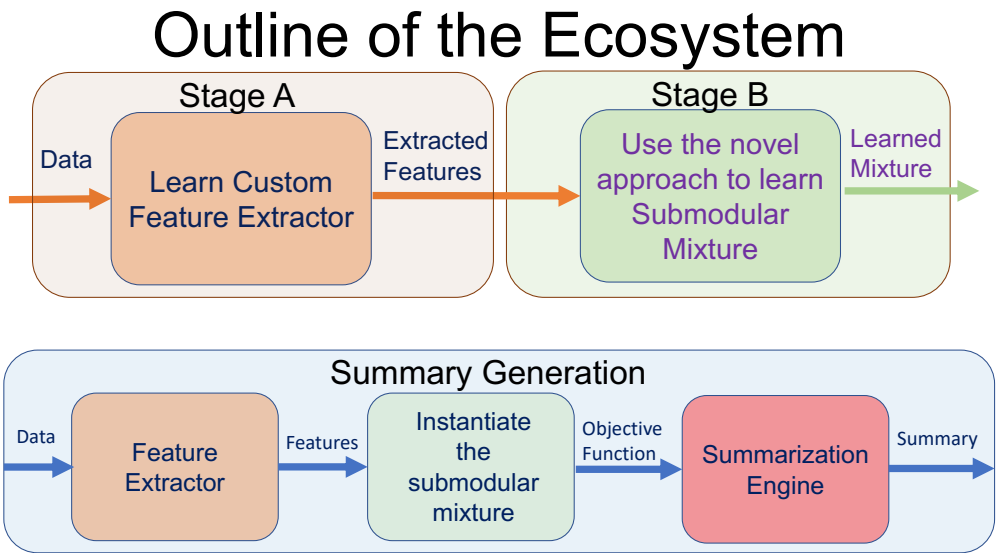

Figure 1.1: This figure presents the overall outline of the ecosystem. Stage A learns a novel constrained neural network for extracting data features that are particularly conducive for use in submodular functions. Stage B learns the weights of a mixture of submodular functions. The component functions in the mixture are instantiated using the features learned in Stage A. The mixture is learned using an innovative procedure described in Chapter 6. The summarization process uses the submodular mixture as the objective. A constrained optimization generates the summary.

This thesis takes a step towards an unsupervised approach to learn the submodular function for extractive summarization. The objective is defined using a weighted mixture of

multiple submodular functions (Chapter 6). The overall outline of the ecosystem is depicted in Figure 1.1. The process of learning the mixture consists of the following:

- **Learn the feature representation for the data:** A prominent characteristic of feature based functions (Eq 1.1) is the features used to define it. The procedure assumes the following as input: (1) number $|U|$ of features, and (2) settings for the hyperparameters. The procedure learns the features U, the modular functions $\{m_u\}_{u \in U}$ where $m_u : V \rightarrow \mathbb{R}_+$ in an unsupervised manner. The detailed approach and analysis are described in Chapter 4.

- **Learn the mixture weights:** It is hard to learn all $O(2^{|V|})$ free parameters [6, 38]. Furthermore, learning just the mixture weights would also require a large amount of training data. This thesis' approach is to reduce the problem down to finding a good setting for a handful of hyperparameters. The hyperparameters determine a meta-optimization used to determine the mixture weights. The relatively small quantity of hyperparameters, allows them to be chosen feasibly with only a minimal amount of evaluation information. Hence, even though the approach is not fully unsupervised, this thesis proposes the first substantial advancement towards the unsupervised learning of submodular mixtures since perhaps [178] who instantiated a function using Gaussian mixture models and HMMs which were themselves produced using unsupervised learning.

The organization of the thesis is in the following manner. A basic introduction to submodular functions and their optimization procedures is given in Chapter 2. The different categories (domain-independent) of automatic summarization are described in Chapter 3. Chapter 4 presents this thesis' novel unsupervised feature learning procedure for instantiating submodular functions. In Chapter 5, applications for the proposed features are discussed.

Chapter 6 describes the innovative mixture learning procedure proposed in this thesis. In Chapter 7, applications for the mixture learning procedure are explored. Chapter 8 discusses future areas of research and the thesis concludes in Chapter 9.

Some of the work described in this thesis has been published in conference proceedings [81]. The work related to constrained video summarization to produce running summaries is based on a preprint. Other works, such as the different flavors of feature generation procedures and asymmetric autoencoders, are being prepared for submission.

Chapter 2

SUBMODULAR FUNCTIONS

Submodular functions are set functions defined over all subsets of a ground set V ($f : 2^V \rightarrow \mathbb{R}$). Any set function defined in this manner is submodular if:

$$f(A) + f(B) \geq f(A \cup B) + f(A \cap B) \;\; A,\; B \subseteq V \tag{2.1}$$

A more intuitive and equivalent condition can be stated as:

$$f(a|A) \geq f(a|B) \;\; \forall A \subseteq B \subseteq V,\; a \in V \setminus B \tag{2.2}$$

where $f(a|A) \triangleq f(a \cup A) - f(A)$ is the gain of adding element a to A. This condition demonstrates the 'diminishing returns' property of submodular functions. This property is often used in summarization procedures for extracting diverse summary [81, 127, 70, 176, 167]. A submodular function is said to be normalized if $f(\emptyset) = 0$. It is monotone and non-decreasing if $f(A) \leq f(B)$ when $A \subseteq B \subseteq V$. A normalized, monotone and non-decreasing submodular function is often referred to as polymatroid function [96, 26, 27].

Figure 2.1 demonstrates an example of a submodular function. For a given set $S \subseteq V$, in this figure, $f(S)$ is defined as the number of categories covered by the items in set S. The available categories are food, ball, tree, and car. In Figure 2.1, set A has only trees. Therefore, $f(A) = 1$. Similarly, set B has a ball, a tree, a car, and a sandwich. Therefore,

B has all four categories and $f(B) = 4$. Set C can be created by adding another ball to set B. The evaluation of set C, however, does not change from the evaluation of B ($f(C) = 4$). Despite the addition of a new ball, the number of categories covered by C is still four (the same as B). This demonstrates the 'diminishing returns' property of the set function $f(\cdot)$ and also shows that it is submodular (in accordance with property 2.2).

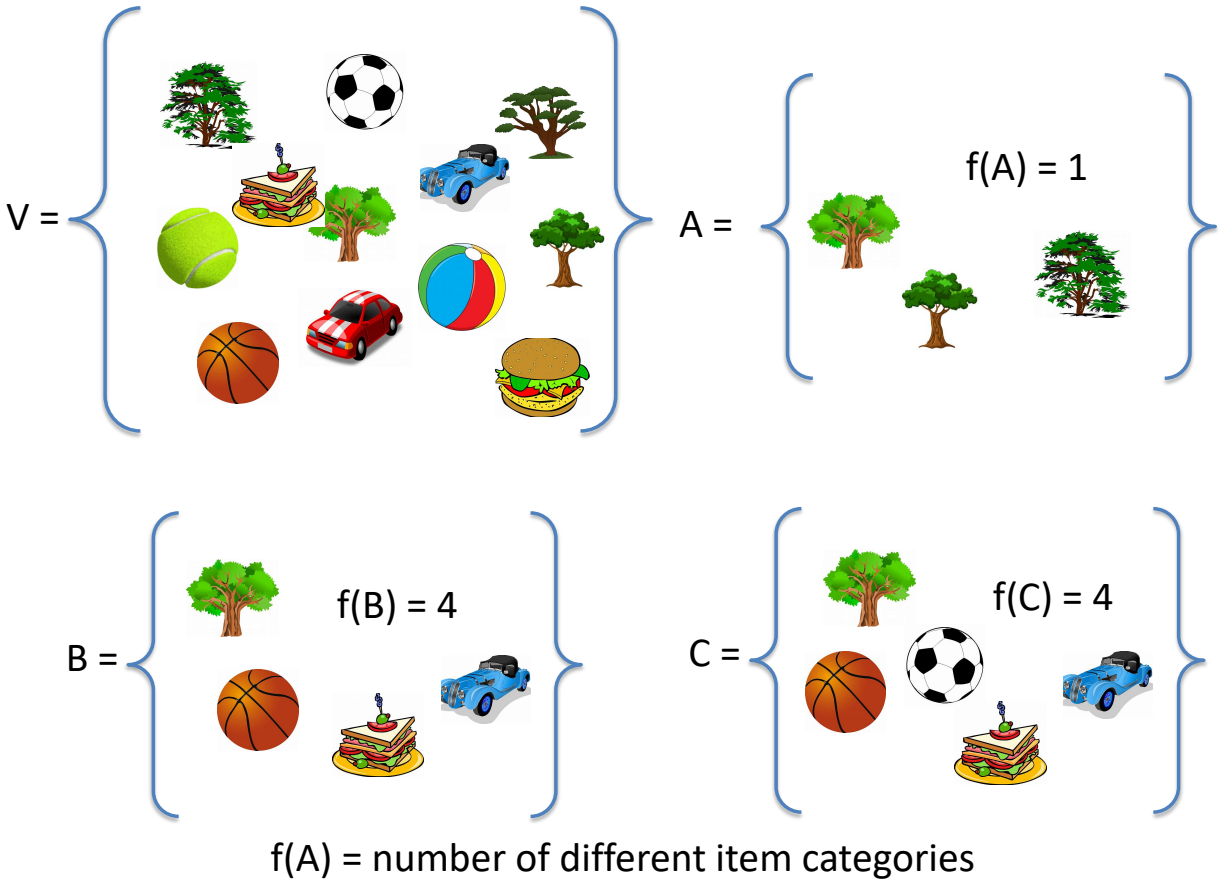

Figure 2.1: Example of a submodular function

It is interesting to note that several common set functions that have been used in the scientific literature are submodular. Some typical examples of submodular functions include:

- Entropy: Given a collection of random variables V, the entropy of $A \subseteq V$ ($H(A)$) is submodular [122].

- Symmetric Mutual Information: Similarly the mutual information between A and $V \setminus A$ ($I(A, V \setminus A)$) is submodular [122].

- Set Cover: Let S_1, S_2, ..., S_n be subsets of $\mathfrak{X}$. Given $V = \{1, 2, \ldots, n\}$, $f(A) = |\bigcup_{a \in A} S_a|$ where $A \subseteq V$ is submodular [171].

- Facility Location function (FL). For sets V and A, $f(A) = \sum_{v \in V} \max_{a \in A} r(v, a)$. Here r is a real number that connects v and a in some form (based on the task at hand). For example, if V is a set of locations then for $A \subseteq V$, the facility location function can model the value provided to customers by choosing the locations from A to open facilities (hence the name) [74]. Facility Location functions can also be used as a summarization objective. A typical example of $r(\cdot, \cdot)$ in this context a similarity value.

- Graph Cut functions. Given a graph $G(V, E)$ defined on vertex set V and edges E, the cut function $f(A) = \sum_{i \in A, j \in V \setminus A} e_{ij}$ is submodular. Here $e_{ij} \in E$ is the edge between $i \in A$ and $j \in V \setminus A$.

This list is just a small set of examples from the numerous submodular functions that have been used in literature.

In recent years, the usage of submodular functions in the field of machine learning has gained traction. They have been used for data summarization [89, 176, 52, 112], as energy functions in probablistic models [44, 60, 48], active learning [50, 175, 179], recommendation systems [134] etc. The submodular function's application is often through: A) Maximization of the submodular function or B) Minimization of it.

2.1 Maximization of Submodular Functions

Submodular function maximization is useful when there is a need for maximizing the diversity in a set (eg: during summarization). Given a ground set V, this optimization problem is set up as:

$$S^* \in \underset{S \subseteq V}{\operatorname{argmax}} f(S). \tag{2.3}$$

If the focus is on constrained submodular maximization (such as budgeted summarization) then the optimization takes the form:

$$S^* \in \underset{S \subseteq V, S \in \mathscr{C}}{\operatorname{argmax}} f(S). \tag{2.4}$$

Here, $\mathscr{C}$ is the set of constraints. The problem itself is NP-Hard for many constraints such as cardinality constraint, matroid constraint, and many more. Fortunately, for polymatroid submodular functions, there exist greedy algorithms [124] that can produce a solution with a $1 - \frac{1}{e}$ guarantee in polynomial time (for cardinality constraint). In practice, there are many flavors of greedy that can be used for submodular maximization [110, 114, 116].

This thesis's focus is on data summarization and therefore constrained submodular maximization is heavily involved in the overall ecosystem.

2.2 Minimization of Submodular Functions

In literature, there is no dearth of applications that can benefit from submodular function minimization. It has been successfully used in image segmentation [62], clustering [123, 119], graph cut [161, 118], supply chain management [152, 97, 105], and many more. If the focus

is on constrained submodular minimization then the optimization takes the form:

$$S^* \in \underset{S \subseteq V, S \in \mathscr{C}}{\operatorname{argmin}} f(S). \tag{2.5}$$

Chapter 3

AUTOMATIC SUMMARIZATION

The field of automatic summarization involves building models that can read in vast amounts of data and automatically output meaningful information from it (such as shown in Figure 3.1). It is often desirable that such information is salient, concise, and can describe the input data in a reasonably comprehensive manner. One way to categorize the different types of summarization is 1) Extractive Summarization and 2) Abstractive Summarization. The focus of this thesis is on extractive summarization.

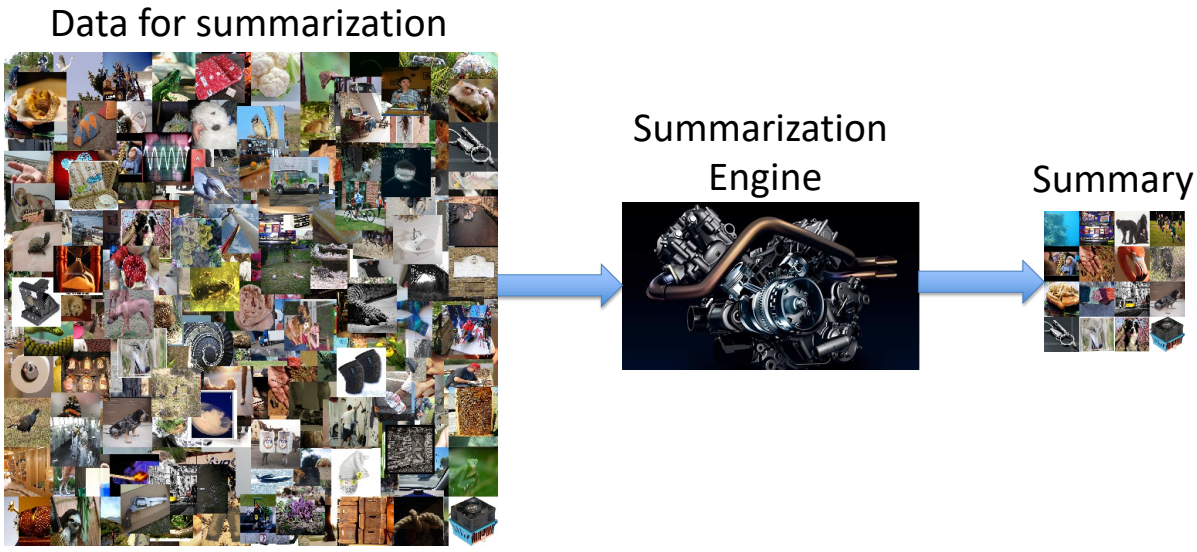

Figure 3.1: The process of automatic summarization.

3.1 Extractive Summarization

Extractive summarization comprises any method that produces a summary containing exact copies of elements from the data points' ground set. Therefore, given a ground set V, these methods generate a summary S such that $S \subseteq V$. Figure 3.2 is an example of extractive summarization. Here, the tennis ball, car, and tree are the same objects that are present in the ground set.

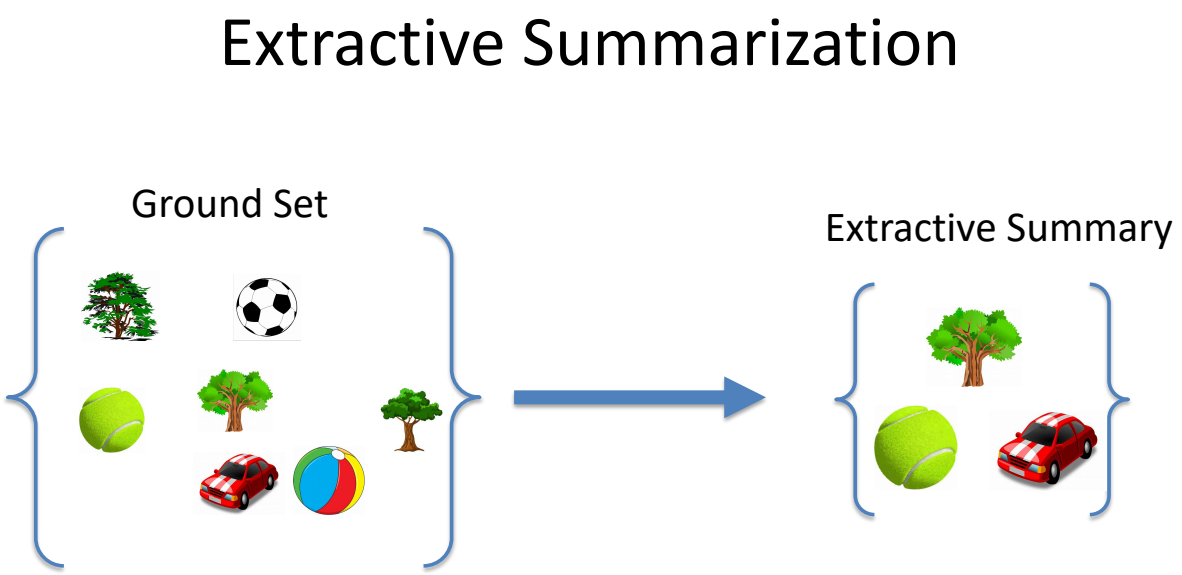

Figure 3.2: An example of Extractive Summarization.

Such methods are under active exploration in several fields, such as computer vision and document analysis. In image summarization and static video summarization, these methods belong to the class of key-frame selection methods. On the other hand, dynamic video summarization selects snippets (chunks of locally contiguous images) instead of just key-frames. Recently, de Avila et al. [29] used k-means clustering of the color histogram [162] features from video frames to generate static summaries. Zhang et al. [186] used LSTMs

to design a supervised approach for the selection of key-frames or key sub shots. Ouyang et al. [131], use the neighborhood of a selected key-frame as a snippet. Yao et al. [182] explore first-person videos' summarization through highlight detection. They split their video segment into spatial and temporal streams. Deep convolutional neural networks for highlight prediction consume each of these streams. The outputs of these networks are fused to generate the highlight score of a video segment.

Extractive techniques are also widely used for document summarization. In this case, the methods select sentences from given documents to create a representative summary. Such techniques have a rich history for document summarization. Luhn [98] introduced a method for sentence extraction based on word frequencies. More recently, Maximal Marginal Relevance (MMR) was used as a method for sentence extraction [28, 106]. Graph-based methods to gauge the importance of different textual units have also been explored [37, 109, 108, 150].

As of late, neural networks were successfully used for extractive summarization. Kaageback et al. [64] employed a recursive autoencoder for sentence extraction. Yin and Pie [183] used convolution neural networks (CNNs) to generate continuous space representations of sentences. They use these representations for scoring the sentences for document summarization. Cheng and Lapata [21] proposed an attention-based encoder-decoder setup for the creation of extractive summaries. Nallapati et al. [121] introduced an RNN based model that can be learned in an end-to-end manner and produce extractive summaries.

Submodular functions have also been successfully used for extractive summarization in both vision and text domains. Tschiatschek et al. [167] learned a mixture of submodular functions for image summarization. Similarly, Gygli et al. [52] used submodular mixtures for video summarization. Xu et al. [180] formulate a summarization of egocentric videos as constrained maximization of the objective function. Chakraborty et al. [16] consider

the problem of adaptive key-frame selection for video summarization. They formulate the problem as unconstrained maximization and define the objective on a complete similarity graph. Submodular functions have also seen usage for extractive document summarization. Lin and Bilmes explore a class of submodular functions and learn submodular mixtures for document summarization [91, 90, 89]. Kulesza and Taskar [78] successfully show that determinantal point processes (DPPs) can be utilized for document summarization.

3.2 Abstractive Summarization

Unlike extractive summarization, abstractive summarization does not need the summary to contain data points that are an exact match to unsummarized data. Figure 3.3 is an example of abstractive summarization. Here, the summary covers concepts present in the ground set. The summary elements, however, are not an exact match to the ground set elements.

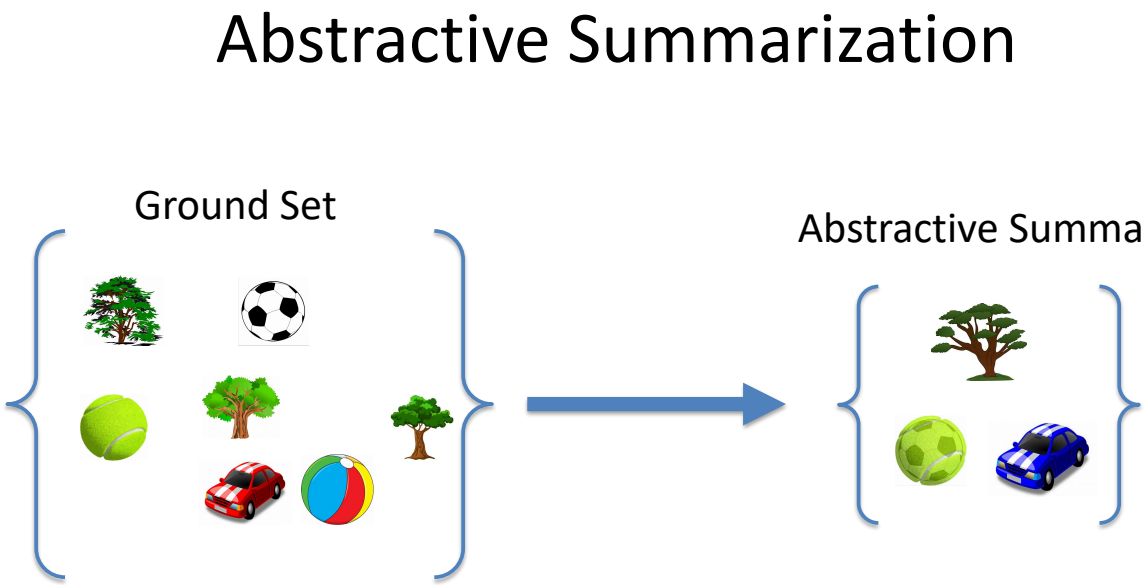

Figure 3.3: An example of Abstractive Summarization.

Therefore, abstractive summarization is a form of summary generation instead of a sum-

mary extraction. The most common example is a document's summary created by a reader in their own words instead of taking sentences directly from the document.

Mowad and Mostafa [117] used a semantic graph creation approach. They generate abstractive summaries with concepts such as ontologies, heuristic rules, and verbs' tenses to create grammatically correct and lucid sentences. More recently, neural networks have been employed for abstractive summarization. Chopra et al. [22] use attention-based recurrent neural networks for abstractive sentence summarization. Tan et al., [163] used an encoder-decoder framework for document-level abstractive summarization. Their approach involved the usage of a graph-based attention mechanism in their neural network. These are just a few examples as abstractive summarization has received extensive attention in the literature, as shown in the survey by Rachabathuni [141].

Chapter 4

UNSUPERVISED FEATURE GENERATION FOR SUBMODULAR SUMMARIZATION

Feature engineering and generation [140, 79, 190, 2] is an integral part of most real-world machine learning frameworks. It involves the automatic or by-hand generation of vector representations of raw data objects. This raw data can itself be highly varied in size, structure, and dimension. It can be in a variety of modalities such as images, video, text, speech, surveys, sensor streams, and many more. The feature extractor's responsibility is to produce features that can be usefully consumed downstream by later stages of the application. The quality of the features depends on the fidelity of the captured information. However, this does not imply that there must necessarily be a large feature set to represent all distinct or peculiar properties of a data object. It is well known that machine learning methods can suffer greatly if the feature dimensionality is too high (i.e., the curse of dimensionality). To counter this, many low-dimensional feature extraction and transformation procedures have been developed and used in practice [137, 57, 72, 142, 3, 23, 164, 145, 75, 126].

The advancement in GPUs (Graphics Processing Units), and the availability of large data sets (for training) has encouraged the usage of deep neural networks for dimensionality reduction [55] and representation learning [8]. Such methods force a network to squeeze the data through a bottleneck before being asked to reconstruct it faithfully. The bottleneck is most commonly much smaller in size than the input data. Therefore, it corresponds to a form of compression. Downstream processing can then more efficiently utilize the compressed form (say for distance calculations) than the original. The size of the bottleneck influences

the quality of reconstruction. Small bottleneck size may correspond to lossy compression (with a distorted reconstruction). Such a process resembles a form of "*gisting*" [129].

There are many ways to construct such a representation, including autoencoders [10, 169, 69] and GANs [47]. A deep neural network based autoencoder consists of two parts. First, there is the encoder. The encoder transforms an input data sample x into a representation in different feature space (often lower-dimensional). The encoder is, therefore, a mapping $\mathfrak{e} : \mathbb{D} \rightarrow \mathbb{R}^d$ to a 'd' dimensional feature space. The second part of the architecture is the decoder. It transforms the 'd' dimensional representation back into the original space of the data sample. The decoder corresponds to $\mathfrak{d} : \mathbb{R}^d \rightarrow \mathbb{D}$. Here $\mathbb{D}$ is the domain of x, the data sample. There is no dearth of ways to create and train such architectures [10, 69, 18, 102, 20].

The outputs of the low dimensional encoder instantiate the submodular objective in the summarization process. In the case of feature-based functions (equation 1.1), the modular functions are built through these representations. Let $U = \{1, 2, \ldots, d\}$ (the dimensions for 'd' dimensional representations). Let $v \in V$ be a data item. The modular score corresponding to $u \in U$ for item v is defined as $m_u(v) = \mathfrak{e}(v)(u)$. It is the u^{th} element in the vector encoding of item v. Furthermore, submodularity requires that $m_u(v) \geq 0 \ \forall u \in U$ and $v \in V$. This can be obtained in the autoencoder by ensuring that the non-linearity at the bottleneck always produces a non-negative output (such as logistic and ReLU). Overall, if $A \subseteq V$ then $m_u(A) = \sum_{a \in A} \mathfrak{e}(a)(u)$.

It is essential to understand that not all non-negative features have the required properties to be useful in feature based functions. It is true that mathematically they can be used to instantiate these functions. However, it does not imply that all non-negative features leverage the inherent characteristics of feature based submodular functions. A core aspect of these functions (equation 1.1) is that a higher value of $m_u(v)$ should imply a higher presence of property u in the data sample v. Thus, the intention is to learn a representation such

that the property u is more pronounced in v_1 than v_2 if $\mathfrak{e}(v_1)(u) < \mathfrak{e}(v_2)(u)$. Autoencoders automatically learn the meaning of u through training. It is, however, not necessary that the property u becomes more notable in the sample v as $\mathfrak{e}(v)(u)$ rises.

Such a characteristic is the backbone of "bag of words" representations. In them, each individual 'word' count (or fractional value) denotes the importance of that word in the data sample. For instance, in n-gram text representations (e.g., TFIDF [148]), the symbol count is proportional to its frequency in a document. The aim is for the autoencoder representations to have a similar property.

The proposed feature learning procedure endeavors to provide a scheme to produce feature representations that denote the degree of prevalence of each of the $|U|$ learned words within the input data sample v. Furthermore, all data elements present in a set $A \subseteq V$ should contribute to u-ness of A in an additive manner i.e., $m_u(A) = \sum_{a \in A} m_u(a)$. This thesis calls this property "*additively contributive*" (AC). Hereafter, these features are called "AC features". Some of the work described here was published as [81].

4.1 Detailed Approach

The intention is to produce features that are likely to have an AC interpretation. Furthermore, an unsupervised approach needs to produce these features. To this end, this thesis introduces a strategy to generate them through constrained autoencoders.

One way to describe the decoding part of the autoencoder is $\mathfrak{d}(\mathfrak{e}(v))$. Here $v \in V$ is the data element. In this setup, the first operation on the encoded representation is often a matrix multiplication. For an encoded representation in $\mathbb{R}_+^d$ (non-negative real values in d dimensions) let $W \in \mathbb{R}^{d' \times d}$ be a $d' \times d$ matrix used for this first operation. Therefore, the expression of the overall decoder is $\mathfrak{d}(\mathfrak{e}(v)) = \mathfrak{d}'(W\mathfrak{e}(v))$. Here, $\mathfrak{d}'(\cdot)$ comprises of all the operations in $\mathfrak{d}(\cdot)$ apart from the first matrix multiplication. Post-training W may contain

both negative and non-negative values. The negative values, however, can cause the encoded representation to not have an AC interpretation. In such a scenario the value of $\mathfrak{e}(v)(u)$ (the value of dimension u in the representation of v) gets multiplied by a negative value from the matrix W. Therefore, an increase in $\mathfrak{e}(v)(u)$ can decrease the contribution of u in the final decoded output. A method to mitigate this issue is to restrict W to only non-negative values. The overall optimization takes the following form:

$$\min_{\mathfrak{d}', \mathfrak{e}, W \in \mathbb{R}_{+}^{d' \times d}} E_{v \sim p}[\|v - \mathfrak{d}'(W\mathfrak{e}(v))\|] \tag{4.1}$$

The addition of constraints increases the complexity of the optimization procedure. This complexity might make it challenging to train the network. It might also result in non-desirable networks. For instance, it can be the case that the trained network converges to a state where the matrix W is close to an identity matrix. In this situation, the majority of decoding is done by $\mathfrak{d}'(\cdot)$ and is equivalent to a non AC constrained setup (with a somewhat asymmetric architecture). If, however, the entire decoder used matrices with only non-negative elements, then such a possibility is eliminated. Expulsive regularization [144, 103] can also be used during training to discourage an identity matrix like structure of W. Furthermore, a highly asymmetric architecture with a much broader and deeper encoder than the decoder might prevent the trained model from bypassing decoder layers. During the experiments, and from a more practical perspective, the strategy of projecting only W back to the positive orthant was sufficient. W was never close to any form of identity transform. Besides, summarization results (Chapters 5 and 7) always demonstrated a notable benefit to AC constrained training over unconstrained training. The trained autoencoder model instantiates the feature-based functions as in Eq. 1.1 with $m_u(A) = \sum_{a \in A} \mathfrak{e}(a)(u)$.

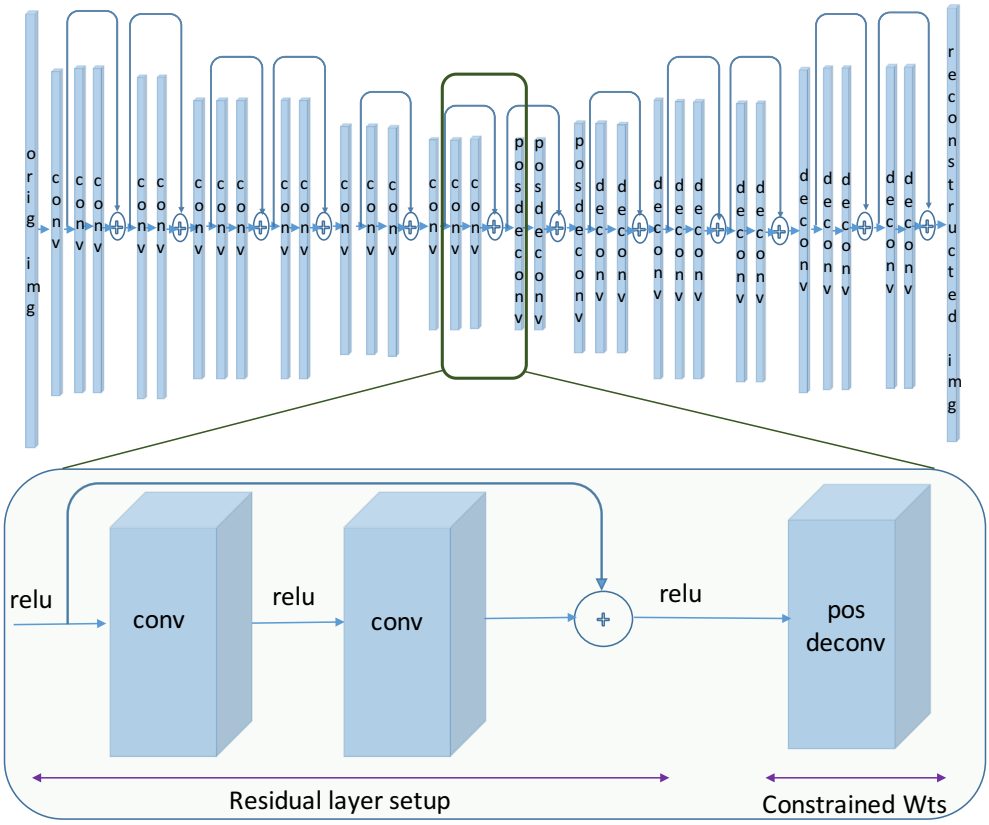

Figure 4.1: Sample autoencoder architecture for producing AC image features. The zoomed view shows that the first layer after the bottleneck is pos deconv. The training procedure constraints the weight matrices in pos deconv layer to be non-negative.

4.2 Related Work

In past literature, there have been instances of features having the characteristic that a higher value implies a higher presence of the corresponding data object. "Bag-of-words" is a well-known example of such features. They have been successfully used in many text processing [149], natural language processing [80] and computer vision [138] tasks. The literature in this area is vast, and these few examples do not do full justice to past explorations.

In computer vision literature, there have been instances where features produced using deep neural networks were converted to a bag of visual words. For example, Tschiatschek et

al. [167] created such visual words. They use layer 17 of OverFeat [151] to generate first level features. k-means and a kernel codebook [17] are used to cluster and encode these features to produce a bag of visual words. This thesis, however, offers a strategy to generate the AC features directly from the neural network. No extra processing step is required.

There have been explorations in past literature that use constrained matrices. It is the backbone of non-negative matrix factorization (NMF [84]). In NMF, a matrix V is factorized into matrices W and H subject to $V \approx WH$. The constraint is that V, W and H can have only non-negative elements.

Trigeorgis et al. [165] explored the use of constrained internal representations in a neural network. The result was a Deep semi-NMF (Semi Non Negative Matrix Factorization) model. The model was trained to factor a data matrix $X^{\pm}$ into $m+1$ factors as: $X^{\pm} \approx Z_1^{\pm} Z_2^{\pm} \dots Z_m^{\pm} H_m^{+}$. Here H_m^{+} indicates that the elements of the matrix are all non-negative. The constituent matrices were further constrained to ensure $H_{m-1}^{+} \approx Z_m^{\pm} H_m^{+}$ (for all factors). This paved the path for decompositions of the form $X^{\pm} \approx Z_1^{\pm} H_1^{+} \approx Z_1^{\pm} Z_2^{\pm} H_2^{+}$ and so on. Trigeorgis et al. [165] also analyze the scenario where there is a non-linear function $g(\cdot)$ between each of the $H_1^{+}, \dots, H_{m-1}^{+}$ representations such that $g(H_i^{+}) \approx Z_{i+1}^{\pm} H_{i+1}^{+}$. They demonstrated that a neural network with such constraints could learn representations of the data beneficial for clustering based on the attributes of the neural network layers. They show on the CMU MultiPIE database [49] that the different levels of a 3 level deep semi-NMF model can learn features that are useful for pose clustering, expression clustering, and identity clustering. They, unlike the strategy proposed in this thesis, do not directly constrain the neural network weights.

Naing and Elhamifar [120] learn features for summarization using partial summaries. They assume that a subset of the summary is available during training and that each data point is a linear combination of the summary elements (seen plus unseen). They construct

the loss as a combination of an encoding loss, a diversity loss, and an autoencoder reconstruction loss. The encoding loss evaluates how well the representations of the summary elements construct the representations of all the data. The diversity loss encourages the summary to be well spread and diverse. The autoencoder loss is used to mitigate overfitting. Furthermore, the encoding loss requires the usage of the full summary. Therefore, the unseen summary elements are predicted using a greedy procedure that uses thresholds. Even though this procedure uses an autoencoder, it does not have any restrictions on the feature representations that encourage the AC characteristic. In addition to this, it directly uses the partially available summary information during training, making it a semi-supervised procedure.

4.3 AC constraints with different flavors of autoencoders

This section explores the use of AC constraints with:

- Variational Autoencoder (VAE) [69] like architecture
- The use of adversarial loss [47, 104] in training of the feature generator.
- Highly asymmetric autoencoders with a deep encoder and a single layer decoder.

VAE like autoencoder architecture: Standard VAE learns parameters for Gaussian distributions at the bottleneck. During inference, each node's output at the bottleneck is sampled from the corresponding distribution (using the learned parameters). This process can generate features that have a negative value. Feature based submodular functions (equation 1.1), however, need their features to be non-negative. One way to generate such features from a VAE is to use an activation function (e.g., ReLU, sigmoid), which might lead to loss of information as support of the learned distributions is not directly set to be non-negative. It

might, therefore, be beneficial to use a distribution with non-negative support. To this end, this work learns the parameters of Weibull distribution at the bottleneck. Hence, during inference, the output at the bottleneck is always non-negative. The use of Weibull distribution at the bottleneck has the added advantage of having a reparameterization trick. Samples from a two-parameter Weibull distribution can be generated using a Uniform distribution defined over support: $[0, 1]$. The AC constraint follows the bottleneck just as it is in the standard autoencoder.

Incorporating adversarial loss in the training: Generative Adversarial Networks (GANs) [47] have gained much traction for generating close to real images. The backbone of GANs is a network that learns to decide whether the data produced by the generator is real or fake. In this thesis, the generator is the autoencoder with AC constraints. The differentiator network is then used to determine the quality of the autoencoder output. A reconstruction loss and an adversarial loss based on Least-Square GAN (LSGAN [104]) trains the entire setup.

Single layer decoder architecture: In this setup, an encoder (possibly deep) is followed by a single layer decoder. Such an architecture has several uses, including the production of features for summarization. An AC constraint on the single layer of the decoder can be used to learn a dictionary corresponding to the learned representation at the bottleneck. The dictionary's presence allows the direct influence of the feature value at the bottleneck to be visible at the output. These single layer architectures are also useful for data storage and transmission. In such setups, the encoder is responsible for majority of the workload. A single layer decoder allows the features to be transmitted and reconstructed into the actual data element in resource-constrained environments.

Figure 4.2 demonstrates the performance of each of these setups on the image summarization data set [167]. AC constraint was utilized for training each of these models on the

ILSVRC 2012 [147] data set. The trained models were used to extract features for images from the image summarization data set. This data set is made up of 14 smaller sets, with each set contains 100 images. These sets have a list of associated user summaries. Each summary contains 10 images. The resultant image features were used to instantiate a mixture of submodular functions (equation 7.2) with each component as a feature based function. A max-margin learning framework [167] was applied to learn the mixtures in a supervised manner.

Figure 4.3 shows comparison between the reconstruction ability of the different models. Figure 4.4 displays the top five best and worst (based on structural similarity [173]) reconstructed images for each model. The bottleneck size is 256. Therefore, the bottleneck size is $\frac{1}{738}$ of the raw input ($256 \times 256 \times 3$).

4.3.1 Architectures for the different autoencoder flavors

Tables 10.2, 10.3, 10.4 and 10.1 show the architectures for the regular autoencoder, VAE like architecture that uses a Weibull distribution at the bottleneck ('va-w'), the differentiator for the setup that incorporates adversarial loss (generator is same as Table 10.2) and the one layer decoder.

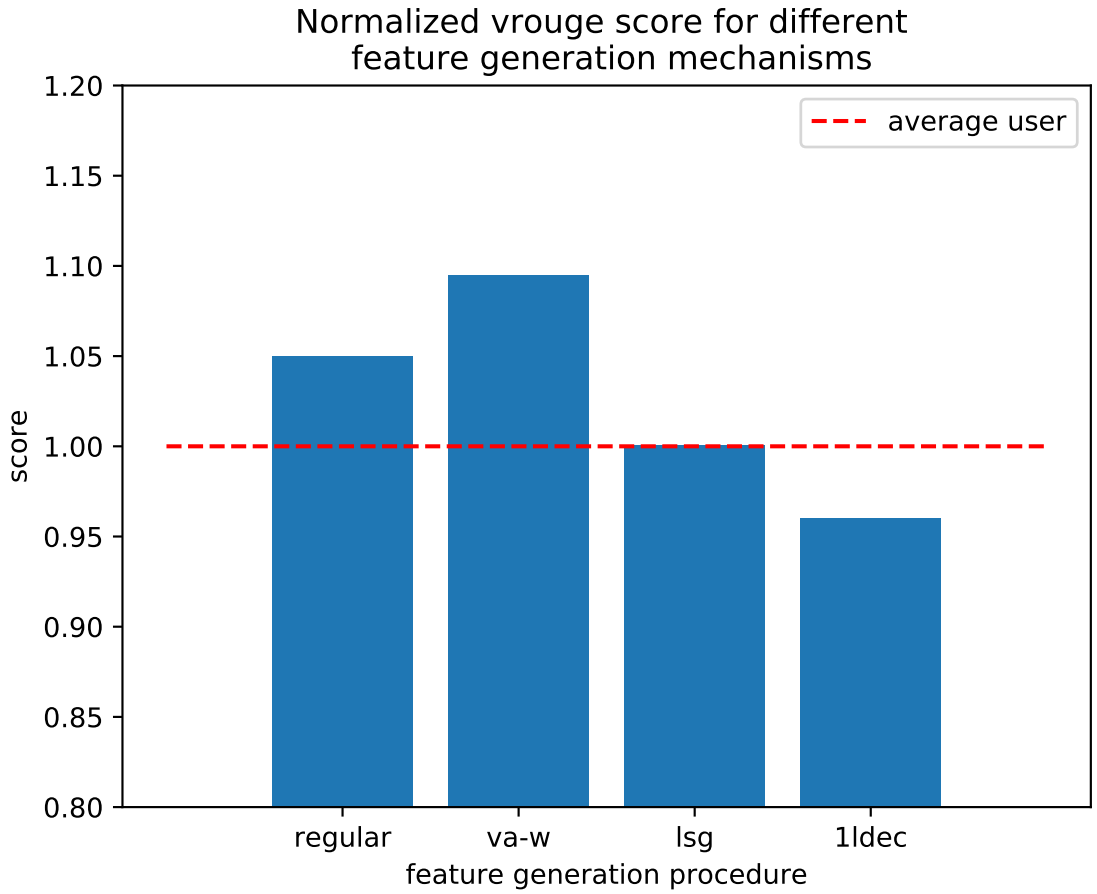

Figure 4.2: Summarization performance (normalized vrouge score) of different flavors of autoencoders on the image summarization data set. The comparison is between 1) regular autoencoders ('reg'), 2) VAE like architecture that uses a Weibull distribution at the bottleneck ('va-w'), 3) an autoencoder that incorporates adversarial loss in training ('lsg'), and 4) an architecture that has the same encoder as the regular setup but uses a single layer decoder ('1ldec'). Each of these models was trained with AC constraint.

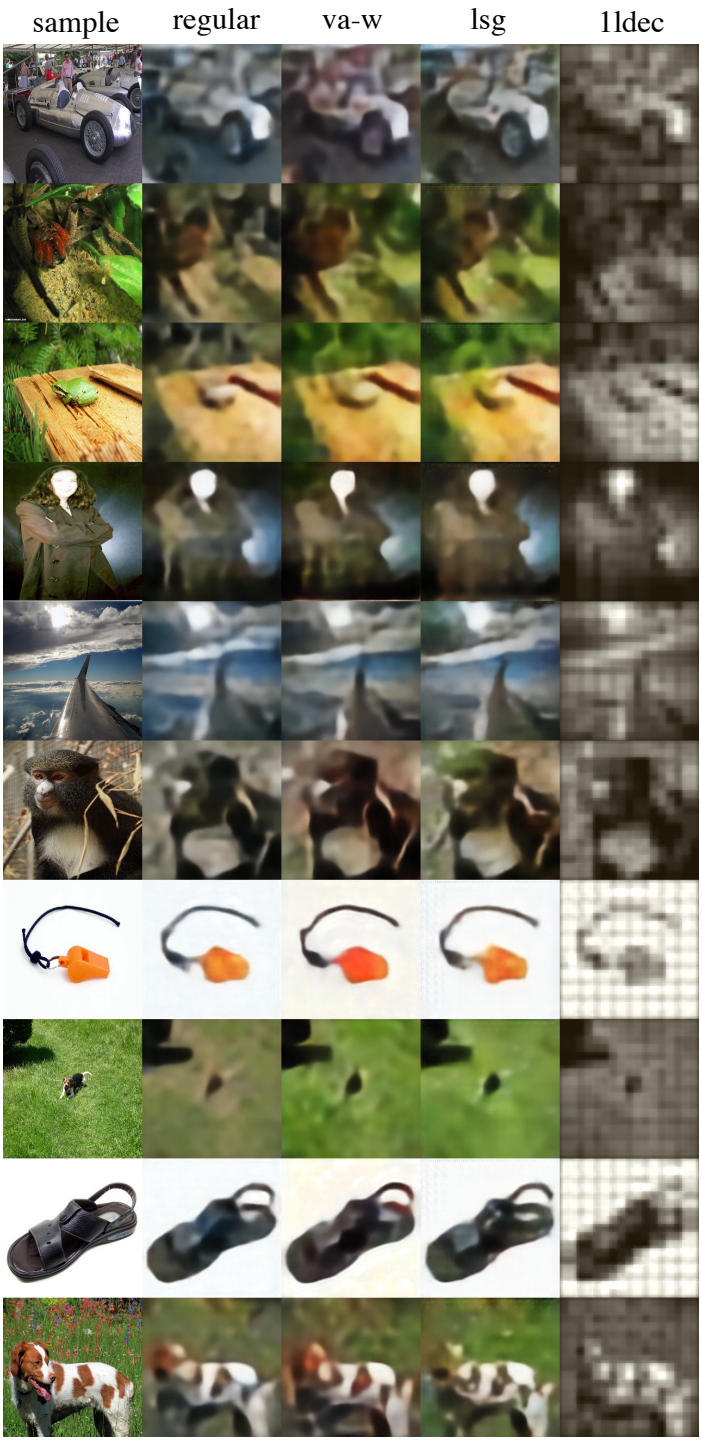

Figure 4.3: Comparison between the reconstruction from different models for a 256 dimensional bottleneck.

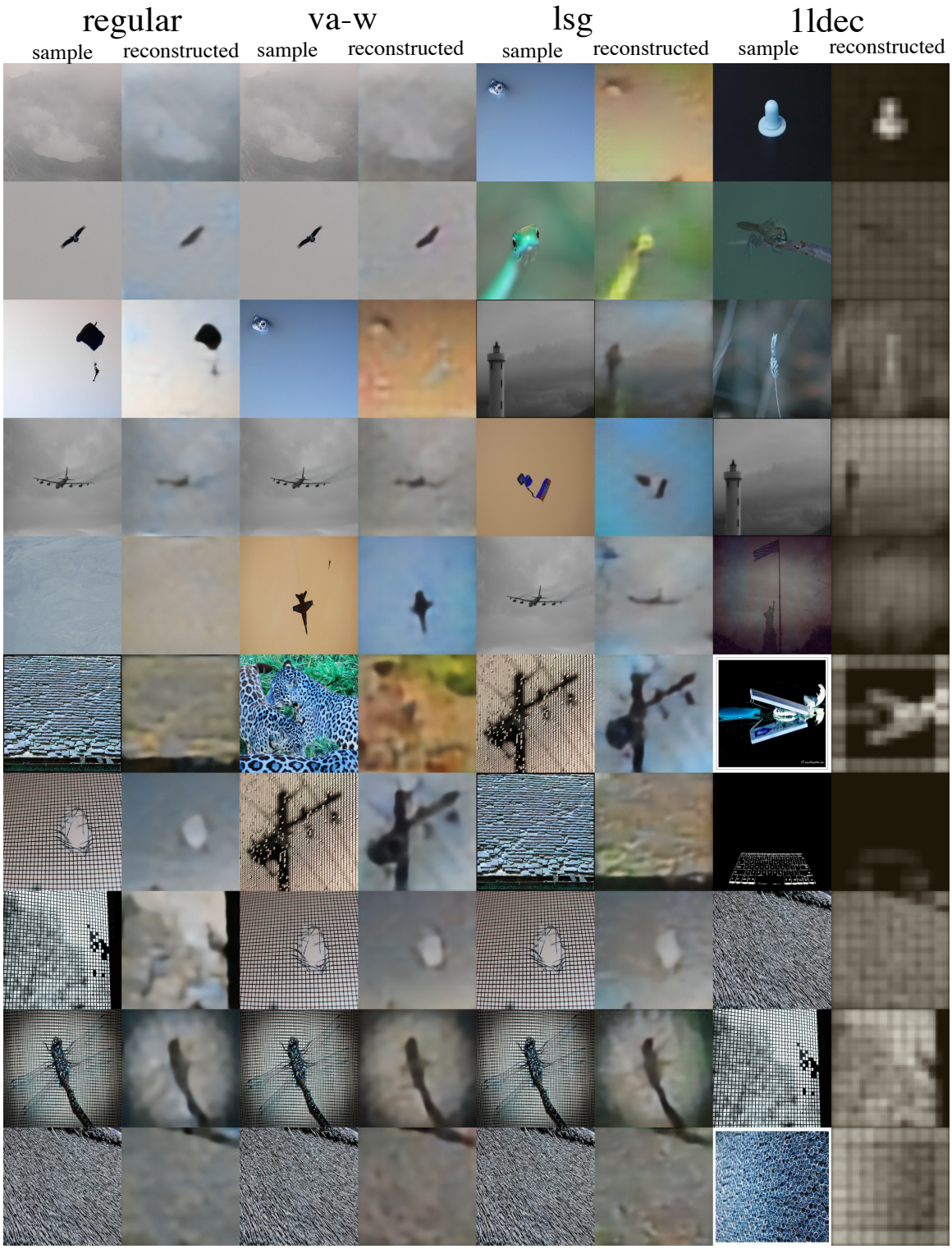

Figure 4.4: The top five best and worst reconstructed images for each model.

Chapter 5

APPLICATION OF UNSUPERVISED FEATURES FOR SUBMODULAR SUMMARIZATION

5.1 Video Summarization

Many of today's intelligent systems regularly generate vast amounts of video data. This data can be either consumer, surveillance, promotional, or commercial videos, or any form of a live internet video stream. It is not feasible for humans to peruse these videos to gain a deep and complete understanding of their content, as they are typically long and redundant. For example, in surveillance settings, most of the video is repetitive, uninformative, and even when motion-triggered, plagued with false positives. This characteristic makes it easy to miss new interesting data when it does finally arrive.

One way to address this problem is to summarize the video content, and hence create an interesting and insightful subset of video "snippets" (i.e., short video segments) for presentation to a human. While a human curator can handcraft summaries, that process itself is time-consuming, error-prone, and costly given the ever-increasing torrent of video data. Therefore, it is of great interest to develop practical methods for automatic video summarization (i.e., to produce summaries that are informative, representative, and much shorter) and that also perform well using standard evaluation methods. Indeed, this problem has gained significant traction in recent years [166, 51, 188, 45, 85].

In general, existing methods approach video summarization from either an *offline* [51] or an *online* [188] perspective. Offline mechanisms require the knowledge of and access to the entire video stream to generate a summary. Such methods, however, require storing the

entire video simultaneously and, therefore, are resource-intensive and, at times, impractical, e.g., for unboundedly long video streams.

Online, or streaming, video summarization methods are an alternative to the above. Given a video stream, an online summarization algorithm produces a summary on the fly and at any time as the data stream elements arrive without using any future information. Such methods can be made to require limited memory since they keep only a small portion of the past video stream (or information thereabout) in memory. Since online methods are computationally cheaper than their batch counterparts, this setting is of particular interest when batch processing a video is too resource consuming on a device, and the application requires access to the historical summary at any given time. It is also useful when the incoming video stream is of unbounded length.

This work focuses on a restricted scenario for online video summarization, where the system has only a fixed (constant) memory budget for retaining the video history. The aim is also to produce a running summary, where a "running summary" is defined as one that, at any given point in time, is representative and informative about the observed video stream from the beginning of the stream up until the current time. The overall video stream itself can potentially be of unbounded duration. This problem can be called *running online video summarization with a fixed memory budget*. Some of the work described here is based on a preprint.

The proposed approach addresses problems that naturally occur in systems with limited resources. Examples include sensor networks where computational resources, including processing power, memory, and network bandwidth, are limited and costly. For instance, suppose a camera sensor with limited memory is placed in an inaccessible environment, and only periodic communication with a control hub is allowed due to limited communications bandwidth. A solution is to perform running online video summarization with a fixed mem-

ory budget and then transmit only the running summaries to the control hub every so often (a transmission requiring a fixed-size batch of information). This procedure reduces memory utilization at the sensor node and the bandwidth usage for communication with the control node. It is also useful when a user wants a historical summary on demand at any time (e.g., video surveillance applications), not just at the end of the stream.

In this work, new algorithms (for single-stream and multi-stream scenarios) are defined for this purpose that perform very well in practice. The algorithms are unique in that they use both an "add gain" (to determine something new) and a "swap gain" (to determine something better) relative to a current summary.

5.1.1 Single Stream Video Summarization

This section focuses mainly on the setup where there is only a single video source that produces the data that needs to be summarized.

Related Work

The problem of video summarization can be categorized into either static or dynamic summarization based on the nature of the selected summary [166]. The goal of static summarization is to choose (from a video) a number of individual images (called key frames) that collectively represent the video. A dynamic summarization scenario focuses on creating a dynamic summary of a video that is composed of chunks of locally contiguous images. In this work, a sequence of such images is called a "*video snippet*".

In the literature, static summarization (also known as key frame selection) is often approached in the offline setting using, say, clustering techniques. In particular, Yu et al. [184] propose selecting key frames of a video using fuzzy "c-mean" clustering of a multi-level representation of the video. Gibson et al. [43], identify key frames in a video based on the

clustering of video frames in an eigenspace via Gaussian mixture models (GMMs). More recently, de Avila et al. [29] use k-means clustering of the color histogram [162] features from video frames to generate static summaries.

Other existing approaches include leveraging information from the web to select key frames [67], and employing supervised techniques (e.g. sequential determinantal point processes [45]) for generating static summaries. Recently, Zhang et al. [186] used LSTMs to design a supervised approach for the selection of key frames or key sub-shots. Mahassen et al. [100] took the unsupervised approach and employed deep adversarial LSTM networks for video summarization. Summarization of egocentric videos has also been an area of interest in recent literature. For example, Lee and Grauman [85] approach it by predicting important objects in a video.

Dynamic video summarization, on the other hand, provides a better viewing experience since each summary snippet is itself a short video segment, thereby yielding true short-interval temporal information. Several mechanisms have been explored for generating dynamic summaries. Snippets can be selected based on shots, specific events, or domains within the video. For instance, Ouyang et al. [131] use the neighborhood of a selected key frame as a snippet. Gong and Liu [46] construct dynamic summaries of a video by concatenating snippets in video shots that are chosen based on their SVD representation of color histogram features. Peyrard and Bouthemy [139] propose to summarize a video by first segmenting it according to camera motion and then selecting snippets based on an event classifier on the segments. Coldefy and Bouthemy [24] study summarization of soccer videos. They propose to generate a summary based on excited speech, motion of the camera, and color features. In Gygli et al. [51], Sun et al. [158], and Herranz and Martinez [54] video snippets are selected according to their quality score for various aspects of a video (e.g., interestingness and representativeness). More recently, Yao et al. [182] explore the summarization of first-person

videos through highlight detection. They split their video segment into spatial and temporal streams. Each stream is fed into a deep convolutional neural network for highlight prediction. The outputs of these networks are fused to generate the highlight score of a video segment. Zhao et al. [187] focus on first extracting shots from a video in an adaptive manner using a bidirectional LSTM and then using a second bidirectional LSTM to assign probabilities to each shot to decide its inclusion in summary. Kanehira et al. [65] explore video summarization in the context of view-point extracted from multiple videos. They focus on summarizing multiple groups of videos based on the view-point extracted from each group. Jin et al. [63] present a mechanism where the user can interact with the summarizer and aid in producing summaries of different sizes from a given video. It augments an offline frame analysis with user input to fast-forward segments of the video. Cai et al. [13] use a VAE pre-trained on web videos in conjunction with an encoder-attention-decoder setup to generate summaries. Rochan and Wang [143] learn a mapping from a set of videos to a set of summaries where there is no direct correspondence between the video set and the summary set. They aim to learn a mapping such that the summaries generated using it have the same distribution as the summary set used in training.

Video summarization has also been examined from an online perspective. For instance, Zhao and Xing [188] select video snippets on the fly based on its reconstruction error from a dictionary, which is generated by the snippets that are already in summary. Almeida et al. [1] consider summarizing compressed videos on the fly. Given a video stream, they add an incoming snippet to the summary if it is sufficiently distinct from already selected snippets. Iparraguirre and Delrieux [58] propose to summarize a video stream in an online fashion by identifying key frames from which video snippets in the summary are constructed. In general, these techniques do not directly fall within the fixed memory budget setting since they do not constrain the size of their summaries. The closest work to the proposed setting is by

Mei et al. [107], which again utilizes a reconstruction-error based summarization technique but abides by a fixed summary budget constraint.

Recently, techniques based on submodular optimization have been applied to video summarization. Gygli et al. [52] learn a mixture of components to model the objective for video summarization. Xu et al. [180] formulate summarization of egocentric videos as constrained maximization of an objective function. Again their approach is in the offline setting. Chakraborty et al. [16] consider adaptive key frame selection for video summarization. They formulate the problem as unconstrained maximization, where the objective is defined on a complete similarity graph, and thus the method is inherently offline. Li et al. [87] propose a framework for summarizing both raw and edited videos. They create a submodular scoring function for a set as a weighted mixture of importance, representativeness, diversity, and "storyness" of that set with respect to a video. They use a supervised max-margin approach to learn the mixture weights and use an accelerated greedy algorithm to produce their summary. Their approach, however, is again offline. On the other hand, Elhamifar and Kalusza [36] use an online summarization approach. However, they focus on unconstrained summarization and are not directly comparable to the approach proposed in this thesis.

The Summarization Framework

In this section, the problem is first formulated as constrained submodular maximization. Next, the summarization objective function is defined. This function particularly suits this task and can be optimized through the proposed online procedure. Finally, the procedure to produce the running summaries is described in detail.

Formulation: Let $V = \{v_1, v_2, \dots\}$ be a video stream (potentially of unbounded duration) with v_t as the constituent snippet for time step t. Let $V_t = \{v_1, v_2, \dots, v_t\}$ be the set of

video snippets that have been observed until time step t. Given a memory budget constraint K, for each time step t the goal is then to produce a summary $S_t \subseteq V_t$ of size no larger than K ($|S_t| \leq K$) such that given a representation criterion, S_t is informative, diverse, and can adequately represent V_t. In the proposed approach the focus is on online summarization and generation of running summaries (of potentially never ending video streams) in a fixed memory budget setting. Therefore, this work attempts to generate good summaries on average based upon a desired scoring function. More formally, let a set function $f : 2^V \rightarrow \mathbb{R}$ represent the quality of any given subset of V. The problem is equivalent to finding S_t for all time t such that the following is achieved:

$$\max_{S_t \subseteq V_t, |S_t| \leq K} \sum_{\tau=1}^{t} f(S_\tau). \tag{5.1}$$

Submodular Objective Function The choice of the representation criterion $f(\cdot)$ plays an important role in determining the quality of the summary. One possible class of functions that have shown potential in the context of summarization is submodular functions. They have shown success in modeling diversity, dispersion, span and coverage in prior investigations of video summarization [52, 180, 16]. For instance, the "facility location" function, i.e.,

$$g_{\text{fac}}(S) = \sum_{v \in V} \max_{s \in S} \omega_{v,s}, \tag{5.2}$$

was applied as the objective in Chakraborty et al. [16], where $\omega_{v,s}$ is the similarity between the video elements (e.g. snippets) v and s. Intuitively, $g_{\text{fac}}(S)$ captures how well the subset S represents the entire data set V. Maximizing $g_{\text{fac}}(\cdot)$ often leads to a collection of representative items. The drawback of this function is that: 1) it is defined on a pairwise similarity graph, which requires both $O(|V|^2)$ memory and compute; 2) evaluating the

function requires access to the entire data V. Therefore the usage of this function in the context of this work (running online summarization) is not possible. Fortunately, the class of submodular functions called 'feature based functions' (equation 1.1) does not need access to the whole data for evaluating a set. For the generation of running video summaries, this work utilizes a modified version of the feature based function in the following form:

$$g_{\text{fea}}^{\max}(S) = \sum_{u \in U} w_u \max_{s \in S} m_u(s), \tag{5.3}$$

where the concave functions $\phi_u(\cdot)$ in the feature based function is replaced with $\max(\cdot)$. Maximizing $g_{\text{fea}}^{\max}(\cdot)$ leads to diverse coverage over the whole feature set U in the selected summary, with the desired coverage of each feature $u \in U$ being controlled by its non-negative weight w_u. Furthermore, this can firmly reject redundancy in the resultant summary since it ensures that no additional gain is achieved when adding identical snippets into a summary.

It is important to note that the function $g_{\text{fea}}^{\max}(\cdot)$ also has the AC characteristic. Therefore, an increase in the property u of the data object cannot cause the evaluation of $g_{\text{fea}}^{\max}(\cdot)$ to decrease. It either remains the same or increases to a higher value.

In addition to $g_{\text{fea}}^{\max}(\cdot)$, it might be desirable for the summary to have certain predetermined preferences. In this case, the overall objective is defined in the following form:

$$f(S) = g_{\text{fea}}^{\max}(S) + \sum_{l} \rho_l q_l(S) \tag{5.4}$$

where $q_l(\cdot)$ is a modular function (that acts as a quality function) to define a given preference and ρ_l is a non-negative weight associated with $q_l(\cdot)$. These characteristics are dependent upon the environment in which the video summarization system is deployed. The preferences can be towards various factors, such as faces, specific people or objects, specific

places, aesthetics, and many more [168, 189].

For instance, a preference can be introduced in the overall objective function that encourages the summarization procedure to select snippets that have faces in it. A simple variant can be based on face detection [189] in the video. In this case, the modular quality function has the following form:

$$d(S) = \sum_{s \in S} d(s), \tag{5.5}$$

where $d(s) = 1$ if the snippet 's' has a face in it and zero otherwise.

Another example of a quality function is one that encourages summaries of long video streams to be diverse in time. First, divide the whole video V into "p" (independent of the size of V) sized bins, where the i^{th} bin C_i consists of a set of snippets $\{v_j\}_{j=(i-1)p+1}^{ip}$, i.e., C_i consists of a sequence of snippets that span from the time step $(i-1)p+1$ to ip, and any pair of bins are disjoint, i.e., $C_i \cap C_j = \emptyset \;\; \forall \; i \neq j$. Let n be the total number of bins in the whole video V, then the time diversity function can be defined as $h : 2^V \rightarrow \mathbb{N}$ in the following manner:

$$h(S) = \sum_{i=1}^{n} \min\{|S \cap C_i|, 1\} \tag{5.6}$$

This function evaluates any subset $S \subseteq V$ as the number of bins that are spanned by this set. Note that one does not need the knowledge of the entire data V (or its length) to evaluate this function in the context of online summarization, since, at any given time t, the summary $S_t \subseteq V_t$ does not span any bin corresponding to the future time steps, i.e., it holds that $\min\{|S \cap C_i|, 1\} = 0$ for all $i > \lceil \frac{t}{p} \rceil$. The time diversity function $h(\cdot)$ valuates a summary higher if the constituent snippets span a higher number of bins.

These are just some of the possible examples of introducing preferences in the objective function. Systems that are deployed in the field can use much more sophisticated tools. These tools will fit in the proposed framework as long as they produce a modular score.

Procedure for Generation of Running Summaries The problem of streaming video summarization in a fixed memory budget along with the generation of running summaries at each time step has several restrictions. A procedure that addresses this problem needs to have the following properties: **1)** at each time step, the generated summary must fit in the given budget; **2)** at any point in time, any stored snippet must be part of the summary; **3)** if an incoming snippet is not introduced into the summary, then it is forever dropped and is considered lost (the procedure no longer has access to it) and **4)** the total length of the video stream is unknown.

Given the submodular objective $f(\cdot)$, a streaming procedure needs to be defined that utilizes $f(\cdot)$ to generate the running summaries. As shown in the seminal paper by Nemhauser et al. [124], the problem defined in formulation 5.1 can be efficiently and near-optimally solved with a constant approximation guarantee of $(1 - \frac{1}{e})$ using a greedy algorithm in the offline setting. Recently, a number of streaming algorithms have been proposed for solving this problem in the online setting. In particular, Badanidiyuru et al. [5] present an efficient algorithm that optimizes Problem 5.1 on the fly while achieving a constant approximation guarantee of $(\frac{1}{2} - \epsilon)$. Unfortunately their approach requires $O(\frac{K \log K}{\epsilon})$ memory where K is the budget of the summary. Hence it is not ideal in the restricted scenario used in this work, where it is required that the procedure stores no more than K items at any given time. A variant of [5] by Kazemi et al. [66] proposes a modified approach that has the same approximation guarantee $(\frac{1}{2} - \epsilon)$ and produces similar results [66], but uses $O(\frac{K}{\epsilon})$ memory. Therefore, even if it is more memory efficient than [5], it is still not ideal in the scenario

Algorithm 1: Procedure for Generation of Running Summaries

Input: δ, α L, K, C, and $V = \{v_1, v_2, \dots\}$.
Output: A running summary S_t for every time step t.
Initialize: $S_0 = \emptyset$, $c_0 = 0$.
for $t = 1, \dots$ **do**
    $g_t \leftarrow f(S_{t-1} \cup v_t) - f(S_{t-1})$;
    if $|S_{t-1}| < K$ **then**
        if $g_t \geq \alpha$ **then**
            $S_t \leftarrow S_{t-1} \cup v_t$;
        else
            $S_t \leftarrow S_{t-1}$;
    else
        $c_t \leftarrow \frac{1}{L} \sum_{\tau=t-L}^{t-1} g_\tau$; $\gamma_t \leftarrow c_t + \delta$;
        $\hat{s} \in \text{argmax}_{s \in S_{t-1}} f(S_{t-1} \cup v_t \setminus s) - f(S_{t-1})$
        $\hat{g}_t \leftarrow f(S_{t-1} \cup v_t \setminus \hat{s}) - f(S_{t-1})$
        if $\hat{g}_t \geq C\frac{f(S_{t-1})}{K}$ *or* $g_t \geq \gamma_t$ **then**
            $S_t \leftarrow S_{t-1} \cup v_t \setminus \hat{s}$
        else
            $S_t \leftarrow S_{t-1}$

focused on in the proposed work.

Buchbinder et al. [12] give a swapping-based streaming algorithm that outputs a near-optimal running summary at every time step while abiding by a fixed budget K. Given a stream of data items, this algorithm unconditionally selects the first K items into the summary. Afterwards, the algorithm maintains a summary S_t of size K at every time step t, and swaps the new item v_{t+1} with an existing item in the summary if the maximum gain in the objective is beyond a threshold. Buchbinder et al. [12] suggest setting the threshold for each given time t as $C\frac{f(S_t)}{K}$, where C is a constant. Moreover, they show that such a scheme produces a near-optimal running summary S_t for any step t as follows: $f(S_t) \geq \frac{C}{(C+1)^2} \max_{|S| \leq K, S \subseteq V_t} f(S)$. Note that the approximation factor is maximized as $1/4$ when

$C = 1$. This thesis call this algorithm 'ThreshStream' and the one by Badanidiyuru et al. [5] as 'SieveStream'.

A new method better suited to video summarization is introduced in this thesis. This modified algorithm (Alg. 1) consists of two stages: (1) adding stage ($|S_t| < K$), and (2) swapping stage ($|S_t| = K$). In the adding stage, the procedure examines each incoming snippet v_t and decides on its inclusion in summary by analyzing the gain of adding v_t to S_{t-1}. The snippet is added to the summary if the adding gain $g_t = f(v_t|S_{t-1})$ is no less than the threshold α. The setting where $\alpha > 0$ prevents the unconditional selection of the first K video snippets in the data stream if the beginning of the video stream is highly redundant. The adding stage finishes when the summary size reaches the budget of K. Note that setting $\alpha = 0$ in the proposed approach reduces to the unconditional selection of the first K items, as is done in ThreshStream. Hence, the adding stage ensures that the initial K snippets are neither arbitrary nor redundant. Filling up the initial summary budget using an arbitrary or redundant snippet, as in ThreshStream, is wasteful and can take some time to recover from, especially when one wants a running summary. It is also important to note that the adding stage is active till the summary reaches the budget. Therefore, for highly diverse or very long streams, the adding stage might be in operation for a relatively small duration (based on the budget). Furthermore, for extremely long streams, the summary snippets towards the end might be very different from the ones during the adding stage as the summary needs to capture any new information in the video stream while remaining inside the budget.

In the swapping stage, a new snippet is added to the summary by swapping it with an existing snippet. Given a time step t, let $\hat{g}_t$ denote the gain of swapping the incoming element v_t with an existing item $\hat{s} \in S_{t-1}$. Here $\hat{s}$ is the snippet from S_{t-1} that provides the maximum swapping gain. The new item v_t is swapped in if either the maximum swapping gain $\hat{g}_t$ exceeds the threshold $C\frac{f(S_{t-1})}{K}$, or (and the novel contribution) the adding gain g_t

exceeds another threshold γ_t. In either case, the new element v_t is swapped with the item $\hat{s} \in S_{t-1}$ that provides the maximum swapping gain. A large maximum swapping gain $\hat{g}_t$ implies that the benefit of adding the new snippet outweighs the loss of removing an item from the summary.

The adding gain being large enough triggers a swap whenever the new element is highly distinct from any current snippet in S_{t-1}. This use of adding gain is critically different from the swapping stage in ThreshStream. When the budget is fixed, there might be no way to improve the objective via a swap (indeed, it is possible that the summary might already be at the optimal set of size K). The adding gain trigger does not attempt to improve the objective, Rather, it triggers a swap to ensure that something new and good is added to the summary while allowing something old, or at least not too good, to be removed. This strategy places a preference on novelty at the potential expense of the objective value. The intuition for this is that with a running summary, the aim is to avoid a situation where a summary, being highly scored by the objective, becomes temporally stale. For example, it can be that the first K elements are optimal, but in a running summarization setting, the first K elements might not make a good summary for all time, even if they have a high score (based on $f(\cdot)$).

The tradeoff between the swapping gain and the adding gain that triggers a swap is determined by the adaptive threshold γ_t, which is computed based on the moving average of g_t over the past L time steps. This allows the thresholds to adjust automatically and smoothly based on the video stream. The swapping stage in the modified variant reduces to ThreshStream when $\gamma_t = \infty$ (or equivalently, $\delta = \infty$), in fact:

Proposition 5.1.1 *For $\alpha = 0$ and $\delta = \infty$, Algorithm 1 will also generate S_t such that $f(S_t) \geq \frac{C}{(C+1)^2} \max_{|S| \leq K, S \subseteq V_t} f(S)$. The approximation factor is maximized as $\frac{1}{4}$ for $C = 1$ [12].*

Experimental Setup

The Data Set The performance of various summarization methods is compared on the videos from SumMe [51] and TVSum50 [156] data sets. The SumMe data set consists of 25 videos. Each video has a set of human generated summaries (the ground truth) from multiple users. Each summary consists of a set of variable length video snippets. A detailed description of the data is given in Gygli et al. [51]. The TVSum50 data set contains 50 videos in 10 categories. The videos were collected from YouTube using the category as the search query. Each video has a set of associated ground truth summaries. A detailed description of the data set is provided in Song et al. [156].

Instantiation of Objective Function The features that are used to instantiate the objective $f(\cdot)$ play an important role in identifying the focus of the summarization process. Previous work has shown that the features produced using DNNs perform well in a number of computer vision tasks [82, 4, 76, 159]. Therefore, in this work $g^{\max}_{\text{fea}}(\cdot)$ is instantiated on a set U of deep features that are generated using a Deep Neural Network (DNN).

The features are produced through a two-step process. First, an autoencoder is trained using the ILSVRC 2012 data set [147]. This trained model is then used to produce fist level features for individual video frames (each frame is an image). Next, these first level features (output of the encoder) from a snippet's constituent frames are combined to train another autoencoder model to produce snippet level features. In the case of AC features, both these models are trained with the AC constraint. The snippet models for SumMe [51] and TVSum50 [156] are trained separately and independently from each other. Table 5.1 shows the autoencoder architecture used for the first level features. Table 5.2 is the architecture used for snippet level features. The snippet features are used to instantiate $g^{\max}_{\text{fea}}(\cdot)$ (equation 5.3) with $w_u = 1$ for all $u \in U$.

In addition to this, a quality function is utilized in the objective function. This function is defined via a Haar feature-based cascade classifier [170] for face detection. More formally, given a video snippet v, the quality function $q(v)$ is either 1 or 0, depending on the presence of a face in the video snippet as predicted by the face detector.

Table 5.1: Neural network structure of the autoencoder used for first level feature production.

Group	Block Type (kernel sz, stride, channels)	# Blocks
conv1	$[\ 3 \times 3\ ]$, 2, 64	1
conv1 (residual)	$\begin{bmatrix} 3 \times 3 \\ 3 \times 3 \end{bmatrix}$, 1, 64	3
conv2	$[\ 3 \times 3\ ]$, 2, 16	1
conv2 (residual)	$\begin{bmatrix} 3 \times 3 \\ 3 \times 3 \end{bmatrix}$, 1, 16	3
conv3	$[\ 3 \times 3\ ]$, 4, 8	1
conv3 (residual)	$\begin{bmatrix} 3 \times 3 \\ 3 \times 3 \end{bmatrix}$, 1, 8	3
conv4	$[\ 3 \times 3\ ]$, 1, 1	1
conv4 (residual)	$\begin{bmatrix} 3 \times 3 \\ 3 \times 3 \end{bmatrix}$, 1, 1	1
deconv4 (residual)	$\begin{bmatrix} 3 \times 3 \\ 3 \times 3 \end{bmatrix}$, 1, 1	1
deconv3	$[\ 3 \times 3\ ]$, 1, 8	1
deconv3 (residual)	$\begin{bmatrix} 3 \times 3 \\ 3 \times 3 \end{bmatrix}$, 1, 8	3
deconv2	$[\ 3 \times 3\ ]$, 4, 16	1
deconv2 (residual)	$\begin{bmatrix} 3 \times 3 \\ 3 \times 3 \end{bmatrix}$, 1, 16	3
deconv1	$[\ 3 \times 3\ ]$, 2, 64	1
deconv1 (residual)	$\begin{bmatrix} 3 \times 3 \\ 3 \times 3 \end{bmatrix}$, 1, 64	3
deconv0	$[\ 3 \times 3\ ]$, 2, 3	1

Table 5.2: Neural network structure of the autoencoder used for snippet feature production in single-stream summarization.

Group	Block Type (kernel sz, stride, channels)	# Blocks
conv1	$[\ 3 \times 3\ ]$, 1, 10	1
conv1 (residual)	$\begin{bmatrix} 3 \times 3 \\ 3 \times 3 \end{bmatrix}$, 1, 10	2
conv2	$[\ 3 \times 3\ ]$, 2, 5	1
conv2 (residual)	$\begin{bmatrix} 3 \times 3 \\ 3 \times 3 \end{bmatrix}$, 1, 5	2
conv3	$[\ 3 \times 3\ ]$, 3, 2	1
conv3 (residual)	$\begin{bmatrix} 3 \times 3 \\ 3 \times 3 \end{bmatrix}$, 1, 2	1
deconv3 (residual)	$\begin{bmatrix} 3 \times 3 \\ 3 \times 3 \end{bmatrix}$, 1, 2	1
deconv2	$[\ 3 \times 3\ ]$, 1, 5	1
deconv2 (residual)	$\begin{bmatrix} 3 \times 3 \\ 3 \times 3 \end{bmatrix}$, 1, 5	2
deconv1	$[\ 3 \times 3\ ]$, 2, 10	1
deconv1 (residual)	$\begin{bmatrix} 3 \times 3 \\ 3 \times 3 \end{bmatrix}$, 1, 10	2
deconv0	$[\ 3 \times 3\ ]$,1, 20	1

Evaluation Mechanism: Running F score It is common to use F-measure based evaluation techniques for judging an automatically generated video summary [45, 51, 156, 52, 180, 186, 100]. Similarly, to quantitatively test the performance of running summaries, this work employs an F-measure based evaluation method that takes the quality of the summary generated for every time step into account, instead of examining only the end summary of a video. Given a video $V = \{v_1, \ldots, v_n\}$ consisting of a set of snippets, let $b(v_i)$ denote the number of frames in the video snippet v_i and $c(v_i)$ the number of intersecting frames between the human summary and the snippet v_i. Thereby, the quality of the summary at time step t relative to the human generated summary can be evaluated through the F-measure F_t, which

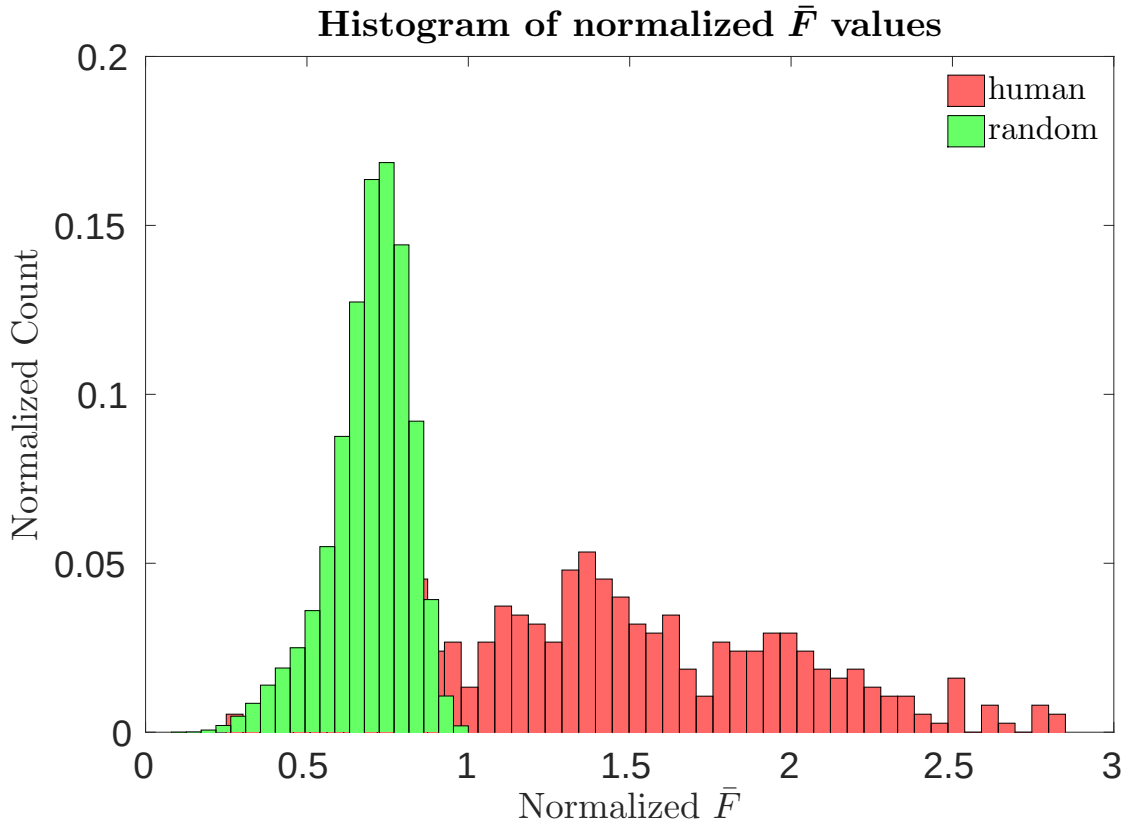

Figure 5.1: Qualitative validation of $\bar{F}$ as the evaluation mechanism.

is defined as follows: $P_t = \frac{\sum_{v \in S_t} c(v)}{\sum_{v \in S_t} b(v)}$, $R_t = \frac{\sum_{v \in S_t} c(v)}{\sum_{v \in V_t} c(v)}$, $F_t = 2\frac{P_t R_t}{P_t + R_t}$, where P_t and R_t are precision and recall at time step t. In the case of multiple human summaries, these quantities are calculated by averaging the scores obtained using individual human summaries. The performance of the online summarization method for producing running summaries from the video V is defined as:

$$\bar{F} = \frac{1}{n} \sum_{t=1}^{n} F_t \tag{5.7}$$

The quality of $\bar{F}$ is empirically gauged using the SumMe data set. It was done as follows. First, compute $\bar{F}$ of each human summary (the ground truth) for individual videos. Each

human summary was compared with every other human summary associated with the same video. Second, compute $\bar{F}$ for 5,000 randomly generated summaries for each video. Next, for each video, normalize the computed $\bar{F}$ by the maximum value over all random summaries for this video. This normalization helps in comparing the performance among videos as the range of $\bar{F}$ values can vary based on the video. Lastly, the histogram of these normalized $\bar{F}$ values is plotted in Figure 5.1. Note that the random summaries have a normalized score upper bounded by 1 as a result of the normalization. Figure 5.1 depicts that the human summaries often score much higher than the random summaries under $\bar{F}$ lending credence to the usage of this evaluation mechanism.

Results The performance of the proposed video summarization procedure for AC and non-AC features is explored using the SumMe [51] and TVSum50 [156] data sets.

The video summarization problem being studied in this work is to generate "running summaries" under constraints (such as bounded memory usage). It has practical importance, and the online nature is just one aspect of the proposed approach. Most video summarization methods (including online variants) do not focus on this exact problem. Therefore, it is difficult (and arguably unfair) to use them as comparisons. Any comparable procedure should satisfy the following properties: **A**) it can operate in an online manner; **B**) it can operate using a fixed limited budget for any arbitrary stream length; **C**) it can generate a running summary conforming to this budget; **D**) the total length of the video stream is unknown or unbounded; **E**) it does not store extra snippets apart from those present in summary at any given time. In the proposed approach, any snippet not in summary is considered lost and is no longer accessible to the summarization procedure. These conditions restrict the options for baseline methods. It is rare for a method to consider the generation of such running summaries.

The proposed video summarization procedure is compared to baseline methods that use a submodular function as their summarization objective. Furthermore, even though SieveStream by Badanidiyuru et al. [5] uses more memory than the proposed procedure, it uses a submodular objective and has a mathematical performance guarantee. To achieve a good guarantee, low ϵ values are used for SieveStream. The performance is also compared with Buchbinder et al. [12] (ThreshStream). Both SieveStream and ThreshStream (instantiated using AC features) act as strong baselines for comparison with the proposed approach.

For each video in SumMe and TVSum50 the size of a video snippet is fixed to 2 seconds (as suggested in TVSum50). Figures 5.2 and 5.3 demonstrate that the proposed single stream video summarization, with the submodular objective instantiated using AC features can outperform the baselines. Furthermore, the same procedure has better performance using AC features as compared with their non-AC counterpart.

5.1.2 Multi-Stream Video Summarization

This section focuses on the setup where multiple video sources produce data in parallel (and synchronously). These numerous and concurrent streams need to be summarized to produce running summaries at each time step. The summaries aim to extract the combined information from all the video streams while abiding by a fixed memory budget. The goal is to produce such summaries of the past at each time step ('running summaries'). Such a setup has practical importance in scenarios such as sensor and camera networks (e.g., Figure 5.4) where not all nodes have the resources to process the data in the network.

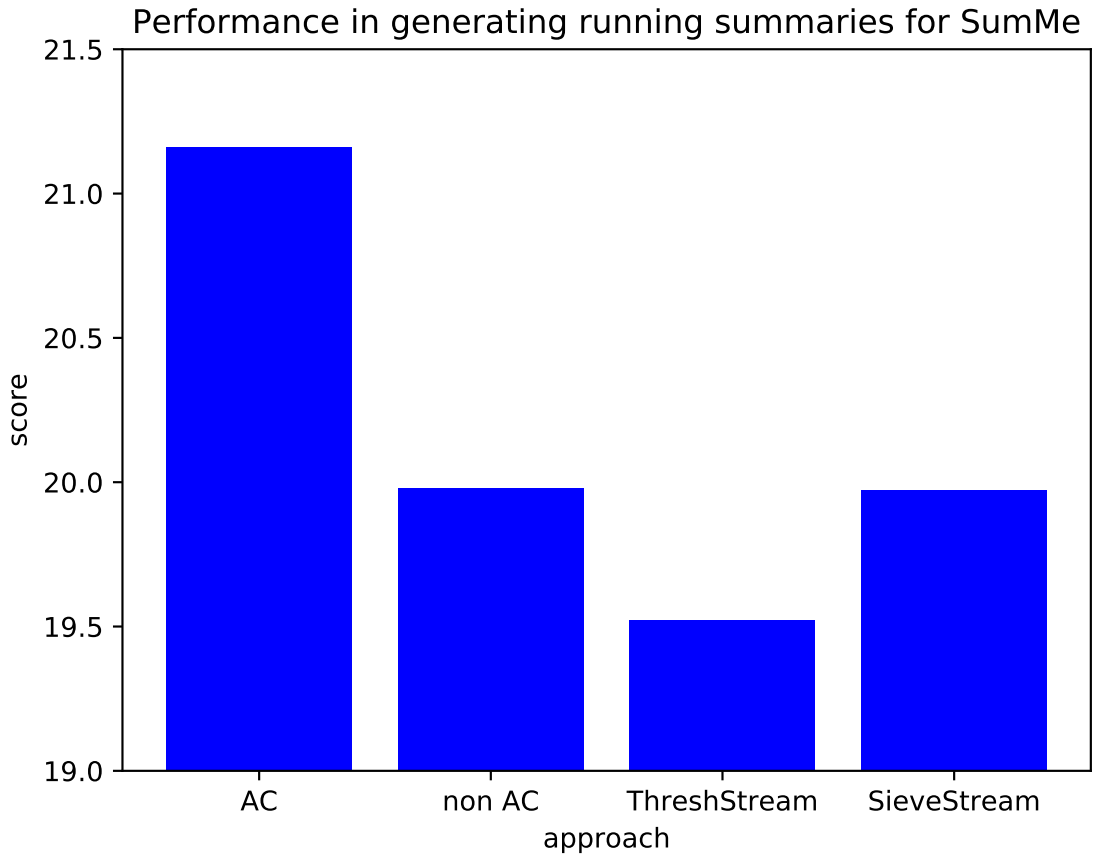

Figure 5.2: Comparison of the performance ($100 \times \bar{F}$ averaged over all videos) of the single-stream video summarization procedure with baselines on the SumMe [51] data set. The performance of the proposed procedure for both AC and non-AC features is also depicted.

Related Work

Although not as common as the single-stream setup, multi-view video summarization has been explored in literature. Fu et al. [39] adapt a shot-graph based approach and summarize the videos as a graph labeling task. Li et al. [86] utilize a correlation map to model correlations with different attributes of key-frames. A rough set based approach is then used for summarization. Panda et al [132] use a stochastic feature embedding approach in conjunc-

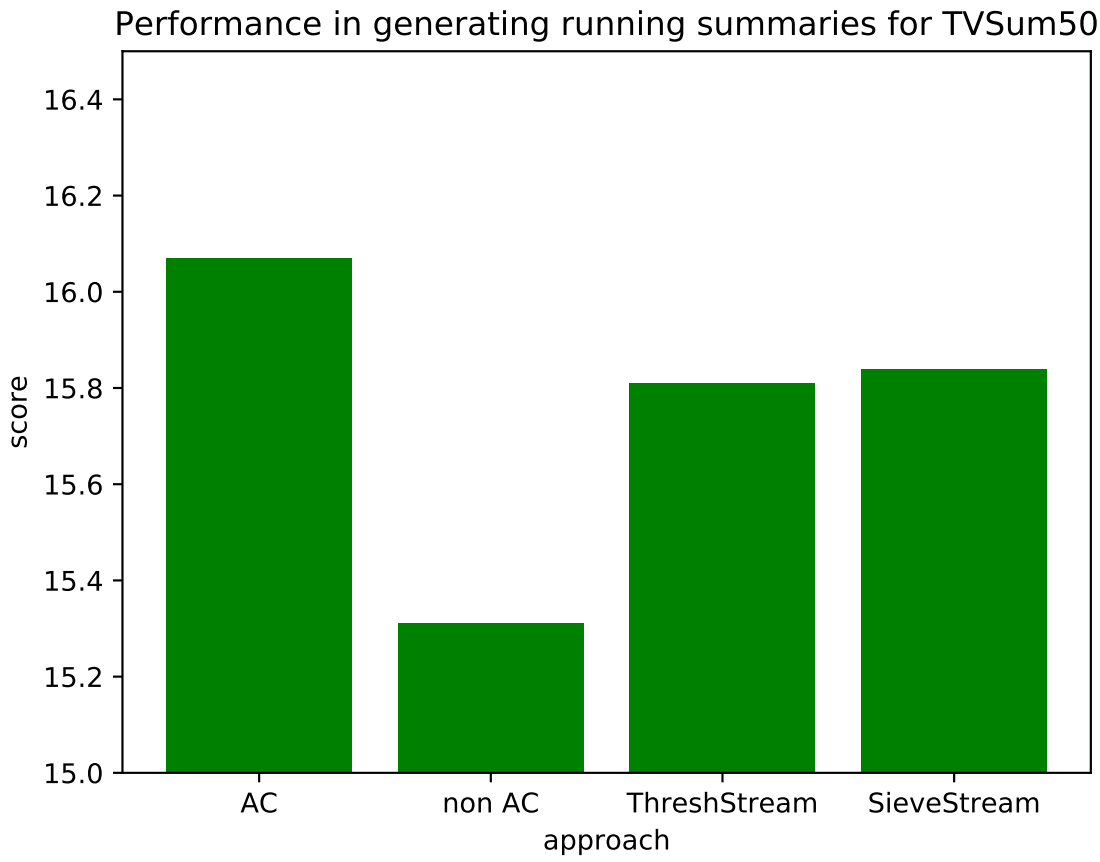

Figure 5.3: Comparison of the performance ($100 \times \bar{F}$ averaged over all videos) of the single-stream video summarization procedure with baselines on the TVSum50 [156] data set. The performance of the proposed procedure for both AC and non-AC features is also depicted.

tion with sparse coding to summarize the videos. Panda et al. [133] also explore the use of both inter-view and intra-view correlation in a joint embedding space. A sparse representative selection approach over the learned joint embedding is then employed to summarize the videos. These approaches are offline and do not directly conform to the focus problem in the proposed multi-stream summarization setup.

Online approaches have also been explored for multi-view video summarization. Ou et al. [130] provide an online approach through a two-step process. First, a gaussian mixture

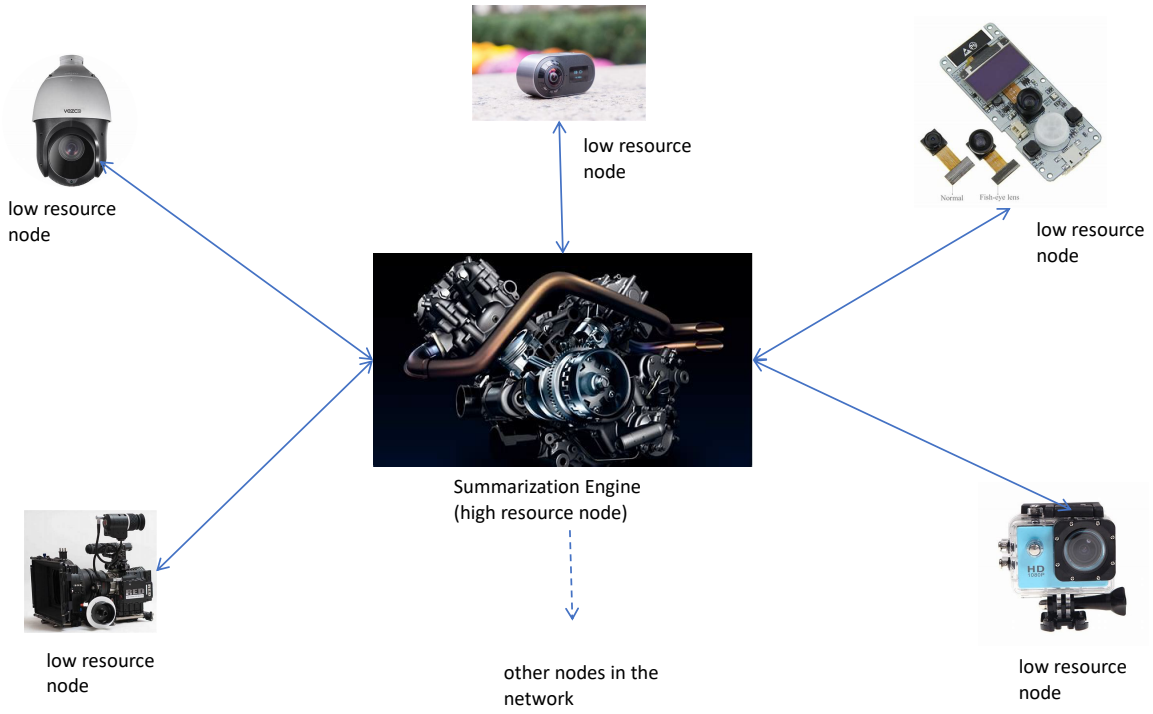

Figure 5.4: A camera network where not all nodes have the resources to process the information flowing in the network.

model (GMM) [157] based online clustering approach is used to cluster frames in each video stream. The frames from this intra-view stage are then processed by the inter-view stage that uses content matching and view-selection to produce the summarized stream. However, there is no restriction of a fixed budget. Furthermore, additional buffers are maintained for each view containing the corresponding historical frames. Therefore, once again, it does not directly conform to the focus problem of the proposed work.

From a submodular perspective, Elfeki et al. [35] use a multi-stream adaptation of determinantal point process (DPP) [99, 77, 78] called Multi-DPP. They use a pre-trained convolutional neural network and a bidirectional LSTM layer to extract spatiotemporal features from each frame of each view. They use a classification framework to determine the quality

of each view at each time step. This information is used by the Multi-DPP to determine which events are selected in summary at each time step. The overall process is offline.

The batch sieve-streaming++ [66] (BSSP) approach can also be used for multi-stream video summarization (with synchronous, overlapping views). This approach maintains multiple candidate summaries at each time step, and the selected candidate is the one that has the highest evaluation based on the submodular objective. These candidate summaries are slowly discarded with time, based on an adaptive threshold. At each time step, new incoming snippets are added to a shared buffer (for all streams), and candidate snippets that are introduced into the summaries are sampled from this buffer. Although this procedure can be used for multi-stream summarization, it has a higher memory requirement than the proposed approach.

The Summarization Framework

In this section, the problem is first formulated as a constrained submodular maximization. Next, the summarization objective function is described. Finally, a multi-stream video summarization procedure is proposed to generate running summaries.

Formulation: Let $V^\ell = \{v_1^\ell, v_2^\ell, \dots\}$ be the ℓ^{th} video stream (potentially of unbounded duration) with v_t^ℓ as the constituent snippet for time step t. Let $V_t^\ell = \{v_1^\ell, v_2^\ell, \dots, v_t^\ell\}$ be the set of video snippets from V^ℓ that have been observed until time step t. Let $\mathscr{V} = \cup_\ell V^\ell$. Let $\mathbb{V}_t = \cup_\ell v_t^\ell$ be all the snippets that are observed at time t. Furthermore, let $\mathscr{V}_t = \cup_\ell V_t^\ell$ be all the snippets from all the streams that have been observed until step t. Therefore, $\mathscr{V}_t = \cup_{\zeta \le t} \mathbb{V}_\zeta$.

Given a memory budget constraint K, for each step t, the goal is to produce a summary $S_t \subseteq \mathscr{V}_t$ of size no larger than K (i.e., $|S_t| \le K$) such that given a representation criterion, S_t

is informative, diverse, and can adequately represent $\mathscr{V}_t$. Similar to the single-stream setup of Section 5.1.1, the proposed multi-stream approach focuses on online summarization and generation of running summaries in a fixed memory budget setting. Therefore, this work attempts to generate good summaries on average based upon a desired scoring function. Hence, if a set function $f : 2^{\mathscr{V}} \rightarrow \mathbb{R}$ represents the quality of any given subset of $\mathscr{V}$, the problem is equivalent to finding S_t for all time t such that the following is achieved:

$$\max_{S_t \subseteq \mathscr{V}_t, |S_t| \leq K} \sum_{\tau=1}^{t} f(S_\tau). \tag{5.8}$$

Submodular Objective Function The characteristics of the single-stream summarization objective function (Section 5.1.1) are also useful for multi-stream summarization. Therefore, the multi-stream summarization procedure uses a similar objective defined as:

$$g_{\text{fea}}^{\text{maxMS}}(S) = \sum_{u \in U} w_u \max_{s \in S} m_u(s), \tag{5.9}$$

Procedure for Generation of Running Summaries from Multiple Streams The multi-stream procedure needs to have characteristics that are similar to the single-stream procedure. These properties are: **1)** at each time step, the generated summary must fit in the given budget; **2)** at any point in time, any stored snippet must be a part of the summary; **3)** if an incoming snippet is not introduced into the summary, then it is forever dropped and is considered lost (the procedure no longer has access to it) and **4)** the total length of the video streams is unknown.

The proposed multi-stream summarization algorithm (Alg. 2) consists of three stages: 1)

Algorithm 2: Multi-Stream Summarization with a Fixed Memory Budget

1 **Input:** f, α, β, δ, η, ℓ, m, k, C, and $\mathscr{V} = \{\mathbb{V}_1, \mathbb{V}_2, \dots\}$, where $|\mathbb{V}_t| \leq m$
2 **Output:** A running summary S_t for every time step t.
3 **Initialize:** $S_0 = \emptyset$, $\hat{\mathfrak{X}} = \emptyset$, $c_0 = 0$.
4 **for** $t = 1, \dots$ **do**
5 $\hat{\mathfrak{X}} = \emptyset$
6 **for** $i = 1, \dots, |\mathbb{V}_t|$ **do**
7 $e^* \in \operatorname{argmax}_{e \in \mathbb{V}_t \setminus \hat{\mathfrak{X}}} f(e | S_{t-1} \cup \hat{\mathfrak{X}})$
8 **if** $f(e^* | S_{t-1} \cup \hat{\mathfrak{X}}) \geq \eta$ **then**
9 $\hat{\mathfrak{X}} \leftarrow \hat{\mathfrak{X}} \cup e^*$
10 **if** $|S_{t-1}| < k$ **then**
11 $\hat{\mathscr{Y}} \in \operatorname{gargmax}_{Y \subseteq \hat{\mathfrak{X}} \cup S_{t-1}, S_{t-1} \subseteq Y, |Y| \leq k} f(Y)$
12 $g_t \leftarrow f(\hat{\mathscr{Y}} | S_{t-1})$
13 **if** $g_t \geq \alpha$ **then**
14 $S_t \leftarrow \hat{\mathscr{Y}}$
15 **else**
16 $S_t \leftarrow S_{t-1}$
17 **else**
18 $c_t \leftarrow \frac{1}{L} \sum_{\tau=t-L}^{t-1} g_\tau$; $\gamma_t \leftarrow c_t + \delta$
19 $\hat{\mathfrak{X}} \in \operatorname{gargmax}_{Z \subseteq \hat{\mathfrak{X}} \cup S_{t-1}, |Z| = k} f(Z)$
20 $g_t \leftarrow f(\hat{\mathfrak{X}} | S_{t-1})$
21 $\hat{g}_t \leftarrow f(\hat{\mathfrak{X}}) - f(S_{t-1})$
22 **if** $\hat{g}_t \geq \beta \frac{f(S_{t-1})}{k}$ *or* $g_t \geq \gamma_t$ **then**
23 $S_t \leftarrow \hat{\mathfrak{X}}$
24 **else**
25 $S_t \leftarrow S_{t-1}$

pruning stage, 2) adding stage ($|S_t| < K$), and 3) swapping stage ($|S_t| = K$).

The goal of the pruning stage is to create a candidate set $\hat{\mathfrak{X}}$ at each step t from the set $\mathbb{V}_t$, the combined set of incoming snippets from all streams. $\hat{\mathfrak{X}}$ is created by iteratively filtering out snippets from $\mathbb{V}_t$ that do not provide gain above a threshold over S_{t-1}. More specifically, a snippet selected in Line 7 is added to $\hat{\mathfrak{X}}$ only if the gain of adding it to $S_{t-1} \cup \hat{\mathfrak{X}}$ is greater

than or equal to a pruning threshold η. This pruning prevents wasteful computation with redundant snippets in the later stages of the summarization procedure.

In the adding stage, the procedure examines the candidate set $\hat{\mathfrak{X}}$ and produces a set $\hat{\mathcal{Y}}$ via a greedy procedure ('gargmax' is the sequence produced using the greedy procedure such as [110]). Snippets from $\hat{\mathcal{Y}} \setminus S_{t-1}$ are added to S_{t-1} if the gain of adding these snippets is greater than or equal to an adding threshold of α. The adding stage ensures that the initial K snippets are neither arbitrary nor redundant.

In the swapping stage, new snippets are added to the summary by swapping them with existing summary snippets. For a time step t, let $\hat{g}_t$ denote the maximum gain of swapping in snippets from the candidate set $\hat{\mathfrak{X}}$ with existing summary snippets. These new snippets can be swapped in if the swapping gain $\hat{g}_t$ is equal to or exceeds threshold $\beta\frac{f(S_{t-1})}{K}$, or the adding gain g_t exceeds another threshold γ_t. A large swapping gain $\hat{g}_t$ implies that the benefit of adding the new snippets outweighs the loss of removing the corresponding existing snippets from the summary. The swap due to a large adding gain is triggered when the new snippets represent something very distinct from the information currently contained within the summary. For example, consider the scenario where all summary snippets are sufficiently different from each other, but none of them contains information conveyed in the new snippets. This situation might occur due to the imposed budget constraints, and a choice has to be made about whether the new snippets should be introduced in summary. The threshold γ_t aids in making this decision.

Experimental Setup

The Data Set There is a scarcity of publicly available multi-stream video summarization data sets with synchronous video streams. Therefore, a video capture framework consisting of 4 fixed cameras was set up during a conference poster session. These cameras captured

video streams synchronously. Hence, the summarization procedure can assume that video snippets from the different cameras arrive concurrently.

Instantiation of Objective Function Similar to the single-stream framework, $g_{\text{fea}}^{\text{maxMS}}(\cdot)$ is instantiated on a set U of deep features that are generated using a Deep Neural Network (DNN).

The features are once again produced through a two-step process. First, an autoencoder is trained using the ILSVRC 2012 data set [147]. This trained model is then used to produce first level features for individual video frames (each frame is an image). Next, these first level features (output of the encoder) from a snippet's constituent frames are combined to train another autoencoder model to produce snippet level features. In the case of AC features, both these models are trained with the AC constraint.

Both single-stream and multi-stream frameworks use the same autoencoder architecture for the first level image features (Table 5.1). Table 5.3 is the architecture used for snippet level features in the multi-stream procedure. The snippet features are used to instantiate $g_{\text{fea}}^{\text{maxMS}}(\cdot)$ (equation 5.9) with $w_u = 1$ for all $u \in U$.

Results

The performance of the proposed multi-stream summarization procedure is compared with batch sieve-streaming++ [66] (BSSP). BSSP uses a submodular function as its objective. It also has a mathematical approximation guarantee of $\frac{1}{2} - \frac{3\epsilon}{2}$ for streaming, cardinality constrained, submodular maximization. Therefore, the same submodular function $g^{\text{maxMS}}(\cdot)$ (equation 5.9) can be used in both summarization procedures. The performance of these procedures is gauged based on the evaluation ($g^{\text{maxMS}}(S_t)$) of this function on the summary set S_t at each time step t.

Table 5.3: Neural network structure of the autoencoder used for snippet feature production in multi-stream summarization.

Group	Block Type (kernel sz, stride, channels)	# Blocks
conv1	$[\ 3 \times 3\ ]$, 1, 20	1
conv1 (residual)	$\begin{bmatrix} 3 \times 3 \\ 3 \times 3 \end{bmatrix}$, 1, 20	2
conv2	$[\ 3 \times 3\ ]$, 2, 5	1
conv2 (residual)	$\begin{bmatrix} 3 \times 3 \\ 3 \times 3 \end{bmatrix}$, 1, 5	2
conv3	$[\ 3 \times 3\ ]$, 3, 2	1
conv3 (residual)	$\begin{bmatrix} 3 \times 3 \\ 3 \times 3 \end{bmatrix}$, 1, 2	1
deconv3 (residual)	$\begin{bmatrix} 3 \times 3 \\ 3 \times 3 \end{bmatrix}$, 1, 2	1
deconv2	$[\ 3 \times 3\ ]$, 1, 5	1
deconv2 (residual)	$\begin{bmatrix} 3 \times 3 \\ 3 \times 3 \end{bmatrix}$, 1, 5	2
deconv1	$[\ 3 \times 3\ ]$, 2, 20	1
deconv1 (residual)	$\begin{bmatrix} 3 \times 3 \\ 3 \times 3 \end{bmatrix}$, 1, 20	2
deconv0	$[\ 3 \times 3\ ]$,1, 50	1

BSSP maintains multiple candidate summaries at each time step throughout the summarization process. Due to this characteristic, it has a memory requirement of $O(B + \frac{K}{\epsilon})$, where K is the summary budget, and B is an additional buffer used in the procedure. Therefore, it has a larger memory requirement than the proposed procedure. Figure 5.5 shows the comparison of the proposed multi-stream summarization method with BSSP ($\epsilon = 0.09$). For both procedures $g^{\text{maxMS}}(\cdot)$ was instantiated using AC features. It can be seen that the proposed method can generate summaries that achieve a higher function evaluation. BSSP, on the other hand, saturates midway. One of the factors that influence this phenomenon is that BSSP does not have a mechanism to swap out snippets from a candidate summary

set. Therefore, once a candidate set reaches budget saturation, it no longer introduces new snippets into the summary irrespective of their novelty.

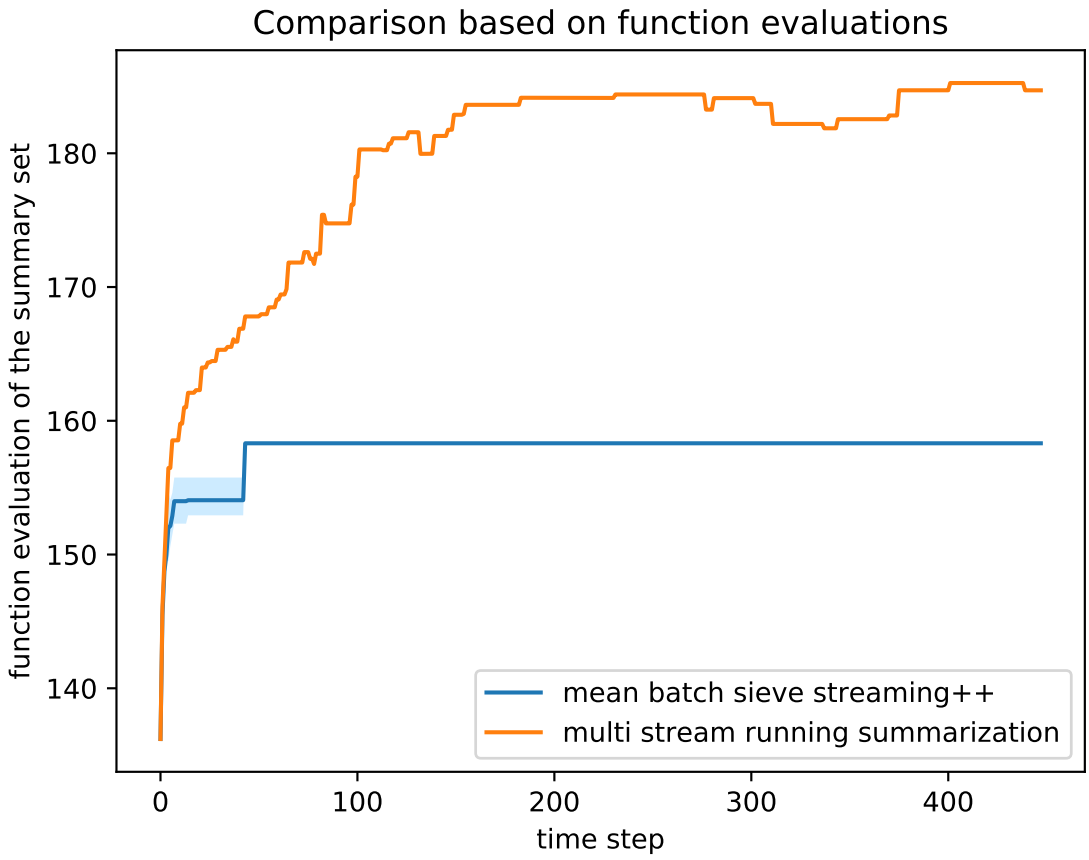

Figure 5.5: Comparison of the performance ($g^{\text{maxMS}}(S_t)$) of the multi-stream video summarization procedure with BSSP.

Chapter 6

UNSUPERVISED LEARNING OF THE WEIGHTS OF A SUBMODULAR MIXTURE

Constrained submodular maximization is the most common procedure for generating summaries (that abide by some constraint $\mathscr{C}$) from a ground set V using an appropriate submodular function $f(\cdot)$. The optimization is as follows:

$$S^* \in \underset{S \subseteq V, S \in \mathscr{C}}{\operatorname{argmax}} f(S). \tag{6.1}$$

Define a monotone, normalized, and non-decreasing submodular function ($F_w : 2^V \rightarrow \mathbb{R}_+$), created as a mixture of other component submodular function as:

$$F_w(A) = \sum_i w_i f_i(A), \tag{6.2}$$

where each $f_i : 2^V \rightarrow \mathbb{R}_+$ is itself a monotone, normalized, and non-decreasing submodular function.

The aim is to learn the weights w_i such that $||w|| = 1$ and $w_i \geq 0 \quad \forall i$. Furthermore, minimal hyperparameters and no ground truth information should be incorporated into the learning objective to learn these weights. The learned function $F_w(\cdot)$ is used for summarization through constrained submodular maximization. Some of the work described here was published as [81].

If a meta-objective $J(w)$ measures the quality of the mixture weights, the problem of learning the mixture weights can be defined as:

$$\max_{w \geq 0, ||w||=1} J(w) \tag{6.3}$$

Due to the lack of supervised information, the objective $J(w)$ is designed to measure a set of properties of $F_w(\cdot)$. The assumption is that these properties are important and necessary for $F_w(\cdot)$ to represent a good summarization objective. These properties are emphasized by setting $J(w)$ itself to be a mixture of meta-objectives, each one measuring one of the properties. The meta-objectives are, in turn, weighted by meta-hyperparameters. These characteristics can be described as follows.

6.1 J_1*: Confidence*

A good summarization objective needs to be confident about the set $S \subseteq V$ that it chooses as the summary. Let $S_{\max}$ denote the set that results from constrained maximization (2.4) of $F_w(\cdot)$. Let $S_{\min}$ denote the set that results from constrained minimization (2.5) of $F_w(\cdot)$. A large difference between $F_w(S_{\max})$ and $F_w(S_{\min})$ would imply that $S_{\max}$ is a good summary with respect to $F_w(\cdot)$ with high confidence.

Hence, maximizing J_1 should ensure a noticeable score gap between the top summaries and the bottom summaries. J_1 is, therefore designed as:

$$J_1(w) = \mathbb{E}_{k \sim p}\left[\left| \max_{S \subseteq V, |S|=k} F_w(S) - \min_{S' \subseteq V, |S'| \geq k} F_w(S')\right|\right] \tag{6.4}$$

where p is a distribution over summary size values.

6.2 J_2: High Entropy of w

High confidence in the summary is welcome. Overconfidence, however, can be undesirable. For example, J_1 (equation 6.4) can have a large value even if only a subset of components in $F_w(\cdot)$ have significant weight. Such sparsity might create a preference towards a small set of components that, in turn, can influence the ability of $F_w(\cdot)$, when maximized, to encourage diversity. Maximizing J_2 encourages the entropy of w to remain high, thereby affecting the tendency to be overconfident. J_2 is, therefore designed as:

$$J_2(w) = -\sum_i w_i log(w_i) \tag{6.5}$$

Hence, J_2 performs the role of a regularizer in the overall objective J.

6.3 J_3: Non-modularity of Summary Scores

The extent of modularity in $F_w(\cdot)$ agrees with the degree to which the summarized elements are independent. A strongly modular summarization objective that works well would imply no redundancy in the original data. It might, therefore, not be useful to spend resources to summarize such data. Hence, assuming substantial redundancy in the data, J_3 gauges the non-modularity of $F_w(\cdot)$. J_3 is designed as:

$$J_3(w) = \mathbb{E}_{k\sim p}\left[\sum_{s\in\hat{S}} F_w(s) \quad - F_w(\hat{S}) \middle| \hat{S} \in \underset{S\subseteq V:|S|\leq k}{\operatorname{argmax}} F_w(S)\right] \tag{6.6}$$

where again p is distribution over summary sizes.

6.4 J_4: Curvature

Total curvature [25] is another mechanism that estimates the submodularity of a set function. A curvature of zero implies that the function is fully modular. On the other hand, a curvature of one indicates that it is fully submodular. J_4 incorporates this characteristic in J and is designed as:

$$J_4(w) = 1 - \min_{j \in V} \frac{F_w(j|V \setminus j)}{F_w(j)}. \tag{6.7}$$

Objectives J_4 and J_3 are related. A maximum value of J_4 ($J_4(w) = 1$), however, does not imply that J_3 is maximized. Therefore, both objectives are included in J.

6.5 J_5: Stability

It is also desirable that the learned model is stable. Ideally, small perturbations to a learned w should not result in drastically different summaries. The objective J_5 is designed to encourage this tendency. Let $\hat{w}$ be a random variable centered around w and governed by distribution $p_w(\hat{w})$. The stability of w can be defined as:

$$J_5(w) = -\mathbb{E}_{p_w}\left[\mathbb{E}_{k \sim p}\left[\max_{S \subseteq V, |S|=k} F_w(S) \quad - \max_{S' \subseteq V, |S'|=k} F_{\hat{w}}(S')\right]^2\right]. \tag{6.8}$$

Hence, maximizing J_5 discourages small perturbations to w from causing drastically different summaries. In practice, w-mean Gaussian noise followed by projection back onto the simplex [34] is used to generate $\hat{w}$.

6.6 J_6: Soft De-duplication before saturation

In an ideal scenario, it is desirable that there are no redundant elements in a partial summary S before the summary has reached its budget ($|S| < k$). Some redundancy is, however, palatable if the f has reached saturation ($f(S) \approx f(V)$). Let $\mathcal{u}(S) = 1 - F_w(S)/F_w(V) \in [0, 1]$ define the unsaturation degree. Therefore, no redundant item should be added to the summary if $\mathcal{u}(S)$ is large. Let $\psi(v, S) = \max_{s \in S} \text{sim}(v, s)$ be the maximum similarity between v and any element in S and $\text{sim}(v, s)$ is the similarity between v and s ($s \in S$). A large $\mathcal{u}(S)$ corresponds to a large $F_w(v|S)$ when $\psi(v, S)$ is small. Such behavior can be encouraged using J_6.

$$J_6(w) = \mathbb{E}_{k \sim p}\left[\sum_{i=1}^{k}\left(1 - \frac{F_w(\hat{S}_i)}{F_w(V)}\right)\frac{F_w(v_i|\hat{S}_{i-1})}{\psi(v_i, \hat{S}_{i-1})} \;\middle|\; \hat{S}_i \in \text{gargmax}_{S \subseteq V, |S|=i} F_w(S), v_i \in \hat{S}_i \setminus \hat{S}_{i-1}\right] \quad (6.9)$$

Subset selection through maximization of submodular functions is done through a greedy procedure. Therefore, J_6 utilizes 'gargmax', i.e., the ordering of the summary elements produced through a greedy procedure (as shown in Algorithm 4).

6.7 Combined Objective

The combined objective takes a weighted mixture of all the meta-objectives,

$$J(w) = \sum_{\ell} \lambda_\ell J_\ell(w). \quad (6.10)$$

The set $\lambda \triangleq \{\lambda_\ell\}_\ell$ act as six meta-hyperparameters. A fixed set λ defines an instantiation of the training objective J. The training procedure uses this instantiation to learn the mixture

weights w. Furthermore, as none of the meta-objectives J_i need supervised feedback, the training is unsupervised.

This approach is, in a way, a strategy for reducing the number of hyperparameters. A summarization objective $f : 2^V \rightarrow \mathbb{R}_+$ may have 2^n free parameters, where $n = |V|$. The use of feature based functions reduces this to the modular functions m_u and the mixture weights w. Chapter 4 discussed how the functions m_u are first learned in an unsupervised manner. The weights w are learned using J. Optimizing the mixture using $J(w)$ reduces the hyperparameters further to the meta-hyperparameters in λ. It is a significant reduction from 2^n.

For a fixed λ, the objective J is optimized using gradient ascent. At each weight update, the following steps are followed:

Algorithm 3: Procedure for updating the weights w while optimizing $J(w)$

1 **Input:** The weight vector w^t at update step t

2 **Output:** The updated weight vector w^{t+1}

3 Choose $k \sim p$

4 Perform discrete optimizations for those meta-objectives that need them (e.g., J_1, J_3, J_5, and J_6

5 Compute the gradient $\nabla_w J(w)$ at the discrete solutions

6 Adjust w^t with a gradient step under a given learning rate

7 Project the updated weights back to the probability simplex as w^{t+1} using [34].

The discrete optimizations in J_3, J_5, J_6, and the first part of J_1 can be near optimally solved, with an approximation guarantee of $1 - 1/e$, using the greedy algorithm [124]. These optimizations all involve submodular maximization subject to a given budget on the summary size. A basic version of the greedy procedure for submodular maximization can take the following form:

Algorithm 4: Basic Greedy

Input: $V = \{v_1, v_2, \dots v_{|V|}\}$. Submodular function $f(\cdot)$. The budget k.

Output: The candidate set S for maximization of $f(\cdot)$.

Initialize: $S = \emptyset$, $u = \emptyset$

while $S < k$ **do**

  $u \in \text{argmax}_{v \in V \setminus S} f(v|S)$

  $S \leftarrow S \cup u$

However, multiple versions of the greedy algorithm, such as accelerated greedy [110, 114], and distributed greedy [116] are explored in literature and are more useful in practice.

There exist several procedures (such as [40]) that can be used to solve the constrained submodular minimization in J_1. This work utilizes semidifferential methods of Iyer et al. [61] that are fast methods to approximately solve the minimization problem. For a function $F_w(\cdot)$, supergradients are defined to formulate a modular approximation of it in the form $m^{g_Y}(X) = F_w(Y) + g_Y(X) - g_Y(Y) \geq F_w(X)$. Here g_Y is the generic supergradient of $F_w(\cdot)$ at Y. This formulation is then used with constraints $\mathscr{C}$ (cardinality constraint in the case of J_1) in the following form:

Algorithm 5: MMin procedure by [61]

Initialize: Start with arbitrary X^0

while *Not Converged* **do**

  Choose a supergradient g_{X^t} at X^t

  $X^{t+1} \in \text{argmin}_{X \in \mathscr{C}} m^{g_{X^t}}(X)$

  t $\rightarrow$ t + 1

Chapter 7

APPLICATIONS OF THE MIXTURE LEARNING PROCEDURE

The proposed mixture learning procedure can potentially be applied to any task/domain where there is a need for learning submodular functions for summarization. This chapter explores the application of this procedure to three distinct summarization tasks: 1) image summarization, 2) textual document extractive summarization, and 3) training data summarization. The empirical experiments also compare the usage of AC-constrained features and unconstrained autoencoder based features for $\{m_u\}_u$. The experiments demonstrate that the mixture learning procedure can outperform several baseline unsupervised methods. Furthermore, AC features outperform their unconstrained counterpart. The experiments used the same autoencoder architecture for both AC and the unconstrained feature case, with the difference being the constraint on the weight matrix (for AC).

Due to the lack of previous work on learning mixtures of submodular functions with no direct utilization of a ground truth summary in the learning objective, the proposed method is compared with other methods such as k-means (for the cardinality constrained case), Query Independent MMR (Maximal Marginal Relevance) [15], and the similarity based "Facility Location" (FL) submodular function [111, 174]. FL and MM provide strong baselines as MMR has the property of diminishing returns and FL is a submodular function. Euclidean distance is used inside similarity calculations in FL and MMR. All of these methods are either unsupervised or require minimal supervision for instantiation of the submodular function. Given a ground set V, a similarity calculation function $\psi(\cdot, \cdot)$ and a budget k, the MMR

procedure used in the experiments is as follows:

Algorithm 6: Query Independent MMR

Input: $V = \{v_1, v_2, \ldots, v_n\}$,$\psi(\cdot,\cdot)$, k

Output: Summary S of size k

Initialize: $S \leftarrow v_j$, $j \sim \text{uniform}(1, n)$

for $i = 2, \ldots k$ **do**

  $v^* = \text{argmin}_{v \in V \setminus S}[\max_{s \in S} \psi(v, s)]$

  $S \leftarrow S \cup v^*$

7.1 Structure of the Submodular Mixture

As explained in Chapter 6 the mixture of submodular functions takes the form:

$$F_w(A) = \sum_i w_i f_i(A) \tag{7.1}$$

This thesis explores the application of the mixture learning procedure where each $f_i(\cdot)$ is a feature based submodular function (equation 1.1). More specifically, the full representation of the mixture takes the form:

$$F_w(A) = \sum_{i \in I} \sum_{u \in U} \sum_{\gamma \in \Gamma_i} w_{i,u,\gamma} \frac{\phi_{i,u}\left(\gamma \frac{\sum_{a \in A} \mathfrak{e}(x_a)(u)}{\sum_{v \in V} \mathfrak{e}(x_v)(u)}\right)}{\phi_{i,u}(\gamma)} = \sum_{i \in I} \sum_{u \in U} \sum_{\gamma \in \Gamma_i} w_{i,u,\gamma} \phi_{i,u,\gamma}(m_u(A)). \tag{7.2}$$

The mixture F_w is composed of several 'mother' functions paired with scaling factors. These 'mother' functions are monotone, non-decreasing concave functions such as $\sqrt{\cdot}$, $\log(1+\cdot)$ and many more. In Equation 7.2, I is the index set of all the different concave functions in

use. Γ_i is the set of scaling factors for $i \in I$. These scalars can also be thought of as calibration parameters for the concave functions. U is the set of features used to define the representation of the data object. Therefore, $w_{i,u,\gamma}$ is the weight associated with the component that uses $\phi_{i,u,\gamma}$, the calibrated mother function associated with index i and feature u.

7.2 Mechanism for Reducing the Number of Mixture Components in F_w

On the one hand, having a rich set of calibration parameters (Γ_i) gives the mixture more opportunity to ensure that the information in the modular functions m_u is used appropriately. On the other hand, it increases the burden on the mixture learning procedure in Chapter 6 since there are more components to learn. Moreover, indiscriminately choosing these components and calibration coefficients can result in highly correlated components, and hence the components themselves become redundant. In order to be more efficient, it is beneficial to have a diverse set of components. This thesis proposes a component selection and summarization method such that the resultant components themselves are diverse. Submodularity is used for this task, but instead of real-world data such as images or documents, the objects being summarized are the instantiated, calibrated mother functions.

The process is as follows. Let, for example, the set of mother functions and their corresponding calibration coefficients be:

- Power functions of the form $x^{0.5}$, $x^{0.6}$, $x^{0.7}$, $x^{0.8}$ and $x^{0.9}$ with
  - $\gamma_{pow} \in \{1\}$.
- Log functions of the form $log(1 + \gamma_{lg}x)/log(1 + \gamma_{lg})$ with
  - $\gamma_{lg} \in \{0.1, 0.2, 0.5, 1, 2, 5, 10, 30, 50, 75, 100, 175, 200, 225, 275, 300\}$.
- Saturation functions of the form $min(\gamma_{sat}x, 1)$ with

– $\gamma_{sat} \in \{1.125, 1.5, 2, 5, 10, 30, 50, 75, 100, 175, 200, 225, 275, 300\}$.

Let this be a ground set $\hat{V}$. First, generate the evaluations of each $\hat{v} \in \hat{V}$ for random subsets of the actual data V (e.g., images in the case of image summarization). Next, use these evaluations to generate a correlation coefficient matrix Σ. Here, the entry at row $\tilde{i}$ and column $\tilde{j}$ is $\Sigma_{\tilde{i},\tilde{j}} = \frac{C_{\tilde{i}\tilde{j}}}{\sqrt{C_{\tilde{i}\tilde{i}} C_{\tilde{j}\tilde{j}}}}$, where C is the covariance matrix. Since Σ can contain negative values, use instead the absolute values, i.e., a matrix $\hat{\Sigma}$ whose $(\tilde{i}, \tilde{j})$ element $\hat{\Sigma}_{\tilde{i},\tilde{j}} = ||\Sigma_{\tilde{i},\tilde{j}}||$. Finally, use the matrix $\hat{\Sigma}_{\tilde{i},\tilde{j}}$ to instantiate a facility location function of the form $g(\hat{A}) = \sum_{\tilde{i} \in \hat{V}} \max_{\tilde{j} \in \hat{A}} \hat{\Sigma}_{\tilde{i},\tilde{j}}$ for $\hat{A} \subseteq \hat{V}$. The top L diverse elements in $\hat{V}$ can now be selected by using a greedy procedure for maximizing $g(\cdot)$.

7.3 Image Summarization

$F_w(\cdot)$ is instantiated through AC features generated from images. The features are generated by training image to image convolutional autoencoders. The design of the network architecture ensured that there was no bypass of the bottleneck. Hence, it did not use any pooling layers. Such layers typically need corresponding unpooling layers that can transfer information between the encoder and decoder parts of the network without passing through the bottleneck. In addition to this, the autoencoder used residual layers [53]. Pairs of similar layers (either convolutional or deconvolution) combined to form these residual layers. Figure 4.1 shows a sample architecture to extract features from images. The residual block at the bottleneck is portrayed as part of the zoomed view. This view also depicts the first deconvolution layer in the decoder part of the architecture. Table 7.1 provides a more detailed structure of the sample architecture.

The bottleneck layer directly connects to only "pos deconv" layer. This is a deconvolutional (transposed convolution)/residual block with non-negative weights. After each backpropagation update, these weights were projected to the non-negative orthant by setting all

negative values to zero. The networks were trained using ADAM [68] with batch normalization and rectified linear units. ILSVRC 2012 data set, part of the ImageNet Large Scale Visual Recognition challenge for object classification [147], was the autoencoder's training set. The data has around 1.28 million images in the training set and 50 thousand images in the validation set. Neural networks trained on this data set have been successful as feature extractors for various tasks [31, 82].

The experiments use a standard dataset for image summarization consisting of 14 sets [167]. Each set contains 100 images. These sets have a list of associated user summaries. The user summaries were collected using Amazon Mechanical Turk and subsequently cleaned as described in Tschiatschek et al. [167]. Each summary contains 10 images. The images in the data set are real-world personal photographs taken during holiday trips. The task of image summarization is formulated as cardinality constrained submodular maximization [167]. The set V corresponds to one of the sets of 100 images that are to be summarized. Given a budget $k = 10$, and the trained F_w, the aim is to find $S^* \in \text{argmax}_{S \subseteq V, |S| \leq k} f(S)$. This optimization is easily approximable with the greedy algorithm [124]. Normalized VROUGE score [167] is used to evaluate the quality of the summary. None of the ground truth summaries from [167] are used during training of F_w. They were used only to evaluate the hypothesized summaries produced via the learned F_w.

The original set of mother functions for defining F_w were: 1) power functions of the form $x^{0.5}$,$x^{0.6}$,$x^{0.7}$,$x^{0.8}$ and $x^{0.9}$; 2) log function of the form $log(1 + \gamma_{lg}x)/log(1 + \gamma_{lg})$; 3) ratio function of the form $(1 - 1/\gamma_{ra}x)/(1 - 1/\gamma_{ra})$; 4) and saturation function of the form $min(\gamma_{sat}x, 1)$. The original set of calibration coefficients are: 1) $\gamma_{pow} \in \{1\}$ for power functions; 2) $\gamma_{lg} \in \{0.1, 0.2, 0.5, 1, 2, 5, 10, 30, 50, 75, 100, 175, 200, 225, 275, 300\}$ for log functions; 3) $\gamma_{ra} \in \{1.125, 1.5, 2, 5, 10, 30, 50, 75, 100, 175, 200, 225, 275, 300\}$ for ratio functions; 4) $\gamma_{sat} \in \{1.125, 1.5, 2, 5, 10, 30, 50, 75, 100, 175, 200, 225, 275, 300\}$ for saturation functions.

The component functions discovered for AC features were: 1) a single power function of the form $x^{0.5}$ with $\gamma_{pow} \in \{1\}$ (γ normalization is ineffectual in this case); 2) log functions of the form $log(1 + \gamma_{lg}x)/log(1 + \gamma_{lg})$ with $\gamma_{lg} \in \{0.2, 300\}$; and 3) saturation functions of the form $min(\gamma_{sat}x, 1)$ with $\gamma_{sat} \in \{5, 10, 30, 50, 75, 100, 225\}$.

The component functions for the non AC case were: 1) a single power function of the form $x^{0.5}$ with $\gamma_{pow} \in \{1\}$; 2) log functions of the form $log(1 + \gamma_{lg}x)/log(1 + \gamma_{lg})$ with $\gamma_{lg} \in \{0.2, 300\}$; and 3) saturation functions of the form $min(\gamma_{sat}x, 1)$ with $\gamma_{sat} \in \{5, 10, 30, 50, 75, 100, 200\}$.

The proposed mixture learning procedure (Chapter 6) is used to learn F_w. The training uses, $p = \text{uniform}[1, 50]$. Fig 7.1 shows the normalized VROUGE (higher is better) results. The results are shown for various $|U| = d \in \{25, 98, 242\}$ to demonstrate the influence of feature dimensionality on performance. These 'd' values correspond to 'tiny', 'small' and 'medium' $|U|$ sizes. Fig 7.1 also compares the performance with several submodular and non submodular procedures and shows that the proposed approach can outperform the baselines. The benefit of AC over non-AC can also be observed.

7.4 Document Summarization

The mixture learning procedure's performance is gauged for document summarization through the 2003 & 2004 NIST Document Understanding Conference (DUC) [33] data sets. This data provides standard benchmarks for query independent multi-document extractive summarization. The model is trained on DUC 03 and evaluated on DUC 04. Summaries consist of text less than 665 bytes long. Porter stemmer is used to pre-process the data. The generated summaries are evaluated using the ROUGE-1 score [88] (No information about ROUGE-1 on DUC03 or DUC04 was used at any time during training). The optimization problem used here in document summarization is similar to the one used for image summarization

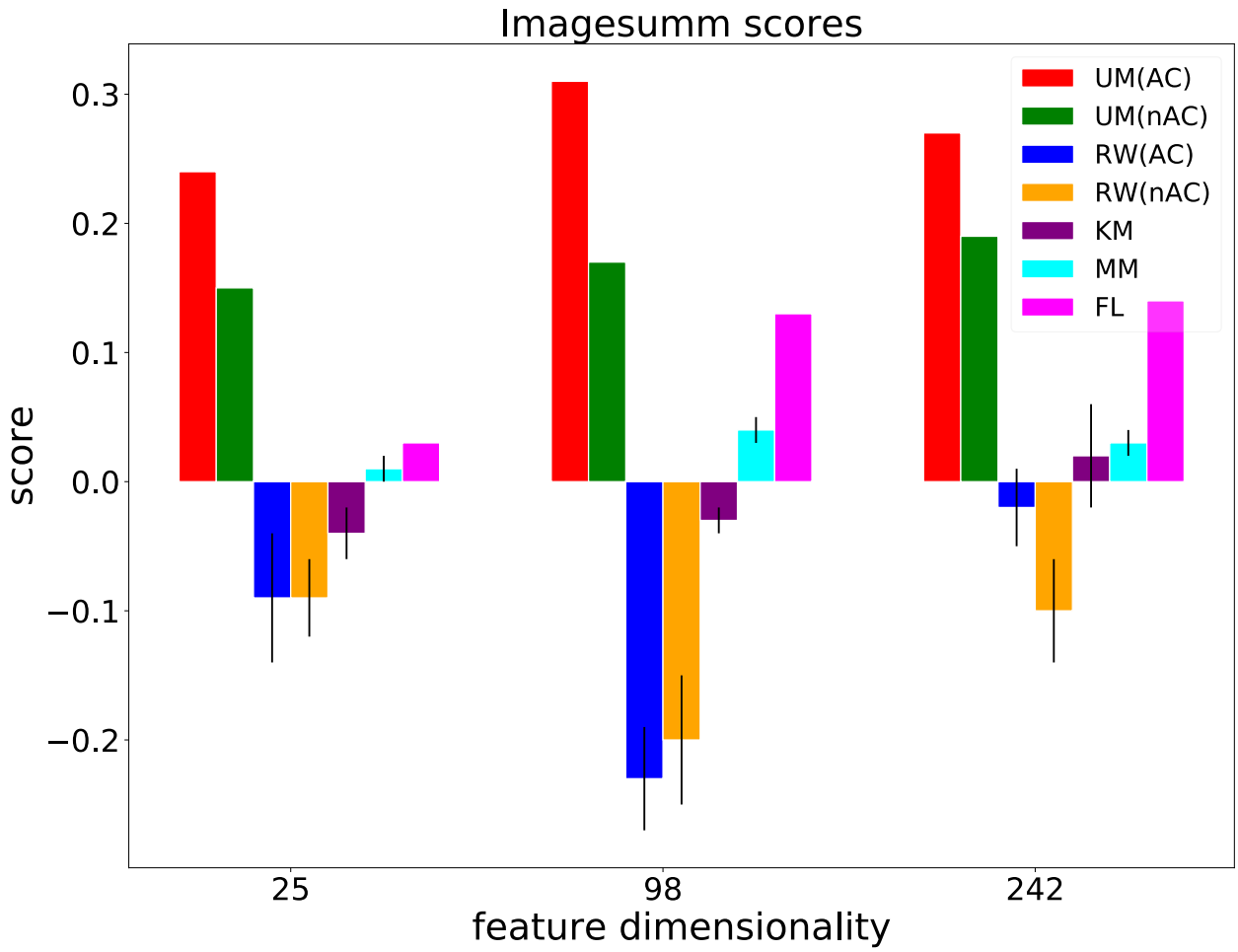

Figure 7.1: Comparison of Unsupervised Mixture (UM) for AC and non-AC (unconstrained) bottleneck features, random weights (RW), k-means (KM) (for cardinality constraint scenario), MMR-like baseline (MM) and a similarity based Facility Location (FL) submodular function for Image Summarization. The comparison is based on normalized vrouge score [167].

Table 7.1: Sample neural network structure of autoencoder for extracting features from images for the image summarization experiments.

Group	Block Type (kernel sz, stride, channels)	# Blocks
conv1	$[\ 11 \times 11\ ]$, 4, 96	1
conv2 (residual)	$\begin{bmatrix} 3 \times 3 \\ 3 \times 3 \end{bmatrix}$, 1, 96	2
conv3	$[\ 5 \times 5\ ]$, 2, 256	1
conv4 (residual)	$\begin{bmatrix} 5 \times 5 \\ 5 \times 5 \end{bmatrix}$, 1, 256	2
conv5	$[\ 3 \times 3\ ]$, 2, 384	1
conv6 (residual)	$\begin{bmatrix} 3 \times 3 \\ 3 \times 3 \end{bmatrix}$, 1, 384	1
conv7	$[\ 5 \times 5\ ]$, 1, 2	1
conv8 (residual)	$\begin{bmatrix} 3 \times 3 \\ 3 \times 3 \end{bmatrix}$, 1, 2	1
deconv9 (residual)	$\begin{bmatrix} 3 \times 3 \\ 3 \times 3 \end{bmatrix}$, 1, 2	1
deconv10	$[\ 5 \times 5\ ]$, 1, 384	1
deconv11 (residual)	$\begin{bmatrix} 3 \times 3 \\ 3 \times 3 \end{bmatrix}$, 1, 384	1
deconv12	$[\ 5 \times 5\ ]$, 2, 256	1
deconv13 (residual)	$\begin{bmatrix} 3 \times 3 \\ 3 \times 3 \end{bmatrix}$, 1, 256	2
deconv14	$[\ 4 \times 4\ ]$, 2, 96	1
deconv15 (residual)	$\begin{bmatrix} 3 \times 3 \\ 3 \times 3 \end{bmatrix}$, 1, 96	2
deconv16	$[\ 12 \times 12\ ]$, 4, 3	1

but with a knapsack constraint [160, 59]. The greedy summarization procedure utilized the summary cost to ascertain that the summary is within budget.

The experiments required the generation of both AC-constrained and unconstrained features, and this was done through a two-step process. First, data from DUC along with data from the NEWS2013 data set [125] is used to generate 500 dimensional sentence vectors. An

LSTM based sequence to sequence network was trained using OpenNMT [71] to generate these sentence vectors. The model used a two-layer encoder along with a decoder that contained 500-dimensional hidden states. Note that this is only one possible formulation for the generation of sentence vectors. Furthermore, in the scenario where faster sentence feature generation is a requirement, other possible options such as Gehring et.al [41] are available in the literature. Next, the sentence vectors of DUC data were used to train both an AC constrained and a non-AC (unconstrained) autoencoder. These trained networks were used to produce AC and non-AC features. These features were then used to instantiate $F_w(\cdot)$. Table 7.2 shows an architecture of the autoencoder that was trained over sentence vectors. In the experiments $\boldsymbol{p} = \text{uniform}[1, M]$ where M is the size of the document under consideration for gradient calculation.

The original set of mother functions were: 1) power functions of the form $x^{0.5}$,$x^{0.6}$,$x^{0.7}$,$x^{0.8}$ and $x^{0.9}$; 2) log function of the form $log(1 + \gamma_{lg}x)/log(1 + \gamma_{lg})$; 3) and saturation function of the form $min(\gamma_{sat}x, 1)$. The original set of calibration coefficients are: 1) $\gamma_{pow} \in \{1\}$ for power functions; 2) $\gamma_{lg} \in \{0.1, 0.2, 0.5, 1, 2, 5, 10, 30, 50, 75, 100, 175, 200, 225,$
$275, 300\}$ for log functions; 3) $\gamma_{sat} \in \{1.125, 1.5, 2, 5, 10, 30, 50, 75, 100, 175, 200, 225,$
$275, 300\}$ for saturation functions.

The summarized components for this task for AC case were: 1) log functions of the form $log(1 + \gamma_{lg}x)/log(1 + \gamma_{lg})$ with $\gamma_{lg} \in \{1.5, 5, 300\}$; and 2) saturation functions of the form $min(\gamma_{sat}x, 1)$ with $\gamma_{sat} \in \{2, 5, 10, 30, 50, 100, 225\}$.

The component functions for the non AC case were: 1) log functions of the form $log(1 + \gamma_{lg}x)/log(1+\gamma_{lg})$ with $\gamma_{lg} \in \{1.5, 30, 300\}$; and 2) saturation functions of the form $min(\gamma_{sat}x, 1)$ with $\gamma_{sat} \in \{2, 5, 10, 30, 50, 100, 225\}$.

The ROUGE-1 recall scores on DUC 04 are reported in Fig 7.2. The benefit of AC over non-AC features is clearly visible in Fig 7.2. Furthermore, the proposed mixture learning

approach can also perform better than the baseline procedures.

Table 7.2: Sample neural network structure (fully connected layers) of autoencoder for extracting features from sentence vectors for the document summarization experiments.

Group	number of nodes
fc1	300
fc2	200
fc3	100
fc4	75
fc5	50
fc6	75
fc7	100
fc8	200
fc9	300
fc10	500

7.5 Training Data Summarization

The training of modern models used in machine learning can be very expensive [128]. In this section, the proposed approach is used to produce a summarization objective that, in turn, is employed to extract a good subset of training data. This selected data is used to train a classification model. The proposed procedure's performance is evaluated based on the trained classification model's accuracy on a test set.

The MNIST dataset [83] was used in the experiments. The standard training set of MNIST was first divided into a set containing 50,000 images (V_{train}) and a validation set containing 10,000 images (V_{val}). Both AC and non-AC features were produced for these sets. Table 7.3 shows an architecture of the neural net model that was used in training. In the experiments, $p \sim \text{uniform}[1, 200]$.

The original set of mother functions were: 1) power functions of the form $x^{0.5}$,$x^{0.6}$,$x^{0.7}$,$x^{0.8}$

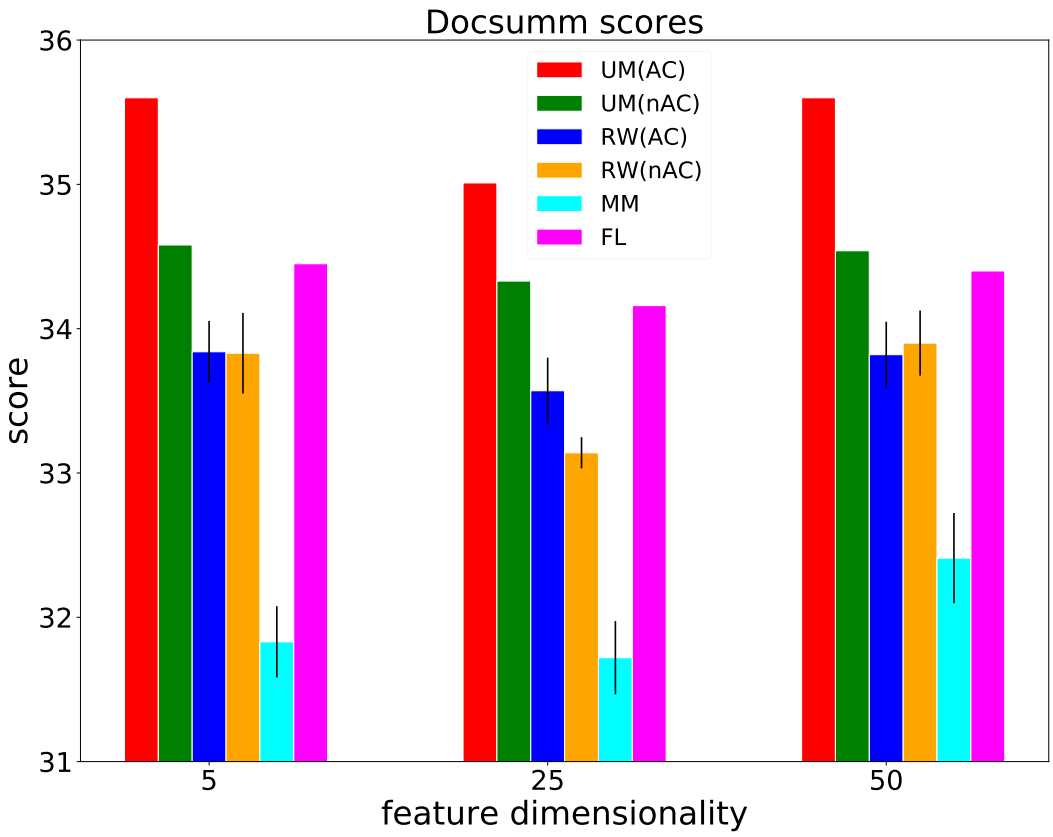

Figure 7.2: Comparison of Unsupervised Mixture (UM) for AC and non-AC (unconstrained) bottleneck features, random weights (RW), MMR-like baseline (MM) and a similarity based Facility Location (FL) submodular function for document summarization. The comparison is based on ROUGE 1 Recall score.

and $x^{0.9}$; 2) log function of the form $log(1 + \gamma_{lg}x)/log(1 + \gamma_{lg})$; 3) and saturation function of the form $min(\gamma_{sat}x, 1)$. The original set of calibration coefficients are: 1) $\gamma_{pow} \in \{1\}$ for power functions; 2) $\gamma_{lg} \in \{0.1, 0.2, 0.5, 1, 2, 5, 10, 30, 50, 75, 100, 175, 200, 225, 275, 300\}$ for log functions; 3) $\gamma_{sat} \in \{1.125, 1.5, 2, 5, 10, 30, 50, 75, 100, 175, 200, 225, 275, 300\}$ for saturation functions.

The summarized components for this task in the AC case were: 1) power function of the form $x^{0.8}$; 2) log functions of the form $log(1 + \gamma_{lg}x)/log(1 + \gamma_{lg})$ with $\gamma_{lg} \in \{300\}$; and 3) saturation functions of the form $min(\gamma_{sat}x, 1)$ with $\gamma_{sat} \in \{30, 75, 175, 275\}$.

The component functions in the non AC case were: 1) power function of the form $x^{0.8}$; 2) log functions of the form $log(1 + \gamma_{lg}x)/log(1 + \gamma_{lg})$ with $\gamma_{lg} \in \{300\}$; and 3) saturation functions of the form $min(\gamma_{sat}x, 1)$ with $\gamma_{sat} \in \{10, 50, 75, 175\}$.

The aim of the summarization procedure (using the learned submodular objective function) was to produce a subset of $k = 1000$ images from V_{train}. A ReLU-based neural network was used for classification. No pre-processing was performed on the data, and all models were trained with the same set of hyperparameters. Results are given in Fig 7.3. The benefit of AC over non-AC features is once again visible.

7.6 Ablation study for $J_i(\cdot)$

The performance of individual $J_i(w)$ for the different tasks (using the largest bottleneck size) is explored as an ablation study. Figure 7.4, displays the rank of each meta-objective. The importance of each meta-objective is dependent on the data and task under consideration. A rough pattern, however, does emerge. J_4 (curvature) is consistently ranked high across tasks. Similarly, J_5 (stability) and J_6 (soft de-duplication) are the next most consistent performers. This pattern lends credence to these meta-objectives' importance and the predominantly submodular nature of the summarization task itself (high curvature consistently performs well).

Table 7.3: Sample neural network structure (fully connected layers) of autoencoder for extracting features from images for the training data subset selection experiments.

Group	number of nodes
fc1	100
fc2	75
fc3	50
fc4	35
fc5	20
fc6	16
fc7	20
fc8	30
fc9	50
fc10	75
fc11	100
fc12	784

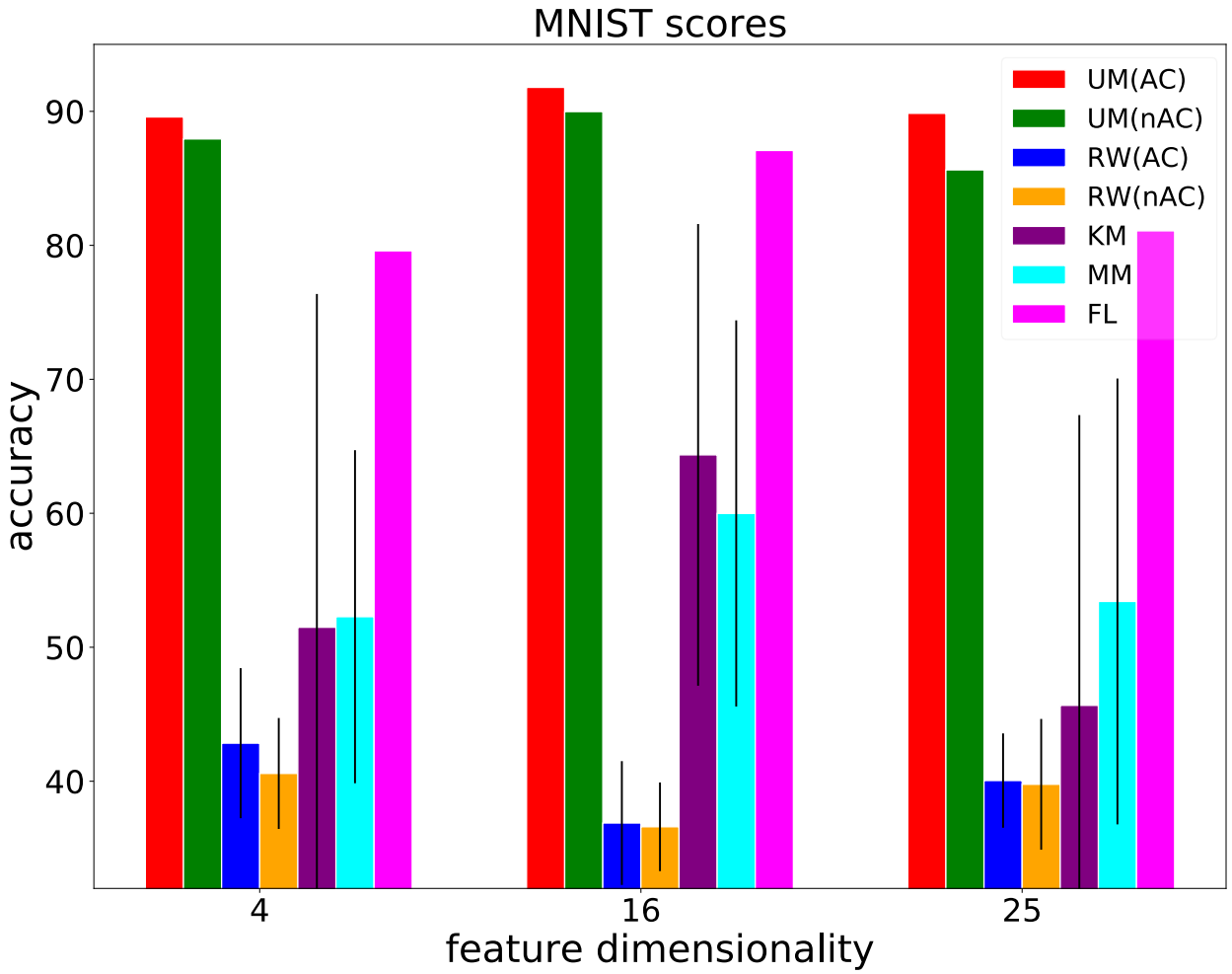

Figure 7.3: Comparison of Unsupervised Mixture (UM) for AC and non-AC (unconstrained) bottleneck features, random weights (RW), k-means (KM) (for cardinality constraint scenario), MMR-like baseline (MM) and a similarity based Facility Location (FL) submodular function for training data subset selection. The comparison is based on classification performance (accuracy in %) for a model trained on the selected subset of training data.

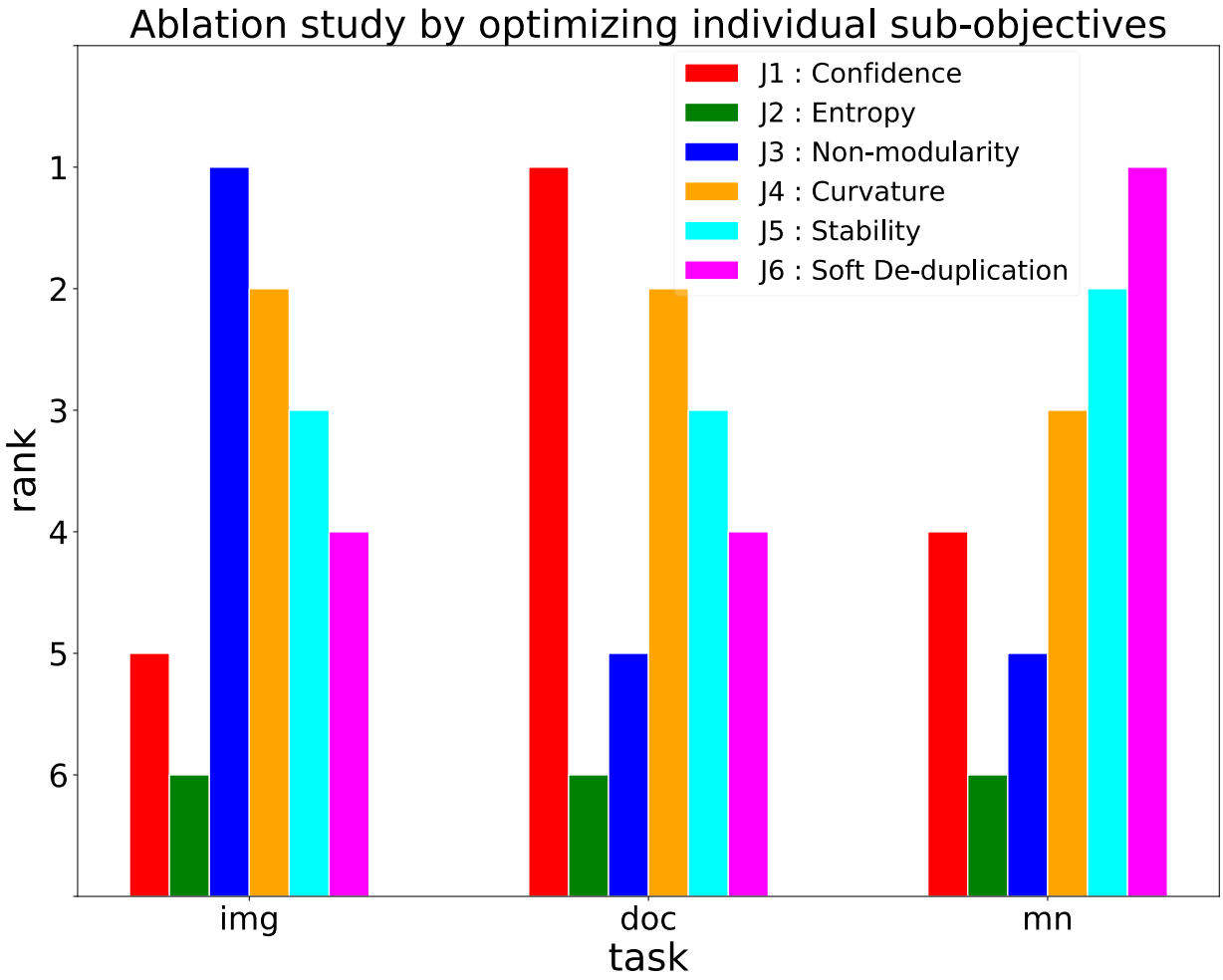

Figure 7.4: Ablation study depicting rank of the meta-objectives for different tasks(img: image summarization, doc: document summarization, and mn: subset selection). Rank 1 is best.

Chapter 8

PROSPECTIVE EXPLORATORY AVENUES

There are many avenues that have potential for future explorations. Some of these are discussed in the following sections.

8.1 Asymmetric Autoencoder with Single Layer Decoder

Autoencoders have seen extensive use in dimensionality reduction procedures. At their core is an encoder-decoder architecture. The encoder is often responsible for the transformation of data into a low dimensional space. Therefore, it also acts as a compressor, while the decoder recovers an approximation of the original and thus acts as a decompressor.

It is often observed that in a compression/decompression procedure, the compression is much more computationally difficult than decompression (this is true, for example, in popular compression tools such as zip, gzip, bzip2, and xz). Figure 8.1 and 8.2 show the compression and decompression timings for different tools on ASCII text data and binary data. Each tool used the '-9' option during compression with 'xz' running in non-multi-threaded mode. The massive difference between the compression and decompression time is observable for each tool.

This observation gives motivation for exploring highly asymmetric autoencoders. The aim is to learn autoencoders with a heavy and deep encoder and a single-layer decoder in an end-to-end manner. Figure 8.3 displays the generic structure of such an autoencoder.

Several tasks, including non-negative dictionary learning, data storage, and the learning of features for summarization can benefit from their use. These autoencoders are also valuable

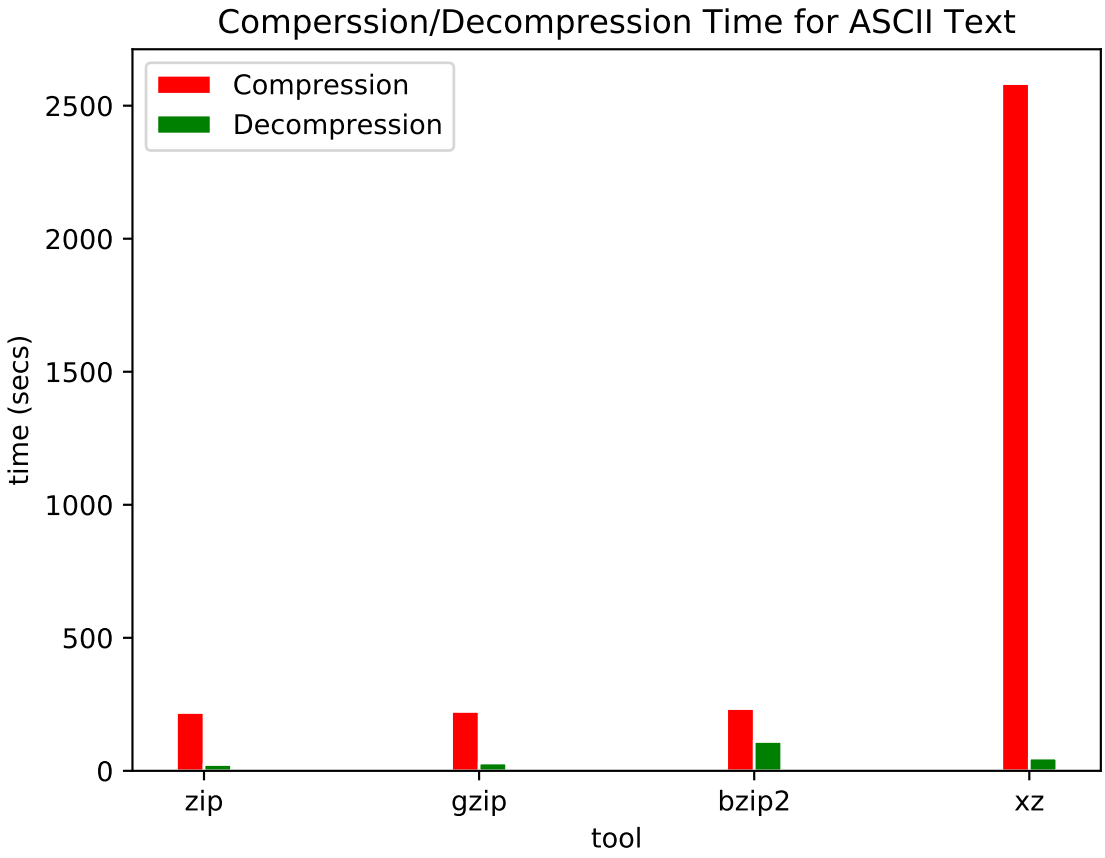

Figure 8.1: Comparison of compression and decompression time of different tools for a sample of ASCII Text data.

for efficient communication and processing of data flowing in a network. The single-layer decoder requires extremely low resources for transforming the encoded data to its original space, thereby reducing the computational burden on the network's nodes. The compressed data can also be easily transmitted over links with low bandwidth. Therefore, the use of such autoencoders can be extremely beneficial in resource constrained environment including sensor networks (such as Figure 8.4).

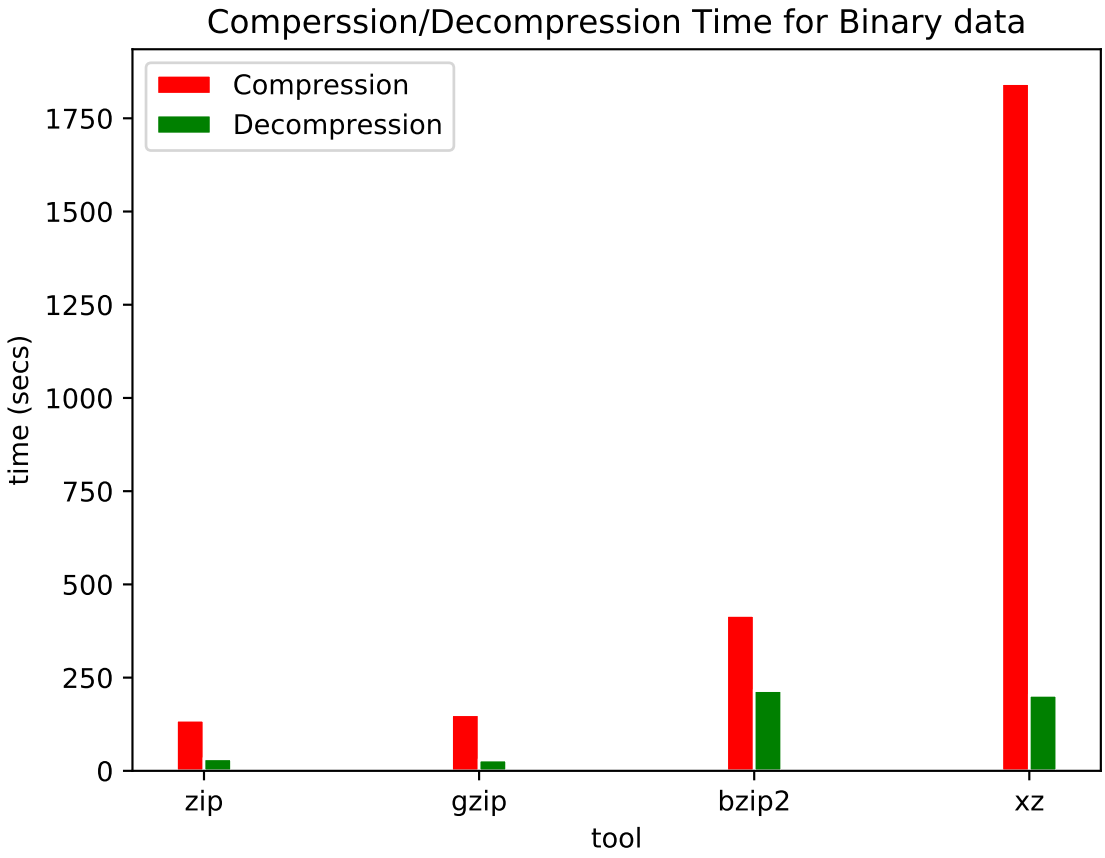

Figure 8.2: Comparison of compression and decompression time of different tools for a sample of binary data.

8.1.1 Related Work

Asymmetric autoencoder architectures have not been extensively explored in the literature. However, there are some works that do utilize them.

Tripathi and Majumdar [101] use asymmetric stacked autoencoders for classification and compression tasks. They hypothesized that by decreasing the number of decoder layers, the problem of overfitting could be mitigated. They also had the opinion that in a stacked decoder setup, each layer adds to the reconstruction loss. Therefore, reducing the number

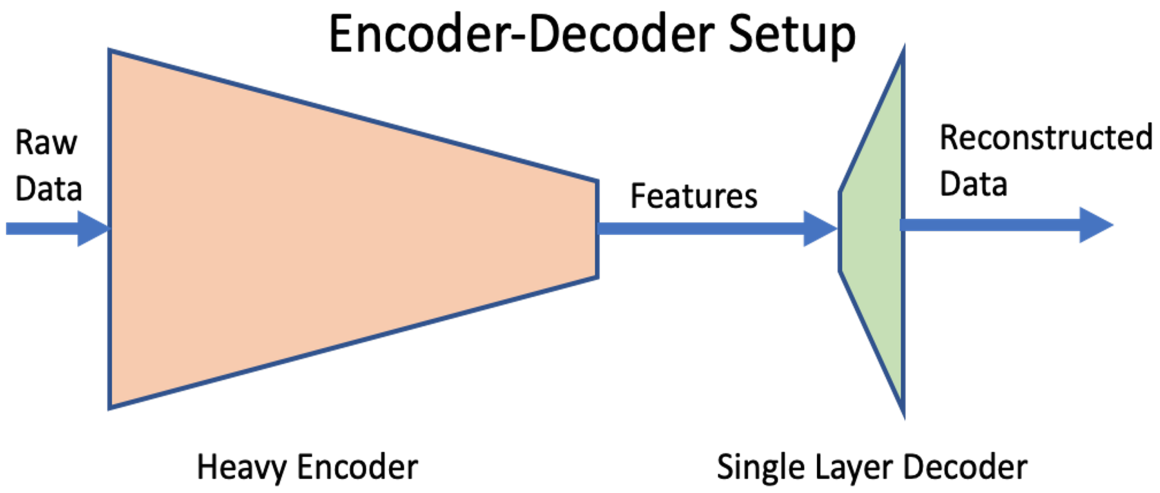

Figure 8.3: A generic structure for a highly asymmetric autoencoder with a single layer decoder.

of decoder layers has the potential to decrease the overall loss. They propose an Augmented Lagrangian Alternating Direction Method of Multipliers (ADMM) [11, 135] based approach to learn the weights of a three layer encoder and a one layer decoder. The features extracted from such a network are used for classification and compression tasks on multiple data sets.

Siddiqua and Fan [153] used an asymmetric autoencoder to retrieve 3D shapes based on depth images. They used a deeper encoder than the decoder to reduce the influence of artifacts on the features used for the retrieval task.

This work further lends credence to research effort towards gaining a better understanding of asymmetric autoencoder models.

8.1.2 One layer decoder architectures for summarization

The highly asymmetric autoencoders can be used for data summarization. In this section, their use is explored for image summarization. The aim is to produce AC features using

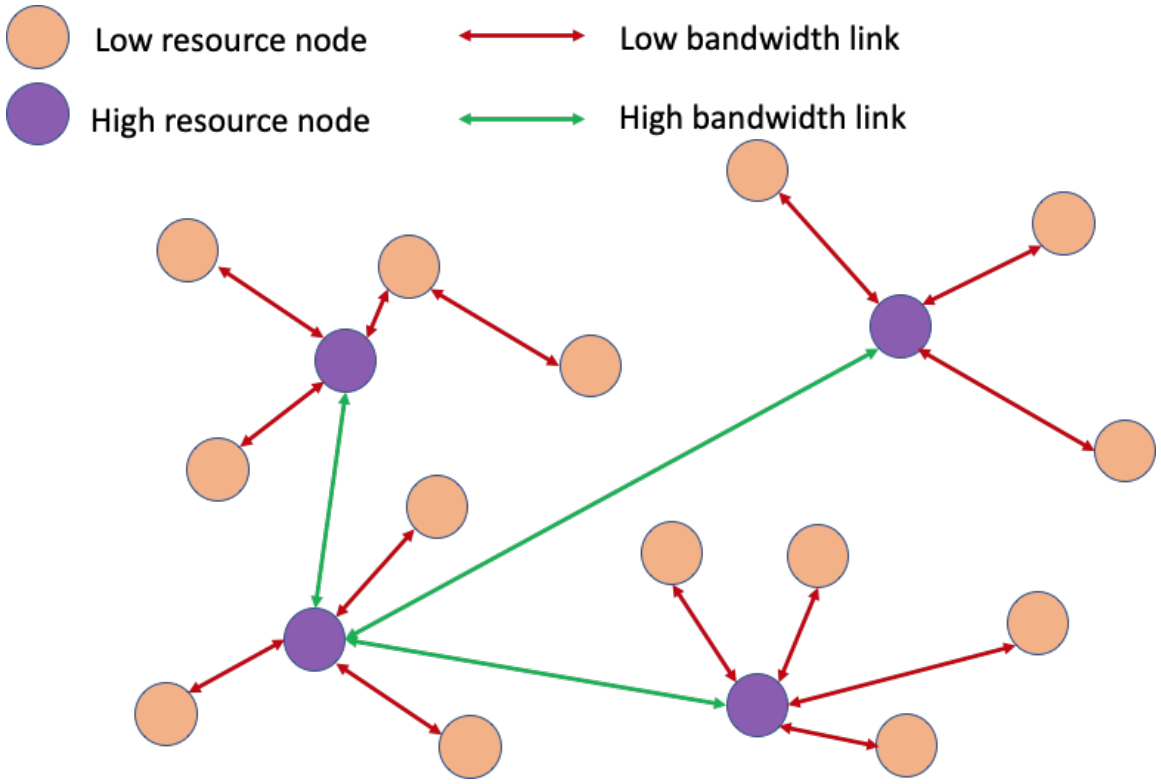

Figure 8.4: A network with low resource communication links and nodes that can benefit from the usage of highly asymmetric autoencoders.

autoencoders with one layer decoder. These features are used to instantiate a mixture of submodular functions (equation 7.2), and the mixtures weights are learned through a max-margin framework [167]. The autoencoders are trained on the ILSVRC 2012 data set [146], and the trained models are used to produce features for the image summarization data set [167].

Figure 8.5 demonstrates the performance of various asymmetric autoencoders with different encoder depths and a one layer decoder. It can be seen that, given enough encoder depth, it is possible to perform better than the average user. Therefore, the model can have high summarization performance and all the added benefits of a single layer decoder architecture.

All models in Figure 8.5 produce 1024 dimensional encoded features and use convolutional and deconvolutional (better known as transposed convolutional) layers in encoder and decoder, respectively. Table 8.1 shows the architecture of the model with a single layer encoder and decoder. Table 8.2 gives the architecture used for other asymmetric autoencoders with different encoder depths. The term 'nb' corresponds to the values depicted in Table 8.3.

The models in Figure 8.5 used convolutional layers in the encoder and a deconvolutional layer in the decoder. However, the use of such layers is not a requirement. It is also possible to use fully connected layers. Figure 8.6 uses models that utilize fully connected layers and produce 256 dimensional bottleneck features. The models show examples of two setups where: 1) the fully connected layer is only in the decoder (conve-fcd), and 2) the fully connected layers are in both encoder and decoder with no convolutional layer in use (fce-fcd). Tables 8.4 and 8.5 show the corresponding architectures. The summarization performance for the model with fully connected layers in both encoder and decoder seems to be better than the one where the fully connected layer is only in the decoder. This behavior is perhaps due to fully connected layers' ability to capture more global information than the convolutional layers. This characteristic can aid in summarization.

Open Questions There are still several open questions on the path to get a better understanding of asymmetric autoencoders with a single layer decoder. Some avenues of explorations are:

- How would a mix of convolutional and fully connected layers in the encoder influence the learned features' performance?

- What other type of encoder architectures can be used ? How do they influence performance?

- What type of storage performance can be extracted using the single layer decoder architectures. Is it possible to further encode the bottleneck features to improve storage efficiency?

These are only a few possible paths for exploration, and further research into them can provide many practical benefits.

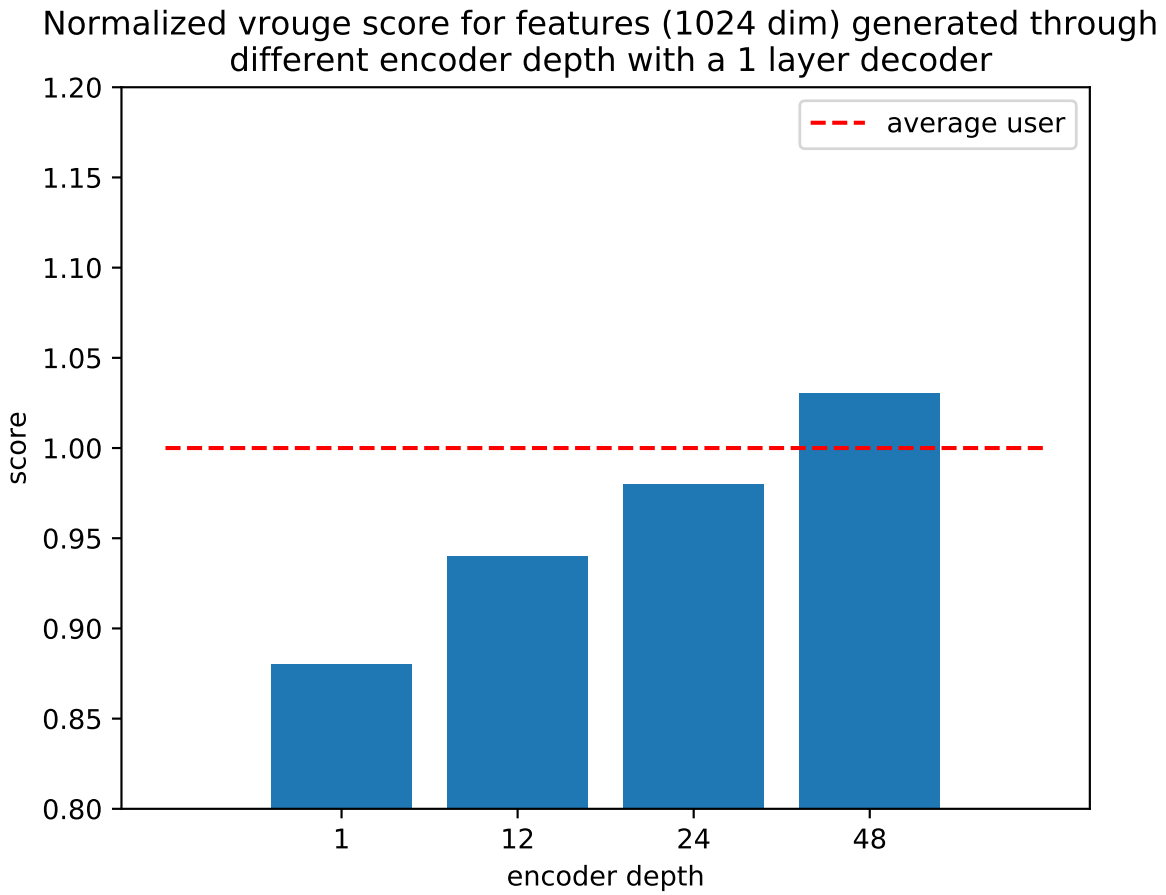

Figure 8.5: The summarization performance is shown for asymmetric autoencoders that use one layer decoders but have encoders with different depths. The encoder and decoder have convolutional and deconvolutional (also called transposed convolutional) layers.

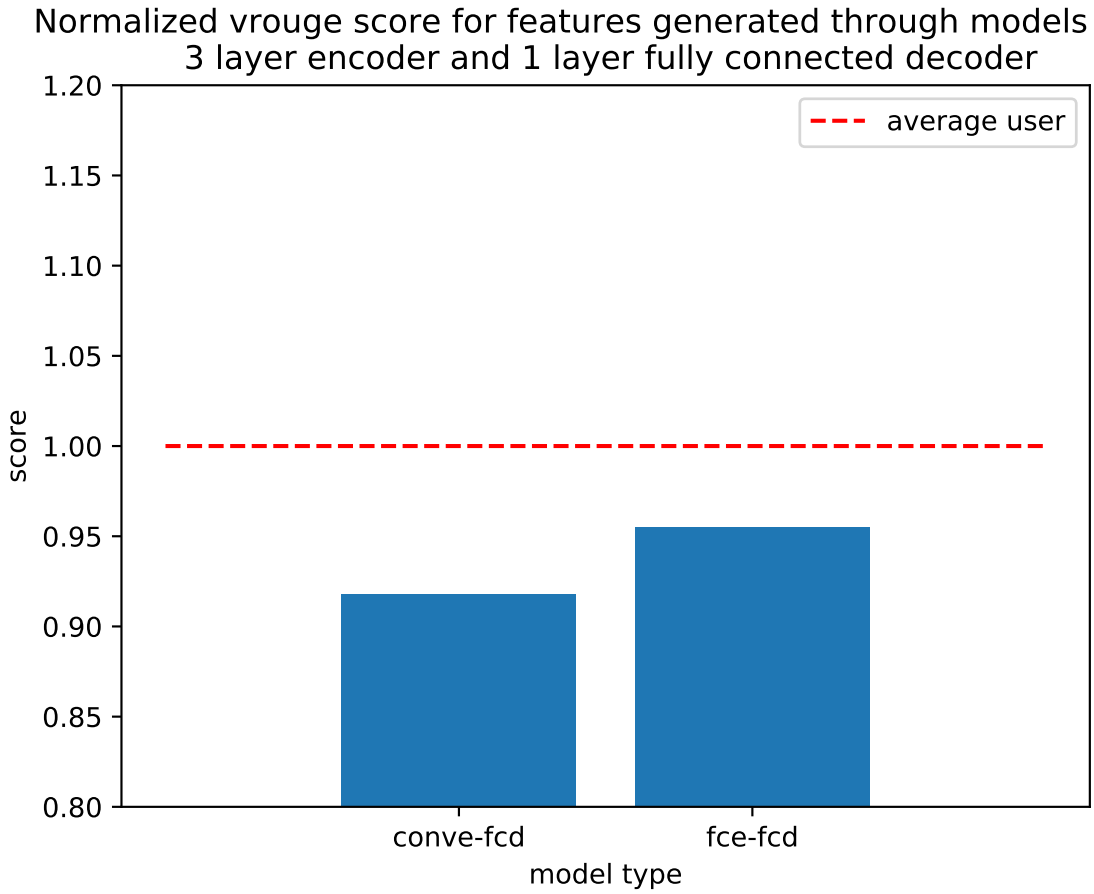

Figure 8.6: The summarization performance is shown for asymmetric autoencoders that use one layer decoders. The decoder comprises of a fully connected layer.

Table 8.1: Neural network structure for asymmetric autoencoder with a single layer encoder and decoder.

Group	Block Type (kernel sz, stride, channels)	# Blocks
conv1	[41 × 41], 17, 4	1
deconv0	[41 × 41], 17, 3	1

Table 8.2: Neural network structure for asymmetric autoencoder with a deep encoder and a single layer decoder.

Group	Block Type (kernel sz, stride, channels)	# Blocks
conv1	$[\ 3 \times 3\ ]$, 2, 64	1
conv1 (residual)	$\begin{bmatrix} 3 \times 3 \\ 3 \times 3 \end{bmatrix}$, 1, 64	nb
conv2	$[\ 3 \times 3\ ]$, 2, 16	1
conv2 (residual)	$\begin{bmatrix} 3 \times 3 \\ 3 \times 3 \end{bmatrix}$, 1, 16	nb
conv3	$[\ 3 \times 3\ ]$, 4, 8	1
conv3 (residual)	$\begin{bmatrix} 3 \times 3 \\ 3 \times 3 \end{bmatrix}$, 1, 8	nb
conv4	$[\ 3 \times 3\ ]$, 1, 4	1
conv4 (residual)	$\begin{bmatrix} 3 \times 3 \\ 3 \times 3 \end{bmatrix}$, 1, 4	1
deconv0	$[\ 41 \times 41\ ]$, 17, 3	1

Table 8.3: The number of blocks 'nb' for different encoder depths (in Figure 8.5).

depth	# Blocks (nb)
12	1
24	3
48	7

8.2 Towards end-to-end learning of the summarization framework

The framework proposed in this thesis is learned as a two-part process. The first part is the feature learning procedure, and the next part is the learning of mixture weights. It is, perhaps, possible to learn both parts together in an end-to-end unsupervised manner.

There have been some explorations in end-to-end learning of submodular functions.

Table 8.4: Neural network structure for asymmetric autoencoder with an encoder with convolutional layers and a single layer decoder with a fully connected layer.

Group	Block Type (kernel sz, stride, channels)	# Blocks
conv1	[3×3], 2, 64	1
conv2	[3×3], 4, 16	1
conv3	[3×3], 4, 4	1
Group	# in , # out	# Blocks
dfc0	256, 196608	1

Table 8.5: Neural network structure for asymmetric autoencoder with an encoder with fully connected layers and a single layer decoder with a fully connected layer.

Group	# in , # out	# Blocks
fc0	196608, 1024	1
fc1	1024, 512	1
fc2	512, 256	1
dfc0	256, 196608	1

Bilmes and Bai [9] and also Dolhansky and Bilmes [30] propose that deep submodular functions (a generalization and superclass of feature based functions) can be learned end-to-end. Kothwade et al. [73] demonstrate that in a supervised setup, both features and the mixture of submodular functions can be learned together in the training procedure.

These works give inspiration for defining an optimization objective that can be used to learn both the feature extractor's parameters and the mixture weights in the submodular function in an unsupervised setting. It will be beneficial, especially when the raw unsummarized data is enormous and has unique characteristics (such as multiple modalities and inherent predispositions). Coupling together the feature learning and mixture learning pro-

cedures may produce submodular functions that are especially suited to summarize the given type of data in such a setup.

Chapter 9

CONCLUSIONS

This work proposes a novel approach towards unsupervised learning of a submodular function (as a mixture of several submodular components) for summarization. The framework consists of a two-part process.

As the first part, a constrained, autoencoder based unsupervised feature learning approach is proposed. The aim is to produce features such that a larger value of a feature implies that the input data has a larger amount of the property defined by that feature. It is similar conceptually to bag-of-words features. Here, however, the 'words' are learned automatically through the constrained autoencoder setup.

As the next part of the process, a mixture of submodular components is instantiated through the learned features. The mixture weights are learned through an innovative mechanism that does not directly utilize any supervised summary information. The training involves optimizing meta-objectives, each of which likely corresponds to a summary's essential characteristic. The meta-objectives are combined using meta-hyperparameters. Empirical evaluations demonstrate that the submodular function learned through this two-part process can outperform baseline approaches.

In addition to this, other applications of the learned features are also explored. The focus application summarizes video streams (of potentially unbounded length) on the fly while abiding by a fixed memory budget. The aim is to produce a summary of the past video streams at each time step while abiding by the constraints. The task is called the 'generation of *running summaries*'. Algorithms for single-stream and multi-stream summarization for

generating running summaries are proposed. Empirical evaluations demonstrate that these procedures, with the submodular function instantiated using the proposed features, can outperform baseline approaches.

Furthermore, it is demonstrated that the proposed constrained training for feature generation can be utilized in several flavors of autoencoder models and losses. The influence of these different setups is also explored for the task of summarization.

Chapter 10

APPENDIX

10.1 Architectures and montages for different flavors of autoencoders

This section describes the architectures for the different flavors of autoencoders used in Section 4.3. Montages of images showing the input and reconstructed images for these different flavors are also depicted in this section.

Table 10.1: Neural network structure of the autoencoder with a single layer decoder.

Group	Block Type (kernel sz, stride, channels)	# Blocks
conv1	$[\ 3 \times 3\ ]$, 2, 64	1
conv1 (residual)	$\begin{bmatrix} 3 \times 3 \\ 3 \times 3 \end{bmatrix}$, 1, 64	3
conv2	$[\ 3 \times 3\ ]$, 2, 16	1
conv2 (residual)	$\begin{bmatrix} 3 \times 3 \\ 3 \times 3 \end{bmatrix}$, 1, 16	3
conv3	$[\ 3 \times 3\ ]$, 4, 8	1
conv3 (residual)	$\begin{bmatrix} 3 \times 3 \\ 3 \times 3 \end{bmatrix}$, 1, 8	3
conv4	$[\ 3 \times 3\ ]$, 1, 1	1
conv4 (residual)	$\begin{bmatrix} 3 \times 3 \\ 3 \times 3 \end{bmatrix}$, 1, 1	1
deconv0	$[\ 41 \times 41\ ]$, 15, 3	1

Table 10.2: Neural network structure of regular autoencoder.

Group	Block Type (kernel sz, stride, channels)	# Blocks
conv1	$[\ 3 \times 3\ ]$, 2, 64	1
conv1 (residual)	$\begin{bmatrix} 3 \times 3 \\ 3 \times 3 \end{bmatrix}$, 1, 64	3
conv2	$[\ 3 \times 3\ ]$, 2, 16	1
conv2 (residual)	$\begin{bmatrix} 3 \times 3 \\ 3 \times 3 \end{bmatrix}$, 1, 16	3
conv3	$[\ 3 \times 3\ ]$, 4, 8	1
conv3 (residual)	$\begin{bmatrix} 3 \times 3 \\ 3 \times 3 \end{bmatrix}$, 1, 8	3
conv4	$[\ 3 \times 3\ ]$, 1, 1	1
conv4 (residual)	$\begin{bmatrix} 3 \times 3 \\ 3 \times 3 \end{bmatrix}$, 1, 1	1
deconv4 (residual)	$\begin{bmatrix} 3 \times 3 \\ 3 \times 3 \end{bmatrix}$, 1, 1	1
deconv3	$[\ 3 \times 3\ ]$, 1, 8	1
deconv3 (residual)	$\begin{bmatrix} 3 \times 3 \\ 3 \times 3 \end{bmatrix}$, 1, 8	3
deconv2	$[\ 3 \times 3\ ]$, 4, 16	1
deconv2 (residual)	$\begin{bmatrix} 3 \times 3 \\ 3 \times 3 \end{bmatrix}$, 1, 16	3
deconv1	$[\ 3 \times 3\ ]$, 2, 64	1
deconv1 (residual)	$\begin{bmatrix} 3 \times 3 \\ 3 \times 3 \end{bmatrix}$, 1, 64	3
deconv0	$[\ 3 \times 3\ ]$, 2, 3	1

Table 10.3: Neural network structure of vae like autoencoder with Weibull distribution at the bottleneck. The parameters of the Weibull distribution are k and λ. The 'conv4' layer feeds into both 'conv$k$' and 'convλ' layers. These layers are then used to generate the bottleneck output.

Group	Block Type (kernel sz, stride, channels)	# Blocks
conv1	$[\ 3 \times 3\ ]$, 2, 64	1
conv1 (residual)	$\begin{bmatrix} 3 \times 3 \\ 3 \times 3 \end{bmatrix}$, 1, 64	3
conv2	$[\ 3 \times 3\ ]$, 2, 16	1
conv2 (residual)	$\begin{bmatrix} 3 \times 3 \\ 3 \times 3 \end{bmatrix}$, 1, 16	3
conv3	$[\ 3 \times 3\ ]$, 4, 8	1
conv3 (residual)	$\begin{bmatrix} 3 \times 3 \\ 3 \times 3 \end{bmatrix}$, 1, 8	3
conv4	$[\ 3 \times 3\ ]$, 1, 1	1
conv$k$	$[\ 3 \times 3\ ]$, 1, 1	1
convλ	$[\ 3 \times 3\ ]$, 1, 1	1
deconv4 (residual)	$\begin{bmatrix} 3 \times 3 \\ 3 \times 3 \end{bmatrix}$, 1, 1	1
deconv3	$[\ 3 \times 3\ ]$, 1, 8	1
deconv3 (residual)	$\begin{bmatrix} 3 \times 3 \\ 3 \times 3 \end{bmatrix}$, 1, 8	3
deconv2	$[\ 3 \times 3\ ]$, 4, 16	1
deconv2 (residual)	$\begin{bmatrix} 3 \times 3 \\ 3 \times 3 \end{bmatrix}$, 1, 16	3
deconv1	$[\ 3 \times 3\ ]$, 2, 64	1
deconv1 (residual)	$\begin{bmatrix} 3 \times 3 \\ 3 \times 3 \end{bmatrix}$, 1, 64	3
deconv0	$[\ 3 \times 3\ ]$, 2, 3	1

Table 10.4: Structure of the differentiator network that is used to introduce adversarial loss in the training.

Group	Block Type (kernel sz, stride, channels)	# Blocks
conv1	$[\ 3 \times 3\ ]$, 2, 64	1
conv1 (residual)	$\begin{bmatrix} 3 \times 3 \\ 3 \times 3 \end{bmatrix}$, 1, 64	1
conv2	$[\ 3 \times 3\ ]$, 2, 16	1
conv2 (residual)	$\begin{bmatrix} 3 \times 3 \\ 3 \times 3 \end{bmatrix}$, 1, 16	1
conv3	$[\ 3 \times 3\ ]$, 4, 8	1
conv3 (residual)	$\begin{bmatrix} 3 \times 3 \\ 3 \times 3 \end{bmatrix}$, 1, 8	1
conv4	$[\ 3 \times 3\ ]$, 1, 1	1
conv4 (residual)	$\begin{bmatrix} 3 \times 3 \\ 3 \times 3 \end{bmatrix}$, 1, 1	1
Group	# in , # out	# Blocks
fc1	256, 1	1

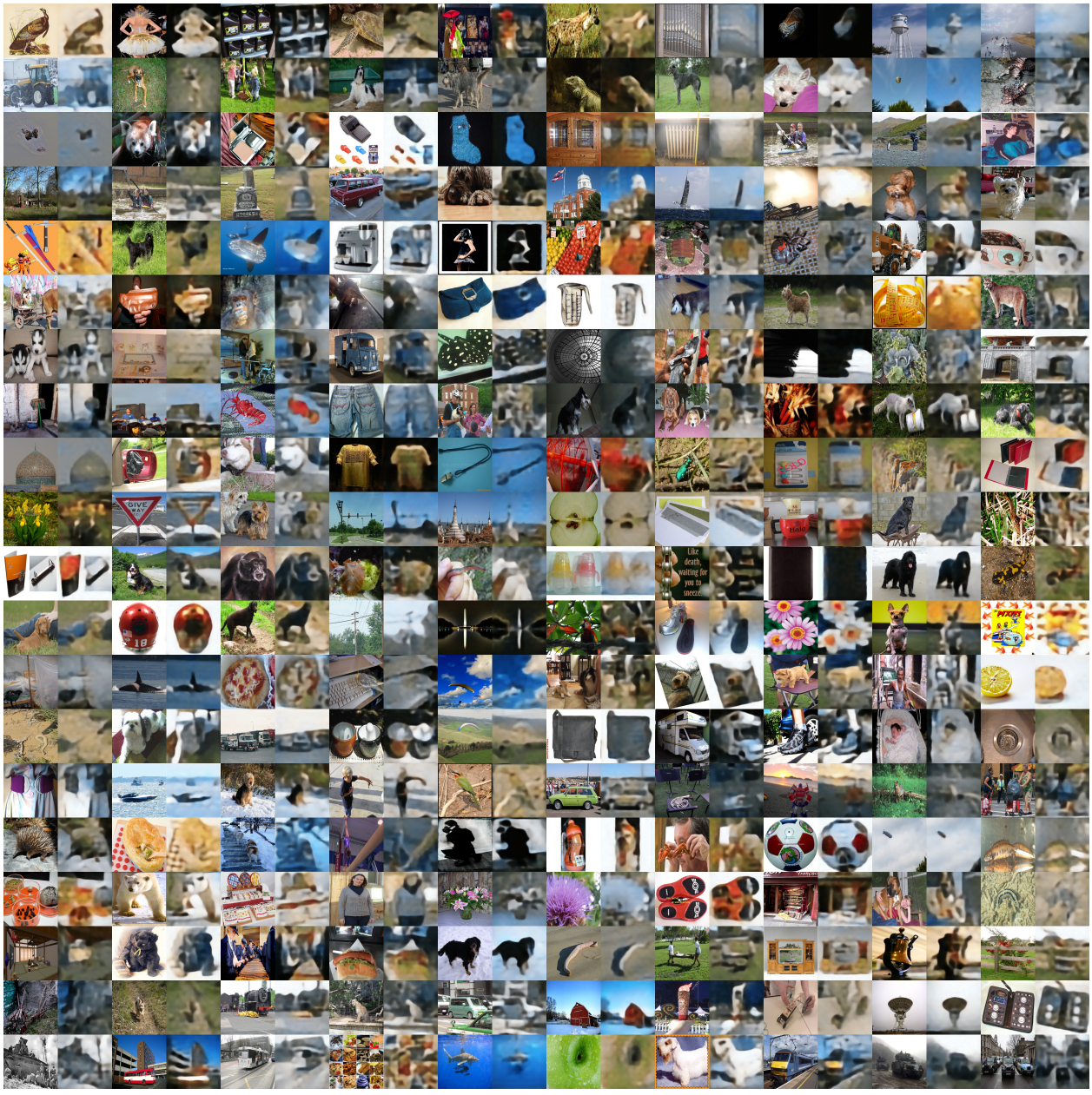

Figure 10.1: Montage of input and reconstructed images for the regular autoencoder with AC constraint.

Figure 10.2: Montage of input and reconstructed images for the va-w autoencoder with AC constraint.

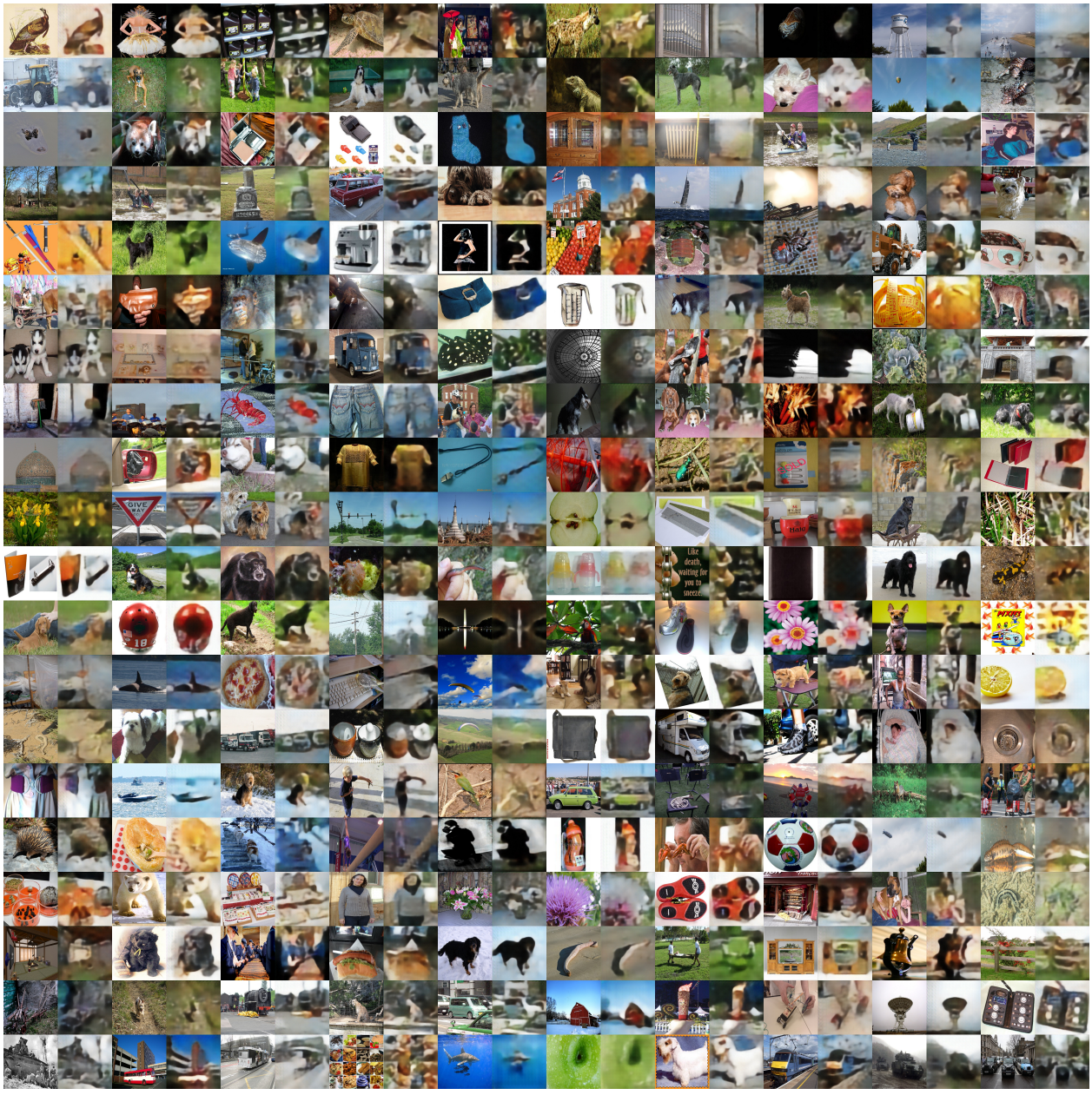

Figure 10.3: Montage of input and reconstructed images for the setup where adversarial loss was incorporated in the training of the autoencoder with AC constraint.

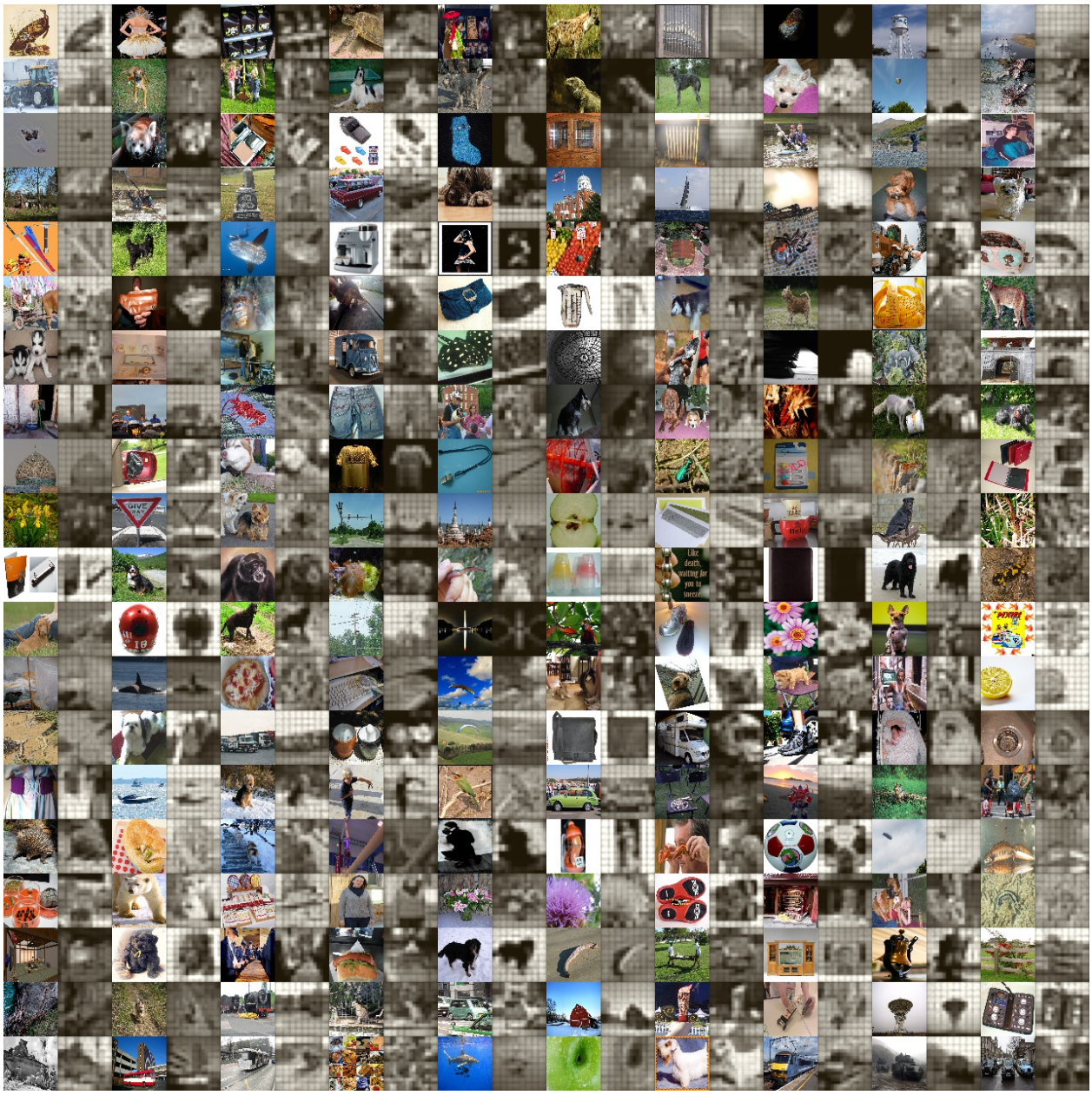

Figure 10.4: Montage of input and reconstructed images for the autoencoder with a single layer decoder and AC constraint.

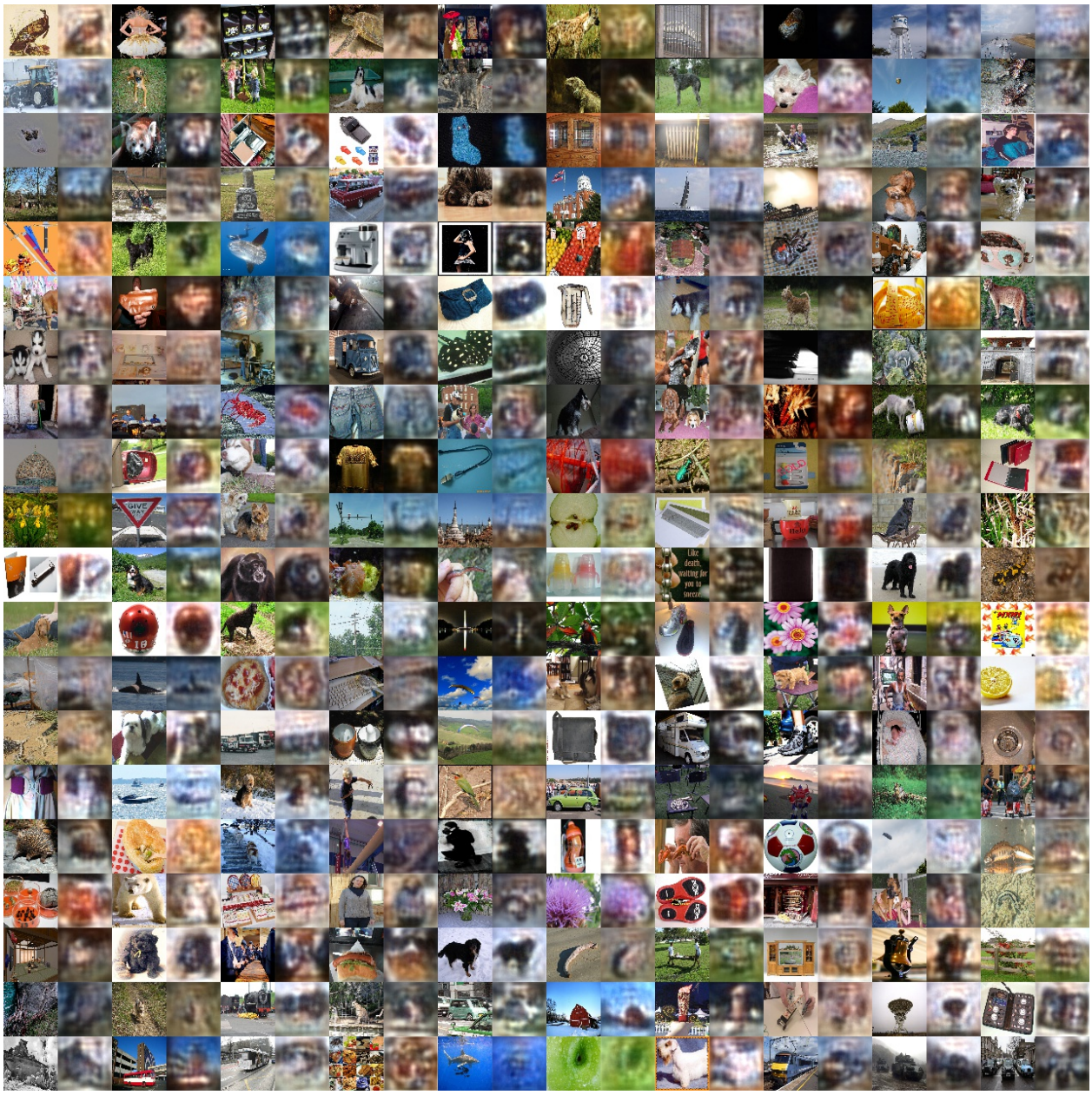

Figure 10.5: Montage of input and reconstructed images for the autoencoder with a single layer decoder, fully connected layers in both encoder and decoder (Table 8.5) and AC constraint.

BIBLIOGRAPHY

[1] Jurandy Almeida, Neucimar J Leite, and Ricardo da S Torres. Online video summarization on compressed domain. *Journal of Visual Communication and Image Representation*, 24(6), 2013.

[2] Michael R Anderson and Michael Cafarella. Input selection for fast feature engineering. In *ICDE*, pages 577–588. IEEE, 2016.

[3] Bishnu Saroop Atal. Automatic speaker recognition based on pitch contours. *The Journal of the Acoustical Society of America*, 52(6B):1687–1697, 1972.

[4] Hossein Azizpour, Ali Razavian, Josephine Sullivan, Atsuto Maki, and Stefan Carlsson. From generic to specific deep representations for visual recognition. In *CVPR Workshops*, 2015.

[5] Ashwinkumar Badanidiyuru, Baharan Mirzasoleiman, Amin Karbasi, and Andreas Krause. Streaming submodular maximization: Massive data summarization on the fly. In *SIGKDD*. ACM, 2014.

[6] Maria-Florina Balcan and Nicholas JA Harvey. Learning submodular functions. In *ACM symposium on Theory of computing*, pages 793–802. ACM, 2011.

[7] Madhushree Basavarajaiah and Priyanka Sharma. Survey of compressed domain video summarization techniques. *ACM Computing Surveys (CSUR)*, 52(6):1–29, 2019.

[8] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A

review and new perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(8):1798–1828, 2013.

[9] Jeffrey Bilmes and Wenruo Bai. Deep submodular functions. *preprint arXiv:1701.08939*, 2017.

[10] Hervé Bourlard and Yves Kamp. Auto-association by multilayer perceptrons and singular value decomposition. *Biological cybernetics*, 59(4-5):291–294, 1988.

[11] Stephen Boyd, Neal Parikh, and Eric Chu. *Distributed optimization and statistical learning via the alternating direction method of multipliers*. Now Publishers Inc, 2011.

[12] Niv Buchbinder, Moran Feldman, and Roy Schwartz. Online submodular maximization with preemption. In *SODA*. ACM-SIAM, 2015.

[13] Sijia Cai, Wangmeng Zuo, Larry S Davis, and Lei Zhang. Weakly-supervised video summarization using variational encoder-decoder and web prior. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 184–200, 2018.

[14] Gruia Calinescu, Chandra Chekuri, Martin Pal, and Jan Vondrák. Maximizing a monotone submodular function subject to a matroid constraint. *SIAM Journal on Computing*, 40(6):1740–1766, 2011.

[15] Jaime Carbonell and Jade Goldstein. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *SIGIR*. ACM, 1998.

[16] Shayok Chakraborty, Omesh Tickoo, and Ravi Iyer. Adaptive keyframe selection for video summarization. In *WACV*. IEEE, 2015.

[17] Ken Chatfield, Victor S Lempitsky, Andrea Vedaldi, and Andrew Zisserman. The devil is in the details: an evaluation of recent feature encoding methods. *BMVC*, 2(4), 2011.

[18] Kai Chen, Mathias Seuret, Marcus Liwicki, Jean Hennebert, and Rolf Ingold. Page segmentation of historical document images with convolutional autoencoders. In *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, pages 1011–1015. IEEE, 2015.

[19] Lin Chen, Andreas Krause, and Amin Karbasi. Interactive submodular bandit. In *Advances in Neural Information Processing Systems*, 2017.

[20] Xi Chen, Diederik P Kingma, Tim Salimans, Yan Duan, Prafulla Dhariwal, John Schulman, Ilya Sutskever, and Pieter Abbeel. Variational lossy autoencoder. 2017.

[21] Jianpeng Cheng and Mirella Lapata. Neural summarization by extracting sentences and words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 484–494. Association for Computational Linguistics, 2016.

[22] Sumit Chopra, Michael Auli, and Alexander M Rush. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 93–98, 2016.

[23] Ronald R Coifman and Stéphane Lafon. Diffusion maps. *Applied and computational harmonic analysis*, 21(1):5–30, 2006.

[24] François Coldefy and Patrick Bouthemy. Unsupervised soccer video abstraction based on pitch, dominant color and camera motion analysis. In *Proceedings of the 12th annual ACM international conference on Multimedia*. ACM, 2004.

[25] M. Conforti and G. Cornuejols. Submodular set functions, matroids and the greedy

algorithm: tight worst-case bounds and some generalizations of the Rado-Edmonds theorem. *Discrete Appl. Math.*, 7(3):251–274, 1984.

[26] William H Cunningham. Decomposition of submodular functions. *Combinatorica*, 3(1):53–68, 1983.

[27] William H Cunningham. Optimal attack and reinforcement of a network. *Journal of the ACM (JACM)*, 32(3):549–561, 1985.

[28] Hoa Trang Dang. Overview of duc 2005. In *Proceedings of the document understanding conference*, volume 2005, pages 1–12, 2005.

[29] Sandra Eliza Fontes de Avila, Ana Paula Brandão Lopes, Antonio da Luz, and Arnaldo de Albuquerque Araújo. Vsumm: A mechanism designed to produce static video summaries and a novel evaluation method. *Pattern Recognition Letters*, 32(1), 2011.

[30] Brian W Dolhansky and Jeff A Bilmes. Deep submodular functions: Definitions and learning. In *Advances in Neural Information Processing Systems*, pages 3404–3412, 2016.

[31] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *International conference on machine learning*, 2014.

[32] Apramey Dube, Anu Helkkula, et al. Customer approach to the use of big data: Wearables for service. In *SERVSIG*, 2016.

[33] The nist document understanding conference (duc) data, 2004.

[34] John Duchi, Shai Shalev-Shwartz, Yoram Singer, and Tushar Chandra. Efficient projections onto the l1-ball for learning in high dimensions. In *International conference on Machine learning*, pages 272–279. ACM, 2008.

[35] Mohamed Elfeki, Aidean Sharghi, Srikrishna Karanam, Ziyan Wu, and Ali Borji. Multi-stream dynamic video summarization. *arXiv preprint arXiv:1812.00108*, 2018.

[36] Ehsan Elhamifar and M Clara De Paolis Kaluza. Online summarization via submodular and convex optimization. In *CVPR*, pages 1818–1826, 2017.

[37] Günes Erkan and Dragomir R Radev. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, 22:457–479, 2004.

[38] V. Feldman and J. Vondrák. Optimal bounds on approximation of submodular and XOS functions by juntas. *CoRR*, abs/1307.3301, 2013.

[39] Yanwei Fu, Yanwen Guo, Yanshu Zhu, Feng Liu, Chuanming Song, and Zhi-Hua Zhou. Multi-view video summarization. *IEEE Transactions on Multimedia*, 12(7):717–729, 2010.

[40] Satoru Fujishige, Takumi Hayashi, and Shigueo Isotani. The minimum-norm-point algorithm applied to submodular function minimization and linear programming. 2006.

[41] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. Convolutional sequence to sequence learning. *preprint arXiv:1705.03122*, 2017.

[42] George Giannakopoulos, Vangelis Karkaletsis, George Vouros, and Panagiotis Stamatopoulos. Summarization system evaluation revisited: N-gram graphs. *ACM Transactions on Speech and Language Processing (TSLP)*, 5(3):1–39, 2008.

[43] David Gibson, Neill Campbell, and Barry Thomas. Visual abstraction of wildlife footage using gaussian mixture models and the minimum description length criterion. In *ICPR*, volume 2. IEEE, 2002.

[44] Jennifer Gillenwater, Alex Kulesza, and Ben Taskar. Near-optimal map inference for determinantal point processes. In *Advances in Neural Information Processing Systems*, pages 2735–2743, 2012.

[45] Boqing Gong, Wei-Lun Chao, Kristen Grauman, and Fei Sha. Diverse sequential subset selection for supervised video summarization. In *Advances in Neural Information Processing Systems*, 2014.

[46] Yihong Gong and Xin Liu. Video summarization and retrieval using singular value decomposition. *Multimedia Systems*, 9(2), 2003.

[47] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, 2014.

[48] Alkis Gotovos, Hamed Hassani, and Andreas Krause. Sampling from probabilistic submodular models. In *Advances in Neural Information Processing Systems*, pages 1945–1953, 2015.

[49] Ralph Gross, Iain Matthews, Jeffrey Cohn, Takeo Kanade, and Simon Baker. Multi-pie. *Image and Vision Computing*, 28(5):807–813, 2010.

[50] 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. AUAI Press, 2011.

[51] Michael Gygli, Helmut Grabner, Hayko Riemenschneider, and Luc Van Gool. Creating summaries from user videos. In *ECCV*, pages 505–520. Springer, 2014.

[52] Michael Gygli, Helmut Grabner, and Luc Van Gool. Video summarization by learning submodular mixtures of objectives. In *CVPR*, 2015.

[53] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[54] Luis Herranz and José M Martínez. An efficient summarization algorithm based on clustering and bitstream extraction. In *ICME*. IEEE, 2009.

[55] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[56] Makoto Hirohata, Yosuke Shinnaka, Koji Iwano, and Sadaoki Furui. Sentence extraction-based presentation summarization techniques and evaluation metrics. In *Proceedings.(ICASSP'05). IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005.*, volume 1, pages I–1065. IEEE, 2005.

[57] Harold Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 24(6):417, 1933.

[58] Javier Iparraguirre and Claudio Delrieux. Speeded-up video summarization based on local features. In *Multimedia (ISM), 2013 IEEE International Symposium on*. IEEE, 2013.

[59] Rishabh Iyer and Jeff Bilmes. Submodular optimization with submodular cover and submodular knapsack constraints. In *Advances in Neural Information Processing Systems*, 2013.

[60] Rishabh Iyer and Jeffrey Bilmes. Submodular point processes with applications to machine learning. In *Artificial Intelligence and Statistics*, pages 388–397, 2015.

[61] Rishabh Iyer, Stefanie Jegelka, and Jeff Bilmes. Fast semidifferential-based submodular function optimization. In *International Conference on Machine Learning*, pages 855–863, 2013.

[62] Stefanie Jegelka and Jeff Bilmes. Submodularity beyond submodular energies: coupling edges in graph cuts. In *CVPR 2011*, pages 1897–1904. IEEE, 2011.

[63] Haojian Jin, Yale Song, and Koji Yatani. Elasticplay: Interactive video summarization with dynamic time budgets. In *Proceedings of the 2017 ACM on Multimedia Conference*, pages 1164–1172. ACM, 2017.

[64] 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.

[65] Atsushi Kanehira, Luc Van Gool, Yoshitaka Ushiku, and Tatsuya Harada. Viewpoint-aware video summarization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7435–7444, 2018.

[66] Amin Karbasi, Ehsan Kazemi, Marko Mitrovic, Morteza Zadimoghaddam, and Silvio Lattanzi. Submodular streaming in all its glory: Tight approximation, minimum memory and low adaptive complexity. 2019.

[67] Gunhee Kim, Leonid Sigal, and Eric P Xing. Joint summarization of large-scale collections of web images and videos for storyline reconstruction. In *CVPR*. IEEE, 2014.

[68] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *preprint arXiv:1412.6980*, 2014.

[69] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *preprint arXiv:1312.6114*, 2013.

[70] Katrin Kirchhoff and Jeff Bilmes. Submodularity for data selection in machine translation. In *EMNLP*, 2014.

[71] Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M Rush. Opennmt: Open-source toolkit for neural machine translation. *preprint arXiv:1701.02810*, 2017.

[72] Virginia C Klema and Alan J Laub. The singular value decomposition: Its computation and some applications. *IEEE Trans. Autom. Control*, 25(2):164–176, 1980.

[73] Suraj Kothawade, Jiten Girdhar, Chandrashekhar Lavania, and Rishabh Iyer. Deep submodular networks for extractive data summarization. *arXiv preprint arXiv:2010.08593*, 2020.

[74] Andreas Krause and Daniel Golovin. Submodular function maximization., 2014.

[75] Alex Krizhevsky and Geoffrey E Hinton. Using very deep autoencoders for content-based image retrieval. In *ESANN*, 2011.

[76] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, 2012.

[77] Alex Kulesza and Ben Taskar. Structured determinantal point processes. In *Advances in neural information processing systems*, pages 1171–1179, 2010.

[78] Alex Kulesza and Ben Taskar. Determinantal point processes for machine learning. *Foundations and Trends® in Machine Learning*, 5(2–3):123–286, 2012.

[79] William La Cava and Jason Moore. A general feature engineering wrapper for machine learning using\epsilon-lexicase survival. In *European Conference on Genetic Programming*, pages 80–95. Springer, 2017.

[80] Mirella Lapata and Frank Keller. The web as a baseline: Evaluating the performance of unsupervised web-based models for a range of nlp tasks. In *HLT-NAACL*, 2004.

[81] Chandrashekhar Lavania and Jeff Bilmes. Auto-summarization: A step towards unsupervised learning of a submodular mixture. In *Proceedings of the 2019 SIAM International Conference on Data Mining*, pages 396–404. SIAM, 2019.

[82] Chandrashekhar Lavania, Sunil Thulasidasan, Anthony LaMarca, Jeffrey Scofield, and Jeff Bilmes. A weakly supervised activity recognition framework for real-time synthetic biology laboratory assistance. In *Ubicomp*, pages 37–48. ACM, 2016.

[83] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[84] Daniel D Lee and H Sebastian Seung. Algorithms for non-negative matrix factorization. In *Advances in Neural Information Processing Systems*, 2001.

[85] Yong Jae Lee and Kristen Grauman. Predicting important objects for egocentric video summarization. *International Journal of Computer Vision*, 114(1):38–55, 2015.

[86] Ping Li, Yanwen Guo, and Hanqiu Sun. Multi-keyframe abstraction from videos. In *2011 18th IEEE International Conference on Image Processing*, pages 2473–2476. IEEE, 2011.

[87] Xuelong Li, Bin Zhao, and Xiaoqiang Lu. A general framework for edited video and raw video summarization. *IEEE Transactions on Image Processing*, 26(8):3652–3664, 2017.

[88] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out: Proceedings of the ACL-04 workshop*, volume 8, 2004.

[89] Hui Lin and Jeff Bilmes. Multi-document summarization via budgeted maximization of submodular functions. In *HLT-NAACL*, pages 912–920, 2010.

[90] 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.

[91] Hui Lin and Jeff Bilmes. Learning mixtures of submodular shells with application to document summarization. In *UAI*. AUAI, 2012.

[92] Hui Lin, Jeff Bilmes, and Shasha Xie. Graph-based submodular selection for extractive summarization. In *2009 IEEE Workshop on Automatic Speech Recognition & Understanding*, pages 381–386. IEEE, 2009.

[93] Hui Lin, Jeff Bilmes, and Shasha Xie. Graph-based submodular selection for extractive summarization. In *Proc. IEEE Automatic Speech Recognition and Understanding (ASRU)*, Merano, Italy, December 2009.

[94] Hui Lin and Jeff A Bilmes. Learning mixtures of submodular shells with application to document summarization. *preprint arXiv:1210.4871*, 2012.

[95] Yuzong Liu, Kai Wei, Katrin Kirchhoff, Yisong Song, and Jeff Bilmes. Submodular feature selection for high-dimensional acoustic score spaces. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 7184–7188. IEEE, 2013.

[96] László Lovász. Matroid matching and some applications. *Journal of Combinatorial Theory, Series B*, 28(2):208–236, 1980.

[97] Yingdong Lu and Jing-Sheng Song. Order-based cost optimization in assemble-to-order systems. *Operations Research*, 53(1):151–169, 2005.

[98] Hans Peter Luhn. The automatic creation of literature abstracts. *IBM Journal of research and development*, 2(2):159–165, 1958.

[99] Odile Macchi. The coincidence approach to stochastic point processes. *Advances in Applied Probability*, 7(1):83–122, 1975.

[100] Behrooz Mahasseni, Michael Lam, and Sinisa Todorovic. Unsupervised video summarization with adversarial lstm networks. *CVPR*, 2017.

[101] Angshul Majumdar and Aditay Tripathi. Asymmetric stacked autoencoder. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 911–918. IEEE, 2017.

[102] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, and Ian Goodfellow. Adversarial autoencoders. In *International Conference on Learning Representations*, 2016.

[103] Jonathan Malkin and Jeff Bilmes. Ratio semi-definite classifiers. In *ICASSP*. IEEE, 2008.

[104] Xudong Mao, Qing Li, Haoran Xie, Raymond Y.K. Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[105] S Thomas McCormick. Submodular function minimization. *Handbooks in operations research and management science*, 12:321–391, 2005.

[106] Ryan McDonald. A study of global inference algorithms in multi-document summarization. In *European Conference on Information Retrieval*, pages 557–564. Springer, 2007.

[107] Shaohui Mei, Zhiyong Wang, Mingyi He, and Dagan Feng. Resource restricted online video summarization with minimum sparse reconstruction. In *Picture Coding Symposium (PCS), 2015*. IEEE, 2015.

[108] Rada Mihalcea. Graph-based ranking algorithms for sentence extraction, applied to text summarization. In *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*, page 20. Association for Computational Linguistics, 2004.

[109] Rada Mihalcea and Paul Tarau. Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing*, 2004.

[110] M. Minoux. Accelerated greedy algorithms for maximizing submodular set functions. *Optimization Techniques*, pages 234–243, 1978.

[111] Pitu Mirchandani and Richard Francis. *Discrete location theory*. 1990.

[112] Baharan Mirzasoleiman, Ashwinkumar Badanidiyuru, and Amin Karbasi. Fast constrained submodular maximization: Personalized data summarization. In *International conference on machine learning*, pages 1358–1367, 2016.

[113] Baharan Mirzasoleiman, Ashwinkumar Badanidiyuru, Amin Karbasi, Jan Vondrák, and Andreas Krause. Lazier than lazy greedy. In *AAAI*, pages 1812–1818, 2015.

[114] Baharan Mirzasoleiman, Ashwinkumar Badanidiyuru, Amin Karbasi, Jan Vondrák, and Andreas Krause. Lazier than lazy greedy. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[115] Baharan Mirzasoleiman, Amin Karbasi, and Andreas Krause. Deletion-robust submodular maximization: Data summarization with the right to be forgotten. In *International conference on machine learning*, 2017.

[116] Baharan Mirzasoleiman, Amin Karbasi, Rik Sarkar, and Andreas Krause. Distributed submodular maximization: Identifying representative elements in massive data. In *Advances in Neural Information Processing Systems*, pages 2049–2057, 2013.

[117] Ibrahim F Moawad and Mostafa Aref. Semantic graph reduction approach for abstractive text summarization. In *Computer Engineering & Systems (ICCES), 2012 Seventh International Conference on*, pages 132–138. IEEE, 2012.

[118] Kiyohito Nagano, Yoshinobu Kawahara, and Kazuyuki Aihara. Size-constrained submodular minimization through minimum norm base. In *International Conference on Machine Learning*, pages 977–984, 2011.

[119] Kiyohito Nagano, Yoshinobu Kawahara, and Satoru Iwata. Minimum average cost clustering. In *Advances in Neural Information Processing Systems*, pages 1759–1767, 2010.

[120] Zwe Naing and Ehsan Elhamifar. Procedure completion by learning from partial summaries. *BMVC*, 2020.

[121] Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *AAAI*, pages 3075–3081, 2017.

[122] Mukund Narasimhan and Jeff Bilmes. A submodular-supermodular procedure with applications to discriminative structure learning. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*, pages 404–412. AUAI Press, 2005.

[123] Mukund Narasimhan, Nebojsa Jojic, and Jeff A Bilmes. Q-clustering. In *Advances in Neural Information Processing Systems*, pages 979–986, 2006.

[124] George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. An analysis of approximations for maximizing submodular set functions –I. *Math. Program.*, 14(1), 1978.

[125] The news 2013 data set, 2013.

[126] Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, and Andrew Y Ng. Multimodal deep learning. In *International conference on machine learning*, 2011.

[127] Chongjia Ni, Lei Wang, Haibo Liu, Cheung-Chi Leung, Li Lu, and Bin Ma. Submodular data selection with acoustic and phonetic features for automatic speech recognition. In *ICASSP*. IEEE, 2015.

[128] The staggering cost of training sota ai models, accesed october, 2020, 2019.

[129] Aude Oliva and Antonio Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International journal of computer vision*, 42(3):145–175, 2001.

[130] Shun-Hsing Ou, Chia-Han Lee, V Srinivasa Somayazulu, Yen-Kuang Chen, and Shao-Yi Chien. On-line multi-view video summarization for wireless video sensor network. *IEEE Journal of Selected Topics in Signal Processing*, 9(1):165–179, 2014.

[131] Jian-Quan Ouyang, Jin-Tao Li, and Yong-Dong Zhang. Replay boundary detection in mpeg compressed video. In *Machine Learning and Cybernetics, 2003 International Conference on*, volume 5. IEEE, 2003.

[132] Rameswar Panda, Abir Das, and Amit K Roy-Chowdhury. Embedded sparse coding for summarizing multi-view videos. In *2016 IEEE international conference on image processing (ICIP)*, pages 191–195. IEEE, 2016.

[133] Rameswar Panda, Abir Dasy, and Amit K Roy-Chowdhury. Video summarization in a multi-view camera network. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 2971–2976. IEEE, 2016.

[134] Shameem A Puthiya Parambath, Nishant Vijayakumar, and Sanjay Chawla. Saga: A submodular greedy algorithm for group recommendation. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[135] Neal Parikh and Stephen Boyd. Proximal algorithms. *Foundations and Trends in optimization*, 1(3):127–239, 2014.

[136] Sun Park and Ju-Hong Lee. Topic-based multi-document summarization using non-negative matrix factorization and k-means. *Journal of KIISE: Software and Applications*, 35(4):255–264, 2008.

[137] Karl Pearson. On lines and planes of closest fit to systems of point in space. *Philos. Mag.*, 2(11):559–572, 1901.

[138] Xiaojiang Peng, Limin Wang, Xingxing Wang, and Yu Qiao. Bag of visual words and fusion methods for action recognition: Comprehensive study and good practice. *Computer Vision and Image Understanding*, 2016.

[139] Nathalie Peyrard and Patrick Bouthemy. Motion-based selection of relevant video segments for video summarization. *Multimedia Tools and Applications*, 26(3), 2005.

[140] Yanmin Qian, Nanxin Chen, Heinrich Dinkel, and Zhizheng Wu. Deep feature engineering for noise robust spoofing detection. *IEEE/ACM Trans. Audio, Speech, Language Process.*, 2017.

[141] Pavan Kartheek Rachabathuni. A survey on abstractive summarization techniques. In *Inventive Computing and Informatics (ICICI), International Conference on*, pages 762–765. IEEE, 2017.

[142] C Radhakrishna Rao. The utilization of multiple measurements in problems of biological classification. *J. Royal Stat. Soc. Series B (Methodological)*, 10(2):159–203, 1948.

[143] Mrigank Rochan and Yang Wang. Video summarization by learning from unpaired data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7902–7911, 2019.

[144] Veronika Rocková, Gemma Moran, and Edward George. Determinantal regularization for ensemble variable selection. In *AISTATS*, 2016.

[145] Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

[146] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 2014.

[147] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *Int. J. Comput. Vis*, 115(3):211–252, 2015.

[148] Gerard Salton and Chris Buckley. Term weighting approaches in automatic text retrieval. Technical report, Cornell University, 1987.

[149] Fabrizio Sebastiani. Machine learning in automated text categorization. *CSUR*, 34(1):1–47, 2002.

[150] Sunchit Sehgal, Badal Kumar, Lakshay Rampal, Ankit Chaliya, et al. A modification to graph based approach for extraction based automatic text summarization. In *Progress in Advanced Computing and Intelligent Engineering*, pages 373–378. Springer, 2018.

[151] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *preprint arXiv:1312.6229*, 2013.

[152] Zuo-Jun Max Shen, Collette Coullard, and Mark S Daskin. A joint location-inventory model. *Transportation science*, 37(1):40–55, 2003.

[153] Ayesha Siddiqua and Guoliang Fan. Asymmetric supervised deep autoencoder for depth image based 3d model retrieval. In *2019 IEEE Visual Communications and Image Processing (VCIP)*, pages 1–4. IEEE, 2019.

[154] Ruben Sipos, Adith Swaminathan, Pannaga Shivaswamy, and Thorsten Joachims. Temporal corpus summarization using submodular word coverage. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 754–763, 2012.

[155] Shengli Song, Haitao Huang, and Tongxiao Ruan. Abstractive text summarization using lstm-cnn based deep learning. *Multimedia Tools and Applications*, 78(1):857–875, 2019.

[156] Yale Song, Jordi Vallmitjana, Amanda Stent, and Alejandro Jaimes. Tvsum: Summarizing web videos using titles. In *CVPR*, 2015.

[157] Chris Stauffer and W Eric L Grimson. Adaptive background mixture models for real-time tracking. In *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*, volume 2, pages 246–252. IEEE, 1999.

[158] Min Sun, Ali Farhadi, and Steve Seitz. Ranking domain-specific highlights by analyzing edited videos. In *ECCV*. Springer, 2014.

[159] Yi Sun, Xiaogang Wang, and Xiaoou Tang. Deep learning face representation from predicting 10,000 classes. In *CVPR*. IEEE, 2014.

[160] Maxim Sviridenko. A note on maximizing a submodular set function subject to a knapsack constraint. *Operations Research Letters*, 32(1):41–43, 2004.

[161] Zoya Svitkina and Lisa Fleischer. Submodular approximation: Sampling-based algorithms and lower bounds. *SIAM Journal on Computing*, 40(6):1715–1737, 2011.

[162] Michael J Swain and Dana H Ballard. Color indexing. *International journal of computer vision*, 7(1), 1991.

[163] Jiwei Tan, Xiaojun Wan, and Jianguo Xiao. Abstractive document summarization with a graph-based attentional neural model. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1171–1181, 2017.

[164] Joshua B Tenenbaum, Vin De Silva, and John C Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

[165] George Trigeorgis, Konstantinos Bousmalis, Stefanos Zafeiriou, and Bjoern Schuller. A deep semi-nmf model for learning hidden representations. In *International conference on machine learning*, 2014.

[166] Ba Tu Truong and Svetha Venkatesh. Video abstraction: A systematic review and classification. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 3(1), 2007.

[167] 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*, 2014.

[168] Pavan Turaga, Rama Chellappa, Venkatramana S Subrahmanian, and Octavian Udrea. Machine recognition of human activities: A survey. *Circuits and Systems for Video Technology, IEEE Transactions on*, 18(11), 2008.

[169] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *JMLR*, 11(Dec):3371–3408, 2010.

[170] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *CVPR*, volume 1. IEEE, 2001.

[171] Jan Vondrák. Submodularity in combinatorial optimization. 2007.

[172] Jan Vondrák. Optimal approximation for the submodular welfare problem in the value oracle model. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 67–74, 2008.

[173] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[174] Kai Wei, Rishabh Iyer, and Jeff Bilmes. Fast multi-stage submodular maximization. In *International conference on machine learning*, pages 1494–1502, 2014.

[175] Kai Wei, Rishabh Iyer, and Jeff Bilmes. Submodularity in data subset selection and active learning. In *International Conference on Machine Learning*, pages 1954–1963, 2015.

[176] Kai Wei, Yuzong Liu, Katrin Kirchhoff, Christopher Bartels, and Jeff Bilmes. Submodular subset selection for large-scale speech training data. In *ICASSP*. IEEE, 2014.

[177] Kai Wei, Yuzong Liu, Katrin Kirchhoff, and Jeff Bilmes. Using document summarization techniques for speech data subset selection. In *HLT-NAACL*, 2013.

[178] Kai Wei, Yuzong Liu, Katrin Kirchhoff, and Jeff Bilmes. Unsupervised submodular subset selection for speech data. In *ICASSP*. IEEE, 2014.

[179] Kuoliang Wu, Deng Cai, and Xiaofei He. Multi-label active learning based on submodular functions. *Neurocomputing*, 313:436–442, 2018.

[180] Jia Xu, Lopamudra Mukherjee, Yin Li, Jamieson Warner, James M Rehg, and Vikas Singh. Gaze-enabled egocentric video summarization via constrained submodular maximization. In *CVPR*. IEEE, 2015.

[181] Ronald R Yager and Jordán Pascual Espada. New advances in the internet of things. 2017.

[182] Ting Yao, Tao Mei, and Yong Rui. Highlight detection with pairwise deep ranking for first-person video summarization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 982–990, 2016.

[183] Wenpeng Yin and Yulong Pei. Optimizing sentence modeling and selection for document summarization. In *IJCAI*, pages 1383–1389, 2015.

[184] Xiao-Dong Yu, Lei Wang, Qi Tian, and Ping Xue. Multilevel video representation with application to keyframe extraction. In *Multimedia Modelling Conference, 2004. Proceedings. 10th International*. IEEE, 2004.

[185] Yisong Yue and Carlos Guestrin. Linear submodular bandits and their application to diversified retrieval. In *Advances in Neural Information Processing Systems*, 2011.

[186] Ke Zhang, Wei-Lun Chao, Fei Sha, and Kristen Grauman. Video summarization with long short-term memory. In *European Conference on Computer Vision*, pages 766–782. Springer, 2016.

[187] Bin Zhao, Xuelong Li, and Xiaoqiang Lu. Hsa-rnn: Hierarchical structure-adaptive rnn for video summarization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7405–7414, 2018.

[188] Bin Zhao and Eric P Xing. Quasi real-time summarization for consumer videos. In *CVPR*. IEEE, 2014.

[189] Wenyi Zhao, Rama Chellappa, P Jonathon Phillips, and Azriel Rosenfeld. Face recognition: A literature survey. *ACM computing surveys (CSUR)*, 35(4), 2003.

[190] Alice Zheng. *Mastering Feature Engineering: Principles and Techniques for Data Scientists*. O'Reilly Media, 2016.