

Heterogeneous Model Collaborations in Language Models

Xiaochuang Han

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

Doctor of Philosophy

University of Washington
2025

Reading Committee:
Yulia Tsvetkov, Chair
Pang Wei Koh
Luke Zettlemoyer

Program Authorized to Offer Degree:
Computer Science and Engineering

© Copyright 2025

Xiaochuang Han

University of Washington

Abstract

Heterogeneous Model Collaborations in Language Models

Xiaochuang Han

Chair of the Supervisory Committee:

Yulia Tsvetkov

Computer Science and Engineering

Language is central to human intelligence and has similarly become integral to artificial intelligence, particularly through advancements in natural language processing and large language models. Modern LLMs focus on model scaling, using one generic and monolithic structure and substantially increasing model sizes. This is a highly homogeneous process—one single type of neural modules is used towards all tasks with the same optimization objective. While scaling LLMs has achieved remarkable successes, it remains challenging for achieving swift control over LLM outputs, reducing hallucinations, enabling fundamental weak-to-strong generalization, and adapting seamlessly to multi-modalities beyond language. In this thesis, I aim to develop novel heterogeneous model collaboration paradigms that address each of the LLM challenges above. In the first part, I introduce a diffusion-based generative LLM that collaborates with discriminative text classifiers for a nuanced controllability at inference time. In the second part, I present a sampling algorithm through a collaboration between two contrastively contextualized language models, reducing hallucinations and enhancing faithfulness to input context. In the third part, I introduce two diffusion LLMs, a large general-purpose model and a much smaller specialized model, collaborating together for strong ensemble performance and robust weak-to-strong generalization compared to autoregressive approaches. Finally, I present an autoregressive LLM architecture learning to generate images and videos through collaboration with non-neural, multimodal compression codecs, overcoming modality barriers and expanding the multimodal generative capabilities of LLMs. Overall, this thesis aims to advance NLP and LLMs beyond monolithic model scaling through modular, heterogeneous model collaborations across language and multimodal domains, ultimately contributing to the pursuit of general machine intelligence.

Acknowledgements

My PhD would not be possible without the support of my mentors, collaborators, friends, and family.

I want to thank Yulia Tsvetkov for being a wonderful advisor and friend—throughout our time at CMU and all the way to the completion of my PhD at UW. She consistently encouraged me and offered unwavering support, especially when I struggled or shifted directions in my projects. Her guidance has inspired me not only in research but also in my career and life.

I want to thank Pang Wei Koh and Luke Zettlemoyer for being incredible mentors. Pang Wei has inspired me both through our engaging discussions over the past year and through his earlier research, which influenced my early work even before we met. Luke’s kind and endless support, especially during the Meta AI Mentorship program, has been invaluable. Without it, much of my research might not have happened at all.

I want to thank Jacob Eisenstein for introducing me to research and NLP. His research style and taste have been influencing and supporting me well beyond our time collaborating at Georgia Tech.

I want to thank everyone in the UW NLP and CSE community—especially Weijia Shi, Alisa Liu, Jiacheng Liu, Shangbin Feng, Preston Jiang, Rock Pang, Kaiming Cheng, Inna Lin, Jingwei Ma, Tianxing He, Sachin Kumar, Vidhisha Balachandran, Chan Young Park, Melanie Sclar, Orevaoghene Ahia, Tsvetshop, and Koh-lab. My PhD journey has been so enjoyable because of you.

I want to thank the Meta FAIR team—especially Chunting Zhou, Lili Yu, Tianlu Wang, Omer Levy, and Marjan Ghazvininejad. My experience there was incredibly enlightening and played a big role in shaping my next career step.

I want to thank all of my other mentors, collaborators, and friends, especially Umashanthi Pavalanathan and Chenhao Tan for supporting me before my PhD journey.

Finally, I want to thank my parents, Song Han and Yingchun Deng, and my fiancée, Yuejun Han. You have been there every step of the way, always reminding me that no matter how my research or work turns out, I'll always be the happiest person in the world.

To my parents

Contents

1	Introduction	13
1.1	Thesis Statement	15
1.2	Thesis Overview	16
2	Background	19
2.1	Standard training of language models	19
2.2	Standard inference of language models	20
3	Collaboration between Diffusion Language Models and Discriminative Classifiers	22
3.1	Background	24
3.1.1	Diffusion model	24
3.1.2	Autoregressive LM	25
3.2	SSD-LM	25
3.2.1	Training	26
3.2.2	Decoding	29
3.2.3	Additional details	31
3.3	Experiments	31
3.3.1	SSD-LM pretraining setup	31
3.3.2	Unconstrained text generation	32
3.3.3	Controlled text generation	36
3.4	Related work	38
3.5	Conclusion and continued discussion	40
3.5.1	Compute limitations	40
3.5.2	A contrastive interpretation of the training loss	41

3.5.3	Connection between our decoding algorithm and the DDPM decoding	42
3.5.4	Detailed setup of the comparison with Diffusion-LM	42
3.5.5	Additional results	43
4	Collaboration between Language Models under Contrastive Contexts	51
4.1	Method	53
4.1.1	Background	53
4.1.2	Context-aware decoding	53
4.2	Experimental setup	54
4.2.1	Datasets and metrics	54
4.2.2	Models and baselines	55
4.3	Results	56
4.4	Analysis	58
4.5	Related work	60
4.6	Conclusion and continued discussion	61
4.6.1	Framework limitations	61
4.6.2	Qualitative analysis	62
5	Collaboration between Diffusion Language Models for Weak-to-strong Generalization	64
5.1	Background	65
5.2	SSD-2	67
5.2.1	Algorithmic improvements over Ssd-LM	69
5.2.2	Inference-time collaboration	71
5.3	Experimental setup	73
5.4	Experiments	76
5.4.1	Inference-time collaboration	76
5.4.2	Ablation study: SSD-2 as a standalone diffusion chat model	77
5.5	Related work	78
5.6	Conclusion and continued discussion	80
5.6.1	Method limitations	80
5.6.2	Eliminating the sampling of context size c in training	81

5.6.3	Pretraining hyperparameters	82
5.6.4	Pretraining losses	83
5.6.5	Decoding speed	83
5.6.6	Standalone SSD-2 finetuned with DOLLY	84
5.6.7	Qualitative examples	84
5.6.8	Evaluation templates	85
6	Collaboration between Language Models and Multimodal Compression Codecs	88
6.1	Background	90
6.1.1	Pixel values: ImageGPT	91
6.1.2	Latent codes: Vector-Quantization models	91
6.2	JPEG-LM and AVC-LM	92
6.2.1	Canonical codecs: JPEG and AVC/H.264	93
6.2.2	JPEG-LM and AVC-LM	93
6.3	Experimental setup	95
6.3.1	JPEG-LM	95
6.3.2	AVC-LM	95
6.3.3	Image generation baselines	96
6.4	Results	96
6.4.1	Qualitative analysis	98
6.4.2	Quantitative results	98
6.4.3	Why JPEG-LM? A case study over long-tail elements in images	101
6.4.4	Proof-of-concept video generation	102
6.5	Related Work	103
6.6	Conclusion and continued discussion	104
6.6.1	Design discussion	105
6.6.2	More qualitative examples from JPEG-LM	107
6.6.3	More qualitative examples from AVC-LM	107
6.6.4	Detailed configurations for the canonical codecs	107
7	Conclusions	111
7.1	Summary of Contributions	111
7.2	Future Work	112

Chapter 1

Introduction

Language is the hallmark of human intelligence. With no other animals possessing a comparable language system, humans use language to accurately describe, predict, and convey the world around us. Language is also a hallmark of artificial intelligence (AI). From the early proposal of the imitation game (Turing, 1950) (i.e., the Turing Test) to modern AI assistants (OpenAI, 2023), language has been integral to building human-like intelligence. In particular, the language aspect of artificial intelligence is often referred to as natural language processing (NLP). One of the most significant advancements in modern NLP is large language models (LLMs), generative deep neural models that not only produce text for traditional NLP tasks such as machine translation and summarization but also engage in useful conversations with humans to answer questions and draft detailed reports (Radford et al., 2019; Ouyang et al., 2022).

Modern LLMs emphasize scaling the models, using one general and monolithic architecture and substantially increasing model size and the amount of data used across domains. Such framework is highly homogeneous — one single type of model building blocks (Vaswani et al., 2017) is used towards all tasks with the same optimization objective (Brown et al., 2020). Despite their success in commercial chatbots (Bubeck et al., 2023), simply scaling large language models remains insufficient or costly for achieving swift control over outputs (Dathathri et al., 2020), reducing hallucinations (Min et al., 2023a), enabling fundamental weak-to-strong generalization (Burns et al., 2023), and adapting seamlessly to multi-modalities beyond language (Yu et al., 2021).

On the other hand, the source of human intelligence—our brains—offers an inspiring alternative in structure. The human brain contains anatomically and functionally

specialized regions or units that collaborate to solve various perceptual and decision-making tasks (Stein and Stanford, 2008; Steinmetz et al., 2019). For example, from a mechanistic perspective, the input and output functions for language rely on the visual and auditory cortices, which process perceptual information essential for encoding language, and on the motor cortex, which generates the motor commands necessary for producing spoken or written language (Kandel et al., 2000). From a developmental perspective, we rely on the hippocampal memory system to learn and retain language structure and symbolic meaning, which is especially critical for adapting to new phrases and the evolving cultural characteristics of language over time (Moser et al., 2015; Rowland et al., 2016). Modern theories of the human brain attribute our unique language capabilities to three heterogeneous systems: a language implementation system, a mediational system, and a conceptual system (Damasio and Damasio, 1992).¹ Notably, each of these systems is also supported by largely heterogeneous components. For example, the language implementation system consists of Wernicke’s and Broca’s areas, which are responsible for language comprehension and production, respectively.

Inspired by the heterogeneous collaborations in human brains and towards addressing the problems of LLMs such as controllability, faithfulness, generalizability, and beyond-language adaptability, this thesis focuses on solutions from a heterogeneity perspective. Analogous to the collaborations in the language implementation system, mediational system, and conceptual system in human brains, I rethink how a variety of different NLP models, both generative and non-generative, autoregressive and non-autoregressive, neural and non-neural, language and multi-modal, can collaborate with each other and manifest fundamental improvements over single large language models monolithically optimized for single objectives.

¹The language implementation system is located in a specific brain region that controls fundamental language functions, such as understanding and generating speech and grammar. The conceptual system consists of distributed brain areas responsible for learning, memory, and conceptual knowledge, not necessarily related to language. Meanwhile, the mediational system surrounds the implementation system and facilitates communication between the implementation and conceptual systems.

1.1 Thesis Statement

My primary goal of this thesis is to develop machine learning methods that enhance single language models through heterogeneous model collaborations, improving the system’s controllability, faithfulness, generalizability, and adaptability beyond the language modality. To achieve this, I introduce fundamental innovations in language model architectures in training, model collaboration algorithms at inference time, and foundational representation learning for multi-modalities. I organize the thesis chapters by LLMs + X, highlighting the type of collaboration enhancing the language model.

Language model + discriminative models In NLP, language models are generative and primarily used for language generation, whereas classifiers of various text attributes are discriminative and reflect language understanding capabilities. In the first part of the thesis, I present *language model + discriminative models*. This generative-discriminative collaboration between language models and classifiers enhances the system’s controllability, enabling outputs with desired attributes, particularly during inference. To facilitate this collaboration, I design a diffusion-based language model that iteratively generates text non-autoregressively with bidirectional context.

Language model + contrastive contexts In the second part of the thesis, I present *language model + contrastive contexts*. This model collaboration involves two language models decoding together but each provided with contrastive contexts. This unique retrieval and routing improves the faithfulness of standard language models and reduces model hallucination in various summarization and knowledge editing tasks, without modifying the weights of language models.

Language model + weak-strong collaboration In the third part of this thesis, I present *language model + weak-strong collaboration*. I demonstrate how small language models with specialized enhancements can collaborate with large general language models to make them even stronger — a weak-to-strong generalization aspiration. In pursuit of such models, I design and train two diffusion-based language models of drastically different scales while enhancing the small model with specialized user inputs.

The diffusion approach is fundamentally more effective than standard autoregressive language models under the same collaboration setup.

Language model + multimodal codecs Finally, in the fourth part of this thesis, I present *language model + multimodal codecs*. I demonstrate that a language model architecture designed for text generation can be trained to generate images and videos, by learning from representations in multimodal compression codecs. This collaboration between language models and multimodal codecs enables a direct adaptation of generative language models into generative multimodal models, highlighting the potential of LLMs as a step toward general intelligence.

My solutions are both inspired by and contribute to advancements in natural language processing and machine learning research. These heterogeneous model collaborations have applications across various language and multimodal domains and can ultimately drive progress toward general machine intelligence.

1.2 Thesis Overview

I begin with relevant background (§2) on the standard architecture, training objective, and sampling algorithm of LLMs. The rest of the thesis consists of four LLMs + X chapters, highlighting the type of collaboration enhancing the language model.

Part I: (Diffusion) LLMs + discriminative models Standard language models are autoregressive. To obtain nuanced controls of autoregressive LLM generations, classifiers had been used to manipulate the model’s output space. However, this has been a very challenging procedure due to standard LLMs’ autoregressive sampling — often the desired attribute is defined over the entire sequence, but the controls can only be done at the level of each token. In §3, I present SSD-LM — a diffusion-based language model with two key design choices. First, SSD-LM is semi-autoregressive, iteratively generating blocks of text, allowing for flexible output length at decoding time while enabling local bidirectional context updates. Second, it is simplex-based, performing diffusion on the natural vocabulary space rather than a learned latent space, allowing us to incorporate classifier guidance and modular control using off-the-shelf classifiers

without any adaptation. SSD-LM is evaluated on unconstrained text generation as well as controllable text generation benchmarks. I show SSD-LM is a strong language model that can collaborate with discriminative attribute classifiers at inference time without any tuning. This work was published at ACL 2023 (Han et al., 2022).

Part II: LLMs + contrastive contexts Standard language models struggle to pay enough attention to the input context. Particularly, the vanilla pretrained LLMs without downstream alignment tuning often generate texts that are unfaithful or contain hallucinations. In §4, I present context-aware decoding (CAD), which samples using two language models in collaboration to mitigate the issue. Without any modifications to the pretrained models’ weights, the two language models are provided with different inputs, one with and one without the key context the system should focus on. The decoding follows a contrastive output distribution that amplifies the difference between the output probabilities when used with and without context. On summarization and knowledge conflict tasks, I show CAD greatly helps LLMs to be faithful to the given context and reduce hallucination. This work was published at NAACL 2024 (Shi and Han et al., 2023).

Part III: (Diffusion) LLMs + weak-strong collaboration Standard language models is not well-compatible with ensembling at inference time. Especially for models with disparate sizes, the small and specialized model rarely helps the large and general model in achieving an even stronger model — a weak-to-strong generalization dilemma. In §5, I present SSD-2, an approach to easily ensemble at inference time a large general-purpose diffusion LLM with smaller, but specialized and contextualized diffusion LLMs. SSD-2 facilitates a collaboration between a standard sized LLM and 100x smaller models that can be customized and deployed by individual users. I show that compared to autoregressive models, the collaboration between diffusion LLMs is more effective, leading to higher-quality model outputs due to their ability to dynamically incorporate bi-directional contexts. This work was published at NAACL 2024 (Han et al., 2023a).

Part IV: LLMs + multimodal codecs Standard language models cannot generate images or videos. However, it is not the autoregressive nature of LLM that prevents it from being a generative visual model — the problem is with the visual data where

the representations are naturally continuous, unlike language. The crux of adapting autoregressive training in language generation to visual generation is discretization — representing continuous data like images and videos as discrete tokens. Common methods of discretizing images and videos include modeling raw pixel values, which are prohibitively lengthy, or vector quantization, which requires convoluted pre-hoc training. In §6, I propose to directly model images and videos as compressed files saved on computers via canonical codecs such as JPEG and AVC. Using the default LLM architecture without any vision-specific modifications, I present JPEG-LM to generate images and AVC-LM to generate videos, directly outputting compressed file bytes in JPEG and AVC formats. JPEG-LM shows strong efficacy in image generation evaluations and especially excels in generating long-tail visual elements. This collaboration between a neural LLM architecture and non-neural, canonical compression codecs helps lower the barriers between language generation and visual generation, facilitating universal multimodal LLMs. This work is currently in submission ([Han et al., 2024](#)).

Chapter 2

Background

Language models (LMs) is a central theme throughout this thesis. In this chapter, I present a brief overview of standard generative language models from both training and inference perspective. This overview establishes a common foundation for all solutions presented in the thesis, while each chapter also includes more detailed background relevant to its specific research questions.

2.1 Standard training of language models

Generative language models are capable of text generation, outputting a sequence of texts based on a given context. It is a universal form of tasks in NLP, unifying text classification, sequence labeling, parsing, translation, question answering, among others (Radford et al., 2019).

The training of language models follows the scheme of maximum likelihood estimation.¹ Given a sequence of text tokens $\mathbf{x} = \mathbf{x}_{1:N} = (x_1, x_2, \dots, x_N)$,² a language model θ is trained to maximize the probability of the token sequence,

$$P(\mathbf{x}; \theta) = \prod_{i=1}^N P(x_i | \mathbf{x}_{<i}; \theta)$$

¹In the realm of production LLMs and chatbots, the training background I present here corresponds to the pretraining stage of LLMs, whereas it is often followed by an alignment stage using additional reinforcement learning objectives (Ouyang et al., 2022).

²Text tokens are usually obtained through tokenization algorithms like byte-pair encoding (Sennrich et al., 2016), breaking original texts to pieces of words.

The language model θ is usually a neural network. The optimization of θ is instantiated as a stochastic gradient descent process over the cross entropy loss \mathcal{L} , with \mathcal{V} representing the vocabulary of all possible tokens and s_θ denoting the scores or logits the model θ predicts for each vocabulary entry,

$$\mathcal{L} = \sum_{i=1}^N -\log P(x_i | \mathbf{x}_{<i}; \theta) = \sum_{i=1}^N -\log\left(\frac{\exp s_\theta(X = x_i | \mathbf{x}_{<i})}{\sum_{x' \sim \mathcal{V}} \exp s_\theta(X = x' | \mathbf{x}_{<i})}\right)$$

Standard LM training has two key aspects: discrete and autoregressive. The input text tokens are discrete variables that draw from a finite vocabulary \mathcal{V} . The language model θ autoregressively predicts the current token x_i with all previous context $\mathbf{x}_{<i}$ ahead of the current token's position. *We will relax both of these aspects in our solutions of generative language model training.*

The architecture of language models is an active area of research. The most prominent one in the current field and used throughout this thesis is transformers (Vaswani et al., 2017). It has five key components: (1) embedding layer (2) normalization layer, (3) self-attention layer, (4) feed-forward layer, (5) projection layer. Notably, the self-attention layer is the crux of transformers, converting layer input \mathbf{h}_{in} to layer output \mathbf{h}_{out} ,

$$\mathbf{h}_{\text{out}} = \text{O}\left[\text{softmax}\left(\frac{\text{Q}(\mathbf{h}_{\text{in}})\text{K}(\mathbf{h}_{\text{in}})^T + M}{\sqrt{d}}\right)\text{V}(\mathbf{h}_{\text{in}})\right]$$

$\text{Q}, \text{K}, \text{V}, \text{O}$ are linear projections to obtain queries, keys, and values for each hidden state in the sequence. M is a causal attention mask shaped as $\begin{bmatrix} 0 & -\infty & \dots & -\infty \\ 0 & 0 & \dots & -\infty \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 \end{bmatrix}$ to enforce autoregressive attention to the context. Throughout all the solutions I present, the key designs of transformers are retained. The selection of the attention mask M and the cross-entropy loss function will be re-designed in the diffusion solutions I present.

2.2 Standard inference of language models

With a trained language model, standard inference of language models sample individual discrete token one at a time from left to right, based on a given context \mathbf{x} and the previously sampled tokens. At time step i , one token $y_i \in \mathcal{V}$ is sampled as the output,

$$y_i \sim P(y_i | \mathbf{y}_{<i}, \mathbf{x}; \theta)$$

$[x, y_{<i>i}, y_i]$ are concatenated as the context for next token's generation. Common sampling strategies in modern language models include temperature sampling (Ackley et al., 1985) and nucleus sampling (Holtzman et al., 2020).

This thesis has a focus on inference-time methods for language models. In §3, §4, and §5, I present solutions to common problems of standard autoregressive inference of language models, such as controllability, hallucination, joint decoding, etc. For the solutions in §3 and §5, I particularly design non-autoregressive language models to support the specific inference. In §6, I present how language model inference can be directly adapted for multimodal generation.

Chapter 3

Collaboration between Diffusion Language Models and Discriminative Classifiers

This chapter discusses work previously published in [Han et al. \(2022\)](#) at ACL 2023 — SSD-LM: Semi-autoregressive Simplex-based Diffusion Language Model for Text Generation and Modular Control.

Diffusion models ([Sohl-Dickstein et al., 2015](#)), trained to iteratively refine noised inputs, have recently emerged as powerful tools for generative modeling in several continuous-valued domains such as images ([Ho et al., 2020](#)), audio ([Kong et al., 2021](#)), video ([Ho et al., 2022](#)), among others. Attempts to adapt them for discrete domains such as text data, however, have only had limited success: prior work have shown to be promising on specialized cases and small datasets ([Hoogeboom et al., 2021](#); [Austin et al., 2021](#); [Li et al., 2022b](#); [Chen et al., 2022b](#)), but diffusion models for text still underperform (and thus are not widely adopted) compared to autoregressive language models (AR-LMs) which remain the state-of-the-art general purpose text generators ([Radford et al., 2019](#); [Brown et al., 2020](#)).

Despite potential advantages of diffusion models for text, there are two key challenges. First, diffusion models generate text non-autoregressively, i.e., they generate (and update) the entire sequence simultaneously rather than token by token left-to-right. Although this property is useful in practice since each output token is informed by a

broader bi-directional context (Lee et al., 2018; Ghazvininejad et al., 2019), it requires pre-defining an output sequence length. This limits the flexibility and applicability of trained models. On the other hand, non-autoregressive training with long sequences is expensive and difficult to optimize. In this chapter, we propose a *semi-autoregressive* solution which strikes a balance between length flexibility and the ability to alter previously generated tokens.

A major advantage of diffusion models over the current standard of autoregressive LMs is their post-hoc controllability using guidance from auxiliary models such as style classifiers (Dhariwal and Nichol, 2021). However, controllability is hard to achieve without compromises in modularity in diffusion-based LMs for text. To enable diffusion generation into discrete text rather than continuous modalities, prior approaches have employed different approximations, e.g., training with embeddings, character, or byte-level methods (Li et al., 2022b; Hoogeboom et al., 2021; Austin et al., 2021; Chen et al., 2022b). In contrast, existing mainstream LMs and the guidance classifiers they derive often operate at a sub-word level with sub-word representations trained jointly with the language model (Devlin et al., 2019; Liu et al., 2019; Raffel et al., 2020). Subsequently, changing the input representations to characters or embeddings requires developing guidance models from scratch, which can be expensive or infeasible in many cases. In this chapter, we propose a *simplex-based* solution which enables the diffusion over discrete texts while maintaining the advantages of diffusion models with plug-and-control guidance models.

In sum, to enable diffusion-based LMs for text we present SSD-LM (§3.2), addressing the above two challenges. SSD-LM is trained to generate text semi-autoregressively—generating blocks of tokens left-to-right with bidirectional context within the block—which offers the benefits of both AR-LMs and diffusion models. It supports training with and generating variable-length sequences. At the same time, it allows refinement within the token block, in contrast to token-level autoregressive decoding where previously generated tokens cannot be modified at all. SSD-LM uses the same tokenization as popular AR-LMs, representing discrete text via a distribution (or simplex) defined over the vocabulary and is trained to reconstruct texts from noisy versions of the distributions. Due to its underlying representation, our method also offers an easy and

modular way of guided (controlled) generation using off-the-shelf text classifiers under the minimal assumption of shared tokenizer.

Our evaluation experiments show, for the first time, that a diffusion-based LM matches or outperforms strong AR-LMs on standard text generation benchmarks (§3.3). We evaluate SSD-LM on two tasks: (1) unconstrained prompt-based generation substantially outperforming existing diffusion LM approaches and performing on par with or outperforming strong autoregressive LM GPT-2 (Radford et al., 2019) on both quality and diversity (§3.3.2); and (2) controlled text generation with guidance from off-the-shelf classifiers (no post-hoc training/adaptation) outperforming competitive controlled text generation baselines (§3.3.3).

3.1 Background

3.1.1 Diffusion model

Since their inception as image generators, diffusion models (and their cousins score-based models (Song and Ermon, 2019)) have been widely adopted as high-quality generative models for multiple data modalities. Here, we briefly describe a simplified view of a canonical method, denoising diffusion probabilistic models (Ho et al., 2020, DDPM) which we adapt in this work for text generation. We assume a given dataset $\mathcal{D} = \{x_0, \dots, x_N\}$ of continuous valued items x_0 (e.g., pixel values of an image) henceforth referred to as x_0 for simplicity.

Training Training a diffusion model first involves adding a series of Gaussian noise to the original data x_0 , through T timesteps:

$$x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon_t \quad (3.1)$$

where $t \in (1, T)$ and $\epsilon_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. $\bar{\alpha}_t = \prod_{t'=1}^t \alpha_{t'}$, where $\alpha_{t'}$ follow a predefined schedule such that $\bar{\alpha}_t \rightarrow 0$ as $t \rightarrow T$. This process is called *forward diffusion*. A diffusion model (parameterized by θ) is trained to reverse this forward process by predicting the added noise ϵ_t given x_t with the following loss:

$$\mathcal{L}(\theta) = \mathbb{E}_{t \sim \mathcal{U}(1, T)} \|\epsilon_\theta(x_t, t) - \epsilon_t\|^2 \quad (3.2)$$

Inference To get an output from this model, we sample $x_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and iteratively reconstruct a sample x_0 by going back in time,

$$x_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_\theta(x_t, t) \right) \quad (3.3)$$

for $t = T, \dots, 1$.¹ The key obstacle in using vanilla diffusion models directly as text generators is that language consists of discrete tokens, i.e., a non-continuous x_0 to which a continuous valued Gaussian noise cannot be added. We propose a straightforward and effective solution by treating tokens as continuous valued simplexes over the vocabulary (Hoang et al., 2017). Other existing methods addressing this problem are discussed in §5.5.

3.1.2 Autoregressive LM

An autoregressive LM optimizes for the likelihood of a sequence of tokens w^0, \dots, w^{L-1} .

$$p_\theta(\mathbf{w}^{0:L}) = \prod_{c=0}^{L-1} p_\theta(w^c \mid \mathbf{w}^{<c}) \quad (3.4)$$

To decode from AR-LMs, one can provide a context $\mathbf{w}^{<c}$ and decode the next token w^c iteratively by predicting $p_\theta(w^c \mid \mathbf{w}^{<c})$ and sampling from it to get the discrete token (Fan et al., 2018; Holtzman et al., 2020). Prior work has shown that these decoding approaches (and by extension the LMs themselves) are prone to degrade when generating long sequences and often devolve into repeating subsequences (Holtzman et al., 2020; Meister et al., 2023). In addition, such LMs do not provide a natural way to incorporate sequence-level control as tokens are generated one at a time without the ability to modify previously generated tokens (Dathathri et al., 2020; Kumar et al., 2022). In this work, we present a method to train a semi-autoregressive LM that decodes blocks of B tokens at a time, alleviating said issues with the support of diffusion models. Existing literature addressing the two issues individually are discussed in §5.5.

3.2 SSD-LM

We introduce **SSD-LM**—**Semi-autoregressive Simplex-based Diffusion Language Model**—adapting key components from both autoregressive LM and vanilla diffusion models.

¹We omit an additional noise term z here for simplicity, which is present in DDPM but not in another variant DDIM (Song et al., 2021).

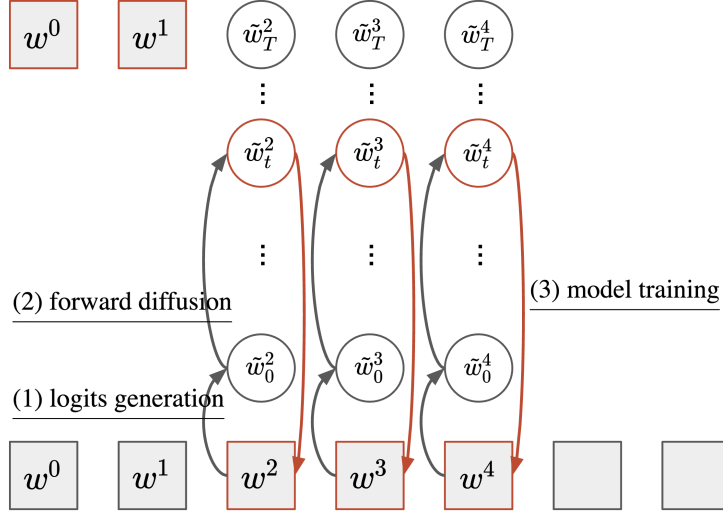


Figure 3.1: Training Ssd-LM (a snapshot at context size $c = 2$, block size $B = 3$). Horizontal axis represents the order of tokens. Vertical axis represents the diffusion timesteps. Shade means observable variables. Square means discrete vocabulary, while circle means continuous logits. Red components are inputs to the learning model θ .

Conceptually, Ssd-LM uses diffusion model to decode $w^{c:c+B}$, a block of tokens of length B , given a Gaussian noise and a context $w^{<c}$ of length c . We show an intuitive diagram and pseudo-code for the training and decoding algorithm of Ssd-LM in Figure 3.1, Figure 3.2, and Figure 3.3.

3.2.1 Training

Continuous data representation To build a continuous representation for discrete tokens, we adopt an *almost-one-hot* simplex representation over the model’s vocabulary V . We define a simple operation logits-generation(.) to map a token w to $\tilde{w} \in \{-K, +K\}^{|V|}$ as follows.

$$\tilde{w}_{(i)} = \begin{cases} +K & \text{when } w = V_{(i)} \\ -K & \text{when } w \neq V_{(i)} \end{cases} \quad (3.5)$$

where i is the index of the vocabulary. We call \tilde{w} the logits for token w , and $\text{softmax}(\tilde{w})$ gives a probability simplex over the vocabulary V , with a probability mass concentrated on the token w . There is no learnable parameter in this mapping.

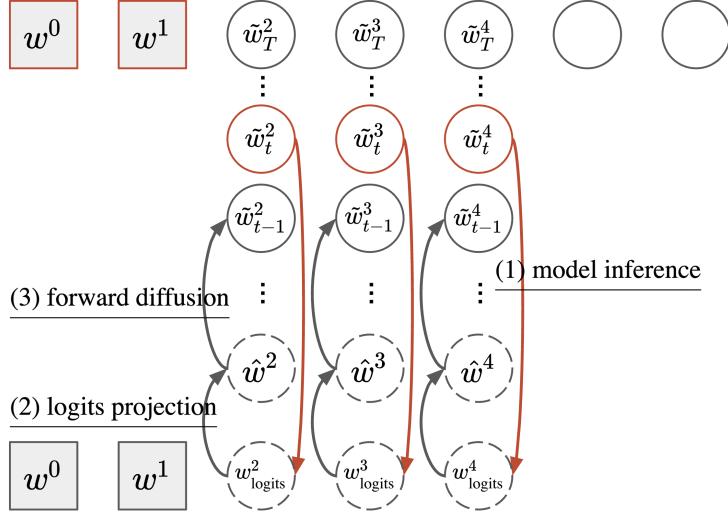


Figure 3.2: Decoding from SSD-LM (continuing Figure 3.1). Red components are inputs to the learned model θ . Dash means intermediate variables.

Forward diffusion Following Ho et al. (2020), we add a time-dependent Gaussian noise to the logits.

$$\tilde{w}_0^{c:c+B} = \text{logits-generation}(w^{c:c+B}) \quad (3.6)$$

$$\tilde{w}_t^{c:c+B} = \sqrt{\bar{\alpha}_t} \tilde{w}_0^{c:c+B} + \sqrt{1 - \bar{\alpha}_t} \epsilon_t \quad (3.7)$$

where $t \in (1, T)$, $\epsilon_t \sim \mathcal{N}(\mathbf{0}, K^2 \mathbf{I})$, and $\bar{\alpha}_t \rightarrow 0$ as $t \rightarrow T$. At the final step T , $\text{softmax}(\tilde{w}_T^{c:c+B})$ are fully noisy simplexes over V , with a logit-normal distribution (Atchison and Shen, 1980).

Loss function In Eq. 3.2, a diffusion model is trained to predict the added noise from the noisy representations. Since the forward diffusion process can be computed in a single step (Eq. 3.1), the notion here is equivalent to predicting the original data representation (Song et al., 2021; Li et al., 2022b). Our objective follows the same intuition but estimates a likelihood instead of the L2 distance while conditioning on additional context:²

$$\mathcal{L}(\theta) = \mathbb{E}[-\log p_\theta(w^{c:c+B} | \tilde{w}_t^{c:c+B}, w^{<c})] \quad (3.8)$$

$$= \mathbb{E} \left[\sum_{j=c}^{c+B-1} -\log p_\theta(w^j | \tilde{w}_t^{c:c+B}, w^{<c}) \right] \quad (3.9)$$

²L2 distance did not work in our pilot study potentially due to the intrinsically skewed simplex representation.

Algorithm 1 Training	Algorithm 2 Decoding (at a given c)
1: repeat 2: $\mathbf{w}^{0:L} \sim q(\mathbf{w}^{0:L})$ 3: $c \sim \text{Uniform}(\{1, \dots, L - B\})$ 4: $\tilde{\mathbf{w}}_0^{c:c+B} = \text{logits-generation}(\mathbf{w}^{c:c+B})$ 5: $t \sim \text{Uniform}(\{1, \dots, T\})$ 6: $\epsilon \sim \mathcal{N}(\mathbf{0}, K^2 \mathbf{I})$ 7: $\tilde{\mathbf{w}}_t^{c:c+B} = \sqrt{\bar{\alpha}_t} \tilde{\mathbf{w}}_0^{c:c+B} + \sqrt{1 - \bar{\alpha}_t} \epsilon$ 8: Take gradient descent step on $\nabla_{\theta} [-\sum_{j=c}^{c+B-1} \log p_{\theta}(w^j \tilde{\mathbf{w}}_t^{c:c+B}, \mathbf{w}^{<c})]$ 9: until converged	1: $\tilde{\mathbf{w}}_T^{c:c+B} \sim \mathcal{N}(\mathbf{0}, K^2 \mathbf{I})$ 2: for $t = T, \dots, 1$ do 3: $\mathbf{w}_{\text{logits}}^{c:c+B} = \text{logits}_{\theta}(\mathbf{w}^{c:c+B} \tilde{\mathbf{w}}_t^{c:c+B}, \mathbf{w}^{<c})$ 4: $\hat{\mathbf{w}}^{c:c+B} = \text{logits-projection}(\mathbf{w}_{\text{logits}}^{c:c+B})$ if uncontrolled, else $\hat{\mathbf{w}}^{c:c+B} =$ logits-projection($\mathbf{w}_{\text{logits}}^{c:c+B} + \lambda \nabla_{\mathbf{w}} f_{\phi}(\cdot)$) 5: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, K^2 \mathbf{I})$ 6: $\tilde{\mathbf{w}}_{t-1}^{c:c+B} = \sqrt{\bar{\alpha}_{t-1}} \hat{\mathbf{w}}^{c:c+B} + \sqrt{1 - \bar{\alpha}_{t-1}} \mathbf{z}$ 7: end for 8: return $\text{argmax} \tilde{\mathbf{w}}_0^{c:c+B}$

Figure 3.3: Training and decoding algorithms for Ssd-LM. The training algorithm starts with sampling a sequence from the pretraining data $q(\mathbf{w}^{0:L})$. The decoding algorithm can be applied m iterations to obtain a $m \cdot B$ -token generation, with the returned B tokens at each iteration appended to the previous generation, increasing c .

$\mathbb{E}[\cdot]$ is a shorthand for $\mathbb{E}_{c \sim \mathcal{U}(1, L-B), t \sim \mathcal{U}(1, T)}[\cdot]$. The architecture for θ throughout this work is a bi-directional Transformer encoder (Vaswani et al., 2017). Specifically, the input to the model is a concatenation of the context $\mathbf{w}^{<c}$ and a sequence of noisy vocabulary simplexes $\text{softmax}(\tilde{\mathbf{w}}_t^{c:c+B})$ of length B . The target output is the original tokens $\mathbf{w}^{c:c+B}$ at positions c to $c + B$.

One minimal modification made to the Transformer model is that in addition to the conventional embedding lookup for $\mathbf{w}^{<c}$, we modify the embedding layer to take as input a distribution over the vocabulary, $\text{softmax}(\tilde{\mathbf{w}}_t^{c:c+B})$, and compute the embedding vector as a weighted sum of the embedding table. A timestep embedding is also added before the first Transformer block to inform the model of the current timestep.³

In §3.5.2, we present another interpretation of the training objective as an intuitive contrastive loss.

³More specifically, we have word embeddings for the context, $\text{Emb}_{\text{ctx}}(\mathbf{w}^{<c})$, and for the noisy diffusion representations, $W_{\text{diff}}[\text{softmax}(\tilde{\mathbf{w}}_t^{c:c+B})]$. The timestep embedding is added to the diffusion word embeddings, $W_{\text{time}}(t/T)$. It is similar to positional embeddings, just not varying across sequence positions. We fold it in θ for notation simplicity.

3.2.2 Decoding

Logits projection Similar to continuous-valued diffusion models, sampling from SSD-LM involves reverse diffusion from $t = T, \dots, 1$ starting with a Gaussian noise. At any timestep t , our model θ takes as input noised logits $\tilde{\mathbf{w}}_t^{c:c+B}$ and estimates the probability distribution of the original tokens in data by first predicting the logits:

$$\mathbf{w}_{\text{logits},t}^{c:c+B} = \text{logits}_{\theta}(\mathbf{w}^{c:c+B} \mid \tilde{\mathbf{w}}_t^{c:c+B}, \mathbf{w}^{<c}) \quad (3.10)$$

which are then converted to a distribution via softmax. To feed this output to the next step of reverse diffusion, $t - 1$, we define a logits-projection operation to build a predicted data representation close to the initial data representation (almost-one-hot mapping; Eq. 3.5). We consider three projection operations.

- *Greedy*: creates an almost-one-hot logit centered at the highest probability token.⁴

$$\hat{\mathbf{w}}_{(i)} = \begin{cases} +K & \text{if } i = \text{argmax}(\mathbf{w}_{\text{logits}}) \\ -K & \text{otherwise} \end{cases} \quad (3.11)$$

- *Sampling*: creates an almost-one-hot logit centered around a token sampled from the output distribution using top- p sampling (Holtzman et al., 2020). p is a hyperparameter.

$$\hat{\mathbf{w}}_{(i)} = \begin{cases} +K & \text{if } i = \text{top-}p\text{-sample}(\mathbf{w}_{\text{logits}}) \\ -K & \text{otherwise} \end{cases} \quad (3.12)$$

- *Multi-hot*: creates an almost-one-hot logit centered around *all* tokens in the top- p nucleus.

$$\hat{\mathbf{w}}_{(i)} = \begin{cases} +K & \text{if } i \in \text{top-}p\text{-all}(\mathbf{w}_{\text{logits}}) \\ -K & \text{otherwise} \end{cases} \quad (3.13)$$

Decoding iteration Starting from pure noise $\tilde{\mathbf{w}}_T^{c:c+B} \sim \mathcal{N}(\mathbf{0}, K^2\mathbf{I})$, in each decoding timestep we compute:

$$\hat{\mathbf{w}}_t^{c:c+B} = \text{logits-projection}(\mathbf{w}_{\text{logits},t}^{c:c+B}) \quad (3.14)$$

$$\tilde{\mathbf{w}}_{t-1}^{c:c+B} = \sqrt{\bar{\alpha}_{t-1}}\hat{\mathbf{w}}_t^{c:c+B} + \sqrt{1 - \bar{\alpha}_{t-1}}\mathbf{z} \quad (3.15)$$

⁴This shares a similar intuition as a greedy clamping trick in the embedding-based diffusion in Li et al. (2022b).

for $t = T, \dots, 1$ and $z \sim \mathcal{N}(\mathbf{0}, K^2 \mathbf{I})$.

At $t = 1$, the final B -token block is computed simply as $\operatorname{argmax} \tilde{w}_0^{c:c+B}$. To generate the next block, we concatenate the generated block to the previous context to create a new context of length $c + B$ and follow the reverse-diffusion process again as described above. This process can be repeated until the maximum desired length is reached.⁵

It is worth noting that our proposed decoding algorithm is novel and different from the DDPM decoding (Eq. 3.3). The DDPM decoding is designed for diffusion in a continuous space and failed to generate sensible outputs in our preliminary experiments based on simplexes. In §3.5.3, we draw a theoretical connection between our decoding algorithm and DDPM decoding, and also highlight the intuitive difference between the two.

Highly-modular control A useful property of continuous diffusion models that naturally arises from their definition is the ability to guide the generated samples to have user-defined attributes at test time. This can be done using gradients from auxiliary models such as classifiers (Dhariwal and Nichol, 2021), e.g., guiding the output of an LM to be of a positive sentiment using a sentiment classifier. There is a vibrant community of developers on platforms such as HuggingFace where many such text classifiers are publicly available. The underlying data representation of Ssd-LM is based on vocabulary simplexes. Hence, as long as a classifier shares the same tokenizer as the LM, it can be used for control in an off-the-shelf manner without modifications. This is in contrast to prior work in diffusion language models that do not support such classifiers due to differences in their input representation space (Hoogeboom et al., 2021; Austin et al., 2021; Li et al., 2022b; Chen et al., 2022b) and require retraining the classifiers from scratch. This ability makes Ssd-LM highly modular for controlled text generation and offers key benefits: (1) Training accurate classifiers for many tasks requires huge amounts of data where retraining them can be quite expensive, and (2) this approach allows control from classifiers that are open to use but have been trained on closed source data.

⁵Alternatively, one can also terminate the process if certain special end-of-sequence tokens have been generated.

To guide SSD-LM to generate texts with a target attribute y via a standalone attribute model $f_\phi(\cdot)$, we update $\mathbf{w}_{\text{logits},t}^{c:c+B}$ (Eq. 3.10) at each timestep t to the form below, drifting according to the gradients from the attribute classifier.

$$\mathbf{w}_{\text{logits},t}^{c:c+B} + \lambda \nabla_{\mathbf{w}_{\text{logits},t}^{c:c+B}} f_\phi(y | \mathbf{w}_{\text{logits},t}^{c:c+B}, \mathbf{w}^{<c}) \quad (3.16)$$

where λ is a hyperparameter balancing the weight of control. The parameters of the standalone attribute model ϕ are frozen. We make a trivial modification to the embedding computation as in §3.2.1, to allow the classifier to take as input a simplex.

3.2.3 Additional details

Forward diffusion coefficient $\bar{\alpha}_t$ We follow Nichol and Dhariwal (2021) for a cosine schedule of $\bar{\alpha}_t$:

$$\bar{\alpha}_t = \frac{r(t)}{r(0)}, \quad r(t) = \cos\left(\frac{t/T + s}{1 + s} \cdot \frac{\pi}{2}\right)^2 \quad (3.17)$$

where s is small offset set to 1e-4 in our work and $\alpha_t = \frac{\bar{\alpha}_t}{\bar{\alpha}_{t-1}}$.

Fewer timesteps T in decoding Decoding from diffusion models requires a series of timesteps (T) which can be computationally expensive if T is large. Following Li et al. (2022b), we consider using a smaller value of T at test time to improve decoding speed. In this work, we primarily experiment with $T_{\text{decode}} = \frac{T_{\text{train}}}{2}$ and $T_{\text{decode}} = \frac{T_{\text{train}}}{5}$.

Flexible decoding block size B Our SSD-LM is trained with a fixed token block size B_{train} . However, the decoding algorithm has a freedom to use a different B_{decode} . In our experiments, we consider both scenarios of $B_{\text{train}} = B_{\text{decode}}$ and $B_{\text{train}} \neq B_{\text{decode}}$. Nevertheless, we leave for future work a more detailed analysis of the impact of the difference between B_{train} and B_{decode} on model performance.

3.3 Experiments

3.3.1 SSD-LM pretraining setup

Model architecture We use a bidirectional Transformer encoder RoBERTa-large (Liu et al., 2019) (0.4B, comparable size to GPT2-medium) as SSD-LM’s underlying archi-

ecture.⁶ Note that RoBERTa uses a general BPE tokenization (Sennrich et al., 2016), same as a variety of LMs such as GPT-2 (Radford et al., 2019), GPT-3 (Brown et al., 2020), OPT (Zhang et al., 2022), etc. Any attribute classifier using the same tokenization strategy can be used to control Ssd-LM in a highly modular way.

Pretraining data, constants, and resource We train Ssd-LM on the same data as GPT2 to make fair comparisons possible: OpenWebText (Gokaslan and Cohen, 2019) which contains 9B tokens. Following Zhang et al. (2022), we consider this data as one contiguous sequence of tokens and break it into sequences of length 200 (same as the maximum sequence length our model accepts). We randomly sample 99% of these sequences for pretraining while leaving the rest as held out for evaluation. We use the following model hyperparameters:⁷

$$L = 200, B_{\text{train}} = 25, T_{\text{train}} = 5000, K = 5$$

We use an aggregated batch size of 6,144 and a learning rate of 1e-4 with an AdamW optimizer (Loshchilov and Hutter, 2019). We trained Ssd-LM for 100K steps, which took about 6 days on 32 Nvidia V100 GPUs.

Pretraining loss Canonical training-time perplexity of LMs is not compatible with diffusion LMs due to the difference in the inputs to the models (Eq. 3.4 and Eq. 3.9). Our pretraining loss is a per-token negative log-likelihood (NLL) that depends on the specific noise schedule being used. Ssd-LM gets an average NLL of 3.87 at the end of pretraining. We show a pretraining loss curve in §3.5.5.

3.3.2 Unconstrained text generation

Setup First, we benchmark Ssd-LM with autoregressive LMs trained on the same data (GPT2) on text generation quality. We randomly sample 1000 sequences from the held-out OpenWebText test data, extract their prefixes as prompts (context), and

⁶We initialize the model with RoBERTa’s weights as well. We observe in our initial exploration that it helps the training loss converge faster than a randomly initialized model. However, given enough computational resources, we conjecture that a randomly initialized model will offer similar performance.

⁷Future work can do a search given more resources.

⁸MAUVE, Dist-1/2/3, and Rep are in percentage. PPL is obtained through a micro average following Holtzman et al. (2020); Pillutla et al. (2021); Meister et al. (2023).

(Length 50)	MAUVE	PPL	$ \Delta_{\log\text{PPL}} $	Dist-	Dist-	Dist-	Zipf	Rep
	↑	$\xrightarrow{\text{gold}}$	↓	1 ↑	2 ↑	3 ↑	$\xrightarrow{\text{gold}}$	↓
<i>Gold continuation</i>	100.00	17.75	0.00	88.62	95.88	93.71	0.88	0.10
<u>GPT2-medium</u> (Best config)								
Top- $p=0.95$	96.57 ± 0.40	12.72 ± 0.07	0.33	66.31 ± 0.11	91.77 ± 0.03	92.75 ± 0.06	1.01	0.26 ± 0.04
<u>GPT2-large</u> (Best config)								
Top- $p=0.95$	96.41 ± 0.78	10.57 ± 0.05	0.51	64.91 ± 0.13	90.88 ± 0.06	92.38 ± 0.05	1.01	0.41 ± 0.06
<u>GPT2-xl</u> (Best config)								
Typical- $\tau=0.95$	97.03 ± 0.50	10.33 ± 0.04	0.54	64.87 ± 0.15	90.69 ± 0.07	92.16 ± 0.05	1.01	0.37 ± 0.04
<u>Ssd-LM-“medium”</u> (Top-3)								
Sampling $p=0.99, T=1000$	97.89	30.68	0.54	68.99	92.60	92.94	1.01	0.16
Sampling $p=0.95, T=1000$	96.64	27.34	0.43	67.75	92.16	92.91	1.01	0.16
Sampling $p=0.9, T=2500$	96.46	20.56	0.14	66.61	91.46	92.56	1.05	0.26

Table 3.1: Unconstrained generation evaluation of Ssd-LM and GPT-2 models at length 50. For GPT-2 models, the results are averaged across 5 random seeds, and we show the best sampling parameter configuration. For our Ssd-LM, we show the top-3 configurations. All configurations are ranked based on MAUVE, with original parameters from Pillutla et al. (2021). The perplexity (PPL) is measured by GPT-Neo-1.3B.⁸

generate continuations from the LMs. We consider three setups: with prompt lengths 25, 50 and 100 with respective output lengths as 25, 50 and 100 tokens. In each setup, we sample 5 continuations for each input context, thus comparing the quality of 5,000 generations from baseline GPT-2 models and our Ssd-LM.

We compare Ssd-LM with GPT2-medium, large and xl models (containing 0.4B, 0.8B and 1.6B parameters respectively) as baselines. For reference, our model size is comparable to GPT2-medium. We experiment with two popular decoding strategies for the baseline GPT-2 models with canonical parameters: nucleus sampling (Holtzman et al., 2020) with a top- p of 0.9 and 0.95, and typical sampling (Meister et al., 2023) with a typical- τ of 0.2 and 0.95.

For Ssd-LM, we consider three logits projection strategies, sampling and multi-hot with $\text{top-}p \in \{0.0, 0.1, 0.2, 0.5, 0.7, 0.9, 0.95, 0.99\}$, and greedy (which is functionally equivalent to the sampling with $\text{top-}p=0$). We use a test block size (B_{decode}) of 25. When generating samples of length 50 or 100, we semi-autoregressively sample in blocks of 25 and feed them as additional context to generate the next block as described in §3.2.2.

We evaluate the generated continuations on two axes: quality and diversity. As automatic quality metrics, we report perplexity measured by a separate, larger language model (GPT-Neo-1.3B, Black et al., 2021). Prior works, however, have shown that low perplexity of generated text is not necessarily an indication of high quality but of degenerate behavior (Nadeem et al., 2020; Zhang et al., 2021) and have proposed closeness to the perplexity of human-written text as a better evaluation. Hence, we also report the difference of log perplexity between the generated text and human-written continuations ($|\Delta_{\log \text{PPL}}|$). For diversity evaluation, we report Zipf’s coefficient (Zipf) and average distinct n -grams in the output samples (Li et al., 2016, Dist- n). In addition, we also report the repetition rate (Welleck et al., 2020; Holtzman et al., 2020, Rep), measuring the proportion of output samples that end in repeating phrases. Finally, we report MAUVE (Pillutla et al., 2021) which evaluates both quality and diversity together by approximating information divergence between generated samples and human-written continuations (from the OpenWebText held-out set).

Results Table 3.1 summarizes our main results on the 50-token prompt and output setup. We report the numbers for the best performing three settings for logits projection and decoding steps T in Ssd-LM. We report the best setting for the baselines. The results for other generation lengths have a similar trend and can be found in §3.5.5.

We find that Ssd-LM, though being smaller in size, outperforms larger GPT-2 models on the unified metric MAUVE. On diversity, Ssd-LM outperforms GPT-2 in Dist- n while achieving lower repetition rates. On perplexity, the results are slightly mixed. We observe a trade-off between MAUVE and perplexity for different settings we considered, indicating that further tuning of the hyperparameters may be required. However, one of our best performing settings (sampling $\text{top-}p=0.9$, $T=2500$) still achieves the closest perplexity to the gold continuation.

(ROCStories)	MAUVE	PPL
<i>Gold continuation</i>	100.00	18.57
<u>Diffusion-LM</u>	46.11	35.96
<u>Ssd-LM</u>	87.22	22.91

Table 3.2: Unconstrained generation results of Ssd-LM and Diffusion-LM on ROCStories with 50 prompt tokens and 50 output tokens. We report the MAUVE score between the gold continuation and model generations. We also show the perplexity (PPL) of model generations measured by GPT-Neo-1.3B.⁹

In §3.5.5, we show the influence of different logits projection strategies and the associated parameters on the output text quality in Figure 3.4. We also show qualitative examples of the generations by Ssd-LM in Table 3.8 and a trajectory of intermediate states during the decoding process in Table 3.9.

Comparison with Li et al. (2022b) A prior work to us, Li et al. (2022b) propose Diffusion-LM, an embedding-based diffusion model trained on two small toy datasets, E2E (Novikova et al., 2017) and ROCStories (Mostafazadeh et al., 2016). In this subsection, we make a diversion to compare the embedding-based Diffusion-LM with our semi-autoregressive, simplex-based Ssd-LM. Following Li et al. (2022b), we train a Diffusion-LM on ROCStories with a default embedding size of 128, 0.1B parameters under a BERT-base (Devlin et al., 2019) structure,¹⁰ and a sequence length of 100. For a fair comparison, *only within this subsection* we train a Ssd-LM with ROCStories sequences of 100 tokens, a decoding block size of 25, and a BERT-base initialization. Further details of the setup can be found in §3.5.4.

On 2,700 held-out ROCStories sequences, we use the first 50 tokens of each sequence as a prompt and have the model generate the next 50. In Table 3.2, we show the MAUVE

⁹Due to a lowercase tokenization of ROCStories, we use BERT-base-uncased as MAUVE’s embedding model here.

¹⁰We train two versions of Diffusion-LM, with and without BERT’s encoder weights as an initialization. The default no-initialization setup as in Li et al. (2022b) works reasonably, while the other degenerates. Details can be found in §3.5.4.

score and perplexity of both models. We observe a substantially higher MAUVE score and lower perplexity with Ssd-LM.

3.3.3 Controlled text generation

Setup To evaluate Ssd-LM’s ability for highly-modular control, we consider the task of sentiment controlled generation where given a prompt, the goal is to generate a continuation with a positive (or negative) polarity. We use a set of 15 short prompts as in Dathathri et al. (2020) and generate 20 samples per prompt per sentiment category, making the total number of generated samples to be 600. Following Mireshghallah et al. (2022), we generate samples with 3 different output lengths: 12, 20 and 50. For guidance, we simply import a popular sentiment classifier¹¹ from HuggingFace trained with Twitter sentiment data with over 58M training examples (Barbieri et al., 2020). This model serves as $f_\phi(\cdot)$ as shown in Eq. 3.16. In addition to quality and diversity of the generated samples, we also evaluate them on control (that is measuring if the generated output is actually positive or negative in polarity). For this, we use an *external* sentiment classifier trained on a different dataset. Specifically, we use a classifier trained with Yelp reviews¹² (Zhang et al., 2015; Morris et al., 2020) following the evaluation setup in the baselines we consider.

Again, we consider the sampling and multi-hot decoding strategies with $\text{top-}p \in \{0.2, 0.5, 0.9\}$, $T_{\text{decode}} \in \{1000, 2500, 5000\}$, and the multiplier for control $\lambda \in \{0, 100, 500, 2000\}$. For the generation of 12/20/50 tokens, we use $B_{\text{decode}}=12/20/25$ and apply the decoding algorithm for $m=1/1/2$ iterations respectively.

Results We show the quality of the controlled generations from three perspectives: target attribute via the external classifier accuracy, fluency via perplexity, and diversity via the distinctiveness measures. In Table 3.3, we show the experimental results for output length 50. The results at length 12 and 20 have a similar trend and can be found in §3.5.5.

Among the baseline methods, DAPT (Gururangan et al., 2020), GeDi (Krause et al., 2021), and DExperts (Liu et al., 2021) require training customized language models aware of the desired attributes (denoted as CM in Table 3.7). PPLM (Dathathri et al.,

¹¹<https://huggingface.co/cardiffnlp/twitter-roberta-base-sentiment>

¹²<https://huggingface.co/textattack/bert-base-uncased-yelp-polarity>

(Length 50)	C-Ext. _(Int.)	PPL	Dist-1/2/3
<u>DAPT</u> ^{CM}	79.8	57.2	61/92/94
<u>PPLM</u> ^{CC}	60.7 <small>(73.6)</small>	29.0	-
<u>FUDGE</u> ^{CC}	59.1	8.4	47/83/92
<u>GeDi</u> ^{CM}	99.2	107.3	71/93/92
<u>DExperts</u> ^{CM}	94.8	37.1	56/90/92
<u>MuCoLa</u> ^{CC}	86.0	27.8	52/76/80
<u>M&M LM</u> ^{HMC}	68.6 <small>(93.8)</small>	122.3	-
<u>Ssd-LM</u> ^{HMC}	<i>94.1</i> <small>(99.0)</small>	23.1	46/84/92

Table 3.3: Controlled text generation results of Ssd-LM and baselines at length 50. We report the external classifier’s accuracy (C-Ext.) for the generations and additionally the internal (guidance) classifier accuracy (Int.) if available. The perplexity (PPL) is computed with GPT2-xl. MuCoLa is the version using two discriminators. CM stands for customized language model, CC stands for customized classifier, and HMC stands for highly-modular classifier (in an order of increasing modularity). The best of all results are boldfaced, and the best of HMC results are italicized.¹³

2020), FUDGE (Yang and Klein, 2021), and MuCoLa (Kumar et al., 2022) require training a customized attribute classifier (CC). While our proposed method Ssd-LM and M&M LM (Miresghallah et al., 2022) can directly import mainstream existing attribute classifiers from platforms like HuggingFace and are thus highly modular (HMC). We show the baseline results as reported in Miresghallah et al. (2022) and Kumar et al. (2022).

Ssd-LM shows strong controllability while possessing great modularity. Ssd-LM outperforms M&M LM, the other HMC method by a large margin. Even when comparing with the CC and CM methods, our method achieves a good balance in control, fluency, and diversity.

¹³PPL is obtained through a macro average following Kumar et al. (2022).

In §3.5.5, we show the impact of the control weight λ and top- p on the attribute accuracy and perplexity in Figure 3.5. We also show qualitative examples of the controlled generations by Ssd-LM in Table 3.8.

3.4 Related work

Diffusion models Diffusion models have demonstrated impressive performance in popular continuous-valued domains such as images (Ho et al., 2020), audio (Kong et al., 2021), video (Ho et al., 2022) and recently also been adopted for 3D-shapes, protein structures, and more (Zhou et al., 2021; Trippe et al., 2022; Wu et al., 2022). Since they are based on adding Gaussian noise, these approaches are not straightforward to apply to discrete valued domains like text. Hoogeboom et al. (2021); Austin et al. (2021) propose diffusing in the discrete space using categorical distributions which are modified using transition matrices. However, these methods do not straightforwardly support control and yield worse results than comparable autoregressive models. Li et al. (2022b) propose to represent each token as a continuous embedding and apply diffusion in the embedding space. They train the LM to generate a fixed length sequence whereas Ssd-LM allows flexibility in the generated sequence length by generating block-wise. Further, their LM is trained with specialized datasets and not evaluated against general-purpose autoregressive LMs on unconstrained text generation. Their method supports post-hoc control but requires training a customized attribute classifier,¹⁴ since the diffusion operates on a learned embedding space. Gong et al. (2022), a concurrent work to ours, extend Li et al. (2022b) to a sequence-to-sequence setup with a similar underlying embedding-based method. Our work is most closely related to Chen et al. (2022b) which transform discrete data into a sequence of bits and represent each bit as +1 or -1 converting it into a continuous-valued domain. For textual data, however, it can lead to extremely long sequences which are difficult to optimize. In this work, we instead maintain a subword based vocabulary but represent each token as a sequence of manually defined logits.

¹⁴The control for diffusion models can also be classifier-free (Ho and Salimans, 2021) but requires training with the target attribute in advance, which is not a focus of this work.

Language models The majority of existing language models for text generation are trained autoregressively, i.e., they predict the next token given previously generated context. This paradigm scaled up both in terms of model size and training data size has resulted in impressive capabilities on many benchmarks (Brown et al., 2020; Chowdhery et al., 2022). However, they generate text one token at a time which does not provide flexible control over attributes of the generated text. Non-autoregressive models which generate the entire output sequence at the same time have also been explored in prior work other than diffusion models (Lee et al., 2018; Ghazvininejad et al., 2019). However, they are primarily focused on improving decoding efficiency and applied for specialized tasks like translation (Gu et al., 2018; Kaiser et al., 2018; Wang et al., 2019) and text editing (Gu et al., 2019). Many of these work have iterative processes in a discrete space, with some exploring continuous representations (Ma et al., 2019; Lee et al., 2020). To address the quality decline with the non-autoregressive methods compared to autoregressive models, prior work have also explored semi-autoregressive approaches (Wang et al., 2018; Qi et al., 2021). In the same vein, our work seeks to address the drawbacks of autoregressive language models and non-autoregressive diffusion models with a middle ground.

Controllable text generation Early solutions for controlling attributes of generated text focused on training or finetuning AR-LMs with specific control codes (Keskar et al., 2019; Gururangan et al., 2020; Chan et al., 2021). These methods are difficult to extend to new controls as it requires retraining the models. More recent work includes decoding approaches from pretrained AR-LMs without modifying the models, through altering the output probability distribution at each step using different control objectives (Dathathri et al., 2020; Krause et al., 2021; Yang and Klein, 2021; Liu et al., 2021; Lu et al., 2021; Pascual et al., 2021). However, these methods do not allow modifying a token once it is generated and are thus suboptimal for controls at the scope of the whole sequence. Closely related to Ssd-LM are Kumar et al. (2021); Qin et al. (2022); Kumar et al. (2022), which propose gradient-based decoding algorithms from AR-LMs. They require computing a backward pass through the LMs for each iteration, an expensive operation. In contrast, Ssd-LM with its semi-autoregressive setup allows editing past tokens via diffusion. In addition, most of these approaches require training

control functions from scratch whereas our model allows using off-the-shelf classifiers. [Mireshghallah et al. \(2022\)](#) propose a non-autoregressive LM based on Metropolis-Hastings sampling. It also supports off-the-shelf classifiers for control, and we therefore use it as a direct baseline for SSD-LM.

3.5 Conclusion and continued discussion

In this chapter, we present SSD-LM, a semi-autoregressive diffusion based language model trained to denoise corrupted simplexes over the output vocabulary. Compared to prior work in text-based diffusion, SSD-LM offers more flexibility in output length by generating blocks of text and an ability to use off-the-shelf attribute classifiers for control without additional tuning. On unconstrained text generation, SSD-LM performs on par with or outperforms strong and larger autoregressive baselines (GPT-2) in generation quality and diversity, while vastly outperforming diffusion baselines (Diffusion-LM). Notably, on controlled text generation, SSD-LM surpasses baselines while possessing an easy-to-use modular design — a collaboration between diffusion language model and classifiers. SSD-LM opens an exciting direction for future research in flexible and modular diffusion-based language generation.

3.5.1 Compute limitations

Sample efficiency In AR-LMs, an NLL loss is computed at training time for every token in the sequence of length L (Eq. 3.4). However, in SSD-LM, each time a pretraining example is sampled, the loss is computed on only B tokens (Eq. 3.9) leading to a lower sample efficiency than AR-LM. Towards improving this efficiency, future work could explore model architectures dedicated to semi-autoregressive diffusion rather than the vanilla Transformer encoder we use in this work.

Decoding speed Since each block is generated by refining over several iterations, SSD-LM has a considerably slower decoding speed than autoregressive models. For example, given a context of 50 tokens (single instance, unbatched), it takes SSD-LM 25 seconds to generate the next block of 25 tokens ($T_{\text{decode}}=1000$). While our work focused on establishing the efficacy of diffusion-based LMs and modular controlled

generation, future work could explore tuning T_{decode} to balance model performance and decoding speed, or more efficient training and decoding algorithms extending ideas from prior work on diffusion models for continuous domains (Song et al., 2021; Nichol and Dhariwal, 2021; Rombach et al., 2022; Meng et al., 2022).

Decoding block size In this work, although we allow setups where $B_{\text{train}} \neq B_{\text{decode}}$, the decoding block size B_{decode} remains the same across m decoding iterations, leaving space for a more flexible decoding schedule. Future work can also explore learning B_{decode} (and B_{train}) rather than using constant pre-defined lengths.

Larger scale experiments with different kinds of controls and their combinations can be done, as well as more sophisticated ways to incorporate them (Kumar et al., 2021). In addition, we plan to explore alternative methods to continuously represent and add noise to discrete text (Bakosi and Ristorcelli, 2013). This work experiments with pretraining data that is primarily in English. Future work can also explore challenges and benefits of diffusion-based LMs in a multilingual setup.

3.5.2 A contrastive interpretation of the training loss

The training of SSD-LM is simply maximizing the likelihood $\log p_{\theta}(\mathbf{w}^{c:c+B} | \tilde{\mathbf{w}}_t^{c:c+B}, \mathbf{w}^{<c})$. This diverts from the exact objective of DDPM that is supported by a variational bound. However, below we give an intuitive interpretation to our objective.

$$\log p_{\theta}(\mathbf{w}^{c:c+B} | \tilde{\mathbf{w}}_t^{c:c+B}, \mathbf{w}^{<c}) \quad (3.18)$$

$$= \log \frac{p_{\theta}(\mathbf{w}^{c:c+B} | \mathbf{w}^{<c}) p_{\theta}(\tilde{\mathbf{w}}_t^{c:c+B} | \mathbf{w}^{c:c+B}, \mathbf{w}^{<c})}{p_{\theta}(\tilde{\mathbf{w}}_t^{c:c+B} | \mathbf{w}^{<c})} \quad (3.19)$$

$$= \underbrace{\log p_{\theta}(\mathbf{w}^{c:c+B} | \mathbf{w}^{<c})}_{\text{likelihood of true data}} - \underbrace{\log p_{\theta}(\tilde{\mathbf{w}}_t^{c:c+B} | \mathbf{w}^{<c})}_{\text{likelihood of noisy data at timestep } t} \\ + \underbrace{\log p(\tilde{\mathbf{w}}_t^{c:c+B} | \mathbf{w}^{c:c+B})}_{\text{forward diffusion process independent of } \theta} \quad (3.20)$$

Optimizing θ is a contrastive objective: maximizing the estimated likelihood of true data, while penalizing the estimated likelihood of noisy data under a broad range of different noise scales.

3.5.3 Connection between our decoding algorithm and the DDPM decoding

We revisit the decoding step in DDPM introduced in Eq. 3.3. Since we know that during the training phase x_t is generated through a one-step forward diffusion process (Eq. 3.1), a model θ predicting the added noise $\epsilon_\theta(x_t, t)$ can therefore be considered as predicting an imaginary x_0 in one-step:

$$\hat{x}_0(x_t, t, \theta) = \frac{1}{\sqrt{\bar{\alpha}_t}}(x_t - \sqrt{1 - \bar{\alpha}_t}\epsilon_\theta(x_t, t)) \quad (3.21)$$

Below we write $\hat{x}_0(x_t, t, \theta)$ as \hat{x}_0 and $\epsilon_\theta(x_t, t)$ as ϵ_θ for simplicity.

Rearranging the DDPM decoding transition (Eq. 3.3), we have:

$$x_{t-1} = \sqrt{\bar{\alpha}_{t-1}}\hat{x}_0 + \sqrt{\frac{\alpha_t - \bar{\alpha}_t}{1 - \bar{\alpha}_t}}\sqrt{1 - \bar{\alpha}_{t-1}}\epsilon_\theta \quad (3.22)$$

$$\approx \sqrt{\bar{\alpha}_{t-1}}\hat{x}_0 + \sqrt{1 - \bar{\alpha}_{t-1}}\epsilon_\theta \quad (3.23)$$

with $\sqrt{\frac{\alpha_t - \bar{\alpha}_t}{1 - \bar{\alpha}_t}} \approx 1$ for most $t \in (1, T)$.¹⁵

Noting the format similarity between Eq. 3.1 and Eq. 3.23, we therefore interpret the DDPM decoding transition from x_t to x_{t-1} as (1) predicting an imaginary \hat{x}_0 , and (2) applying a *compensating* forward diffusion step with a deterministic noise ϵ_θ .

Our decoding strategy in Eq. 3.15 is in a very similar form as Eq. 3.23. We also predict the initial data representation with θ and apply a forward diffusion step. The difference is that we sample a noise z instead of using the deterministic ϵ_θ , to encourage exploration.

3.5.4 Detailed setup of the comparison with Diffusion-LM

To compare with Diffusion-LM (Li et al., 2022b), we apply block concatenation on ROCStories similarly as OpenWebText, resulting in 50K training sequences of 100 tokens. We train Diffusion-LM with a default batch size of 64, learning rate of 1e-4, and 400K steps. We train Ssd-LM with a batch size of 512, learning rate of 1e-4, and 20K steps. Both models use a tokenizer of BERT-base-uncased. For Ssd-LM, additional hyperparameters like decoding block size and one-hot constant remain the same as the main Ssd-LM benchmarked with GPT-2. For Diffusion-LM, the evaluation in the main

¹⁵Specifically, we adopt a cosine schedule for $\bar{\alpha}_t$ (Nichol and Dhariwal, 2021), and $\sqrt{\frac{\alpha_t - \bar{\alpha}_t}{1 - \bar{\alpha}_t}} > 0.98$ for 98% of all t , with some outliers as $t \rightarrow 0$ and $t \rightarrow T$.

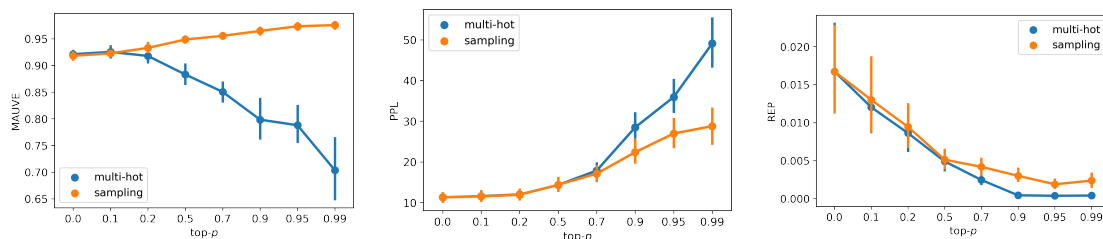


Figure 3.4: Influence of different decoding logits projection strategies and associating top- p for Ssd-LM on various text quality metrics. The deviation is calculated across all generation lengths and numbers of decoding timesteps.

paper is an infilling task. We use same decoding hyperparameters as Li et al. (2022b). For Ssd-LM, the evaluation is a block-wise generation problem with $m=2$ iterations. The result of Ssd-LM in Table 3.2 is obtained with a decoding configuration of $T_{\text{decode}}=2500$ and top- $p=0.5$.

Our Ssd-LM in this subsection is initialized with BERT. For a fair comparison, apart from the default Diffusion-LM reported in Table 3.2, we train another Diffusion-LM initialized with the encoder weights of BERT. However, this leads to degenerated results that are much worse than the default Diffusion-LM and our Ssd-LM: a MAUVE score of 0.4 out of 100 and a PPL of 73157. This problem is not due to overfitting, as all checkpoints of the model show the same degenerated result. Since Li et al. (2022b) did not explore this setup in their original work as well, we conjecture that Diffusion-LM may be incompatible with pretrained weights from existing non-diffusion models by nature, a disadvantage to our Ssd-LM.

3.5.5 Additional results

Figure 3.4 shows the influence of different logits projection strategies and the associated parameters on the unconstrained generations’ output text quality. We observe that reducing top- $p \rightarrow 0$ (greedy projection) can lead to a low perplexity but it is undesirable due to a high repetition rate. We also find the multi-hot projection strategy is overall worse performing than the sampling projection strategy in our setup, indicating it is better to commit the intermediate states to single rather than multiple tokens. This can be because our logits mapping involves putting probability mass on singular tokens.

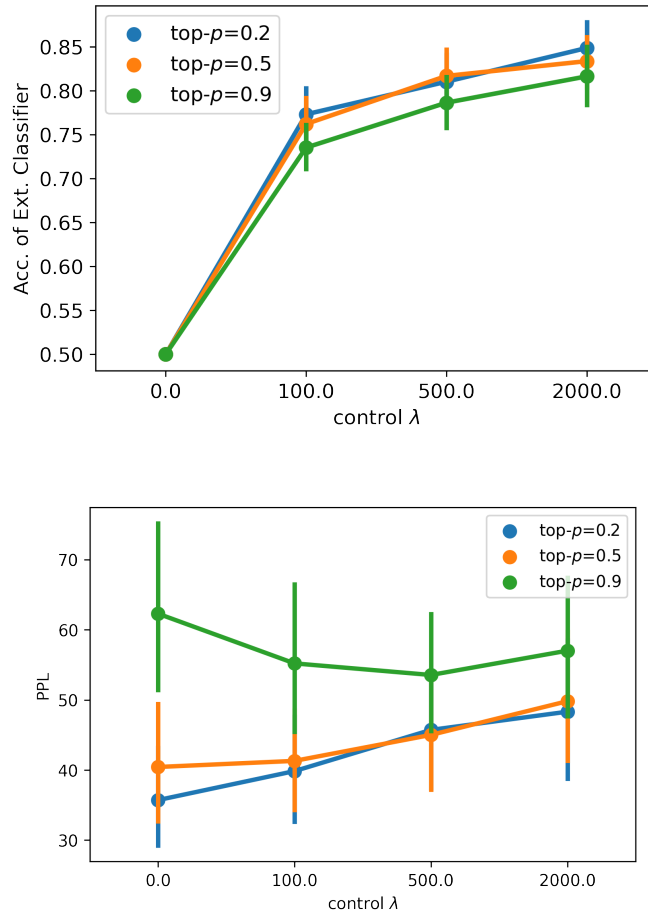


Figure 3.5: Influence of different control weight λ and different top- p . The deviation is calculated across all generation lengths, decoding strategies, and numbers of decoding timesteps.

The multi-hot projection may still be a viable strategy if future work uses multi-hot logits mapping for the input tokens.

Figure 3.5 shows the impact of the control weight λ and top- p on the attribute accuracy and perplexity in controlled text generation. As expected, a larger control weight leads to a better external classifier accuracy. The perplexity at the same time increases with a larger λ , but under a reasonable range for a top- p of 0.2 and 0.5.

Figure 5.3 shows the pretraining loss trajectory. Table 3.4, Table 3.5, Table 3.6, and Table 3.7 show additional evaluation results of Ssd-LM generations. Table 3.8 and Table 3.9 show qualitative examples of Ssd-LM generations.

(Length 25)	MAUVE	PPL	$ \Delta_{\log \text{PPL}} $	Dist-1	Dist-2	Dist-3	Zipf	Rep
	↑	$\xrightarrow{\text{gold}}$	↓	1 ↑	2 ↑	3 ↑	$\xrightarrow{\text{gold}}$	↓
<i>Gold continuation</i>	100.00	21.24	0.00	93.93	93.54	88.23	0.84	0.10
<u>GPT2-medium</u> (Best config)								
Top- $p=0.95$	97.35 \pm 0.29	14.31 \pm 0.07	0.39	73.63 \pm 0.11	90.44 \pm 0.13	87.75 \pm 0.13	1.01	0.21 \pm 0.05
<u>GPT2-large</u> (Best config)								
Top- $p=0.95$	97.01 \pm 0.56	12.14 \pm 0.06	0.55	71.94 \pm 0.10	89.84 \pm 0.06	87.66 \pm 0.06	1.02	0.23 \pm 0.08
<u>GPT2-xl</u> (Best config)								
Top- $p=0.95$	97.29 \pm 0.80	11.90 \pm 0.09	0.57	72.02 \pm 0.04	89.58 \pm 0.14	87.39 \pm 0.13	1.00	0.22 \pm 0.02
<u>Ssd-LM-“medium”</u> (Top-3)								
Sampling $p=0.99, T=1000$	98.41	38.30	0.58	75.61	90.85	87.58	0.98	0.10
Sampling $p=0.99, T=2500$	98.33	30.89	0.37	75.04	90.64	87.54	1.02	0.18
Sampling $p=0.95, T=1000$	98.18	33.79	0.46	74.70	90.67	87.62	0.99	0.18

Table 3.4: Unconstrained generation evaluation of Ssd-LM and GPT-2 models at length 25. PPL is computed with GPT-Neo-1.3B (Black et al., 2021). For GPT-2 models, the results are averaged across 5 random seeds, and we show the best sampling parameter configuration. For our Ssd-LM, we show the top-3 configurations. All configurations are ranked based on MAUVE, with original parameters from Pillutla et al. (2021).

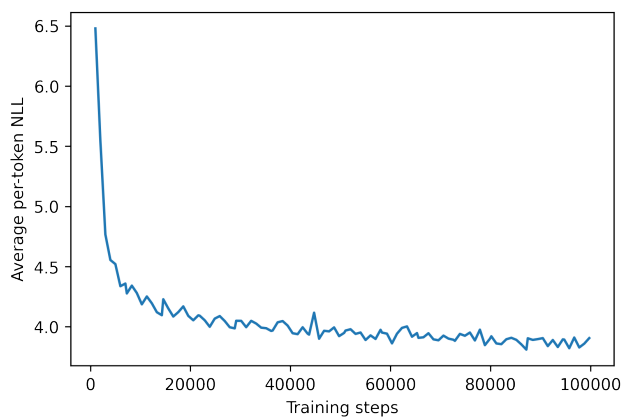


Figure 3.6: Per-token negative log-likelihood during Ssd-LM’s pretraining.

(Length 100)	MAUVE	PPL	$ \Delta_{\log \text{PPL}} $	Dist-1	Dist-2	Dist-3	Zipf	Rep
	\uparrow	$\xrightarrow{\text{gold}}$	\downarrow	1 \uparrow	2 \uparrow	3 \uparrow	$\xrightarrow{\text{gold}}$	\downarrow
<i>Gold continuation</i>	100.00	14.83	0.00	81.40	96.21	96.12	0.90	0.20
<u>GPT2-medium</u> (Best config)								
Top- $p=0.95$	97.54 \pm 0.43	11.68 \pm 0.03	0.23	58.48 \pm 0.02	90.82 \pm 0.04	94.56 \pm 0.03	1.01	0.50 \pm 0.10
<u>GPT2-large</u> (Best config)								
Top- $p=0.95$	97.36 \pm 0.22	9.43 \pm 0.03	0.45	56.96 \pm 0.11	89.43 \pm 0.10	93.96 \pm 0.09	1.02	0.60 \pm 0.06
<u>GPT2-xl</u> (Best config)								
Top- $p=0.95$	97.53 \pm 0.34	9.17 \pm 0.04	0.48	57.10 \pm 0.11	89.35 \pm 0.09	93.76 \pm 0.08	1.00	0.58 \pm 0.06
<u>Ssd-LM-“medium”</u> (Top-3)								
Sampling $p=0.95, T=1000$	97.67	23.38	0.45	60.17	91.30	94.89	1.02	0.30
Sampling $p=0.99, T=2500$	97.36	21.17	0.35	60.02	90.93	94.52	1.04	0.44
Sampling $p=0.99, T=1000$	97.10	26.41	0.57	61.26	91.91	95.11	1.01	0.32

Table 3.5: Unconstrained generation evaluation of Ssd-LM and GPT-2 models at length 100. PPL is computed with GPT-Neo-1.3B (Black et al., 2021). For GPT-2 models, the results are averaged across 5 random seeds, and we show the best sampling parameter configuration. For our Ssd-LM, we show the top-3 configurations. All configurations are ranked based on MAUVE, with original parameters from Pillutla et al. (2021).

(Length 12)	C-Ext. _(Int.)	PPL	Dist-1/2/3
<u>DAPT</u> ^{CM}	66.7	106.5	65/85/79
<u>PPLM</u> ^{CC}	58.0 <small>(71.7)</small>	113.1	-
<u>FUDGE</u> ^{CC}	62.6	12.5	52/76/77
<u>GeDi</u> ^{CM}	93.6	460.6	65/76/69
<u>DExperts</u> ^{CM}	87.4	69.0	65/85/80
<u>MuCoLa</u> ^{CC}	89.0	38.7	49/72/73
<u>M&M LM</u> ^{HMC}	65.1 <small>(94.3)</small>	264.1	-
<u>SsD-LM</u> ^{HMC}	79.3 <small>(90.5)</small>	58.1	60/83/80

Table 3.6: Controlled text generation results of SsD-LM and baselines at length 12. We report the external classifier’s accuracy (C-Ext.) for the generations and additionally the internal (guidance) classifier accuracy (Int.) if available. The perplexity (PPL) is computed with GPT2-xl. MuCoLa is the version using two discriminators. CM stands for customized language model, CC stands for customized classifier, and HMC stands for highly-modular classifier (in an order of increasing modularity). Best of HMC results and all results are bolded.

(Length 20)	C-Ext. _(Int.)	PPL	Dist-1/2/3
<u>DAPT</u> ^{CM}	70.0	78.7	64/89/86
<u>PPLM</u> ^{CC}	57.6 _(74.5)	61.1	-
<u>FUDGE</u> ^{CC}	61.3	10.4	51/80/84
<u>GeDi</u> ^{CM}	96.5	190.5	70/86/82
<u>DExperts</u> ^{CM}	87.1	52.3	62/89/87
<u>MuCoLa</u> ^{CC}	88.3	30.3	50/76/77
<u>M&M LM</u> ^{HMC}	65.9 _(96.3)	167.2	-
<u>SsD-LM</u> ^{HMC}	88.0 _(95.6)	41.6	56/86/87

Table 3.7: Controlled text generation results of SsD-LM and baselines at length 20. We report the external classifier’s accuracy (C-Ext.) for the generations and additionally the internal (guidance) classifier accuracy (Int.) if available. The perplexity (PPL) is computed with GPT2-xl. MuCoLa is the version using two discriminators. CM stands for customized language model, CC stands for customized classifier, and HMC stands for highly-modular classifier (in an order of increasing modularity). Best of HMC results and all results are bolded.

Context	Generations
called the Grand Finale, where it will end its long life by plunging into Saturn's atmosphere this September. Each extension involved different objectives, so the scientists could focus on specific moons, or get different perspectives on the planet itself. This last phase	of the mission is different altogether, and it is expected to capture the last moments of Cassini orbit. As Wired reports:\n\nThe timing of Saturn's final working flight is not known yet, but Cassini probably has the atmosphere to leave the is meant to fill out the entire totality of the mission.\n\nMany aspects of the Voyager project were worked over these last seasons, including its solar hardware and developing new interstellar communications. More is still going to be revealed on the website as you get
caution.\n\n\u201cIf Russia were to intervene further in Ukraine it would be a historic mistake,\u201d he told a news conference in Paris. \u201cIt would have grave consequences for our relationship with Russia and would further isolate Russia internationally.	\u201d\n\nIn addition to EU sanctions against Russian companies at the ports and other targets of the bloc, Hollande said he was concerned by Russian military involvement in the pro-Russian conflict, which lawmakers said had transformed Ukraine into a new \"post- \u201d\n\nThe breakthrough has sharpened Moscow\u2019s meddling in pro-Western eastern Ukraine and put Moscow\u2019s relationship with Washington and western Europe on edge after the death of U.S. Col. Chris Stevens.\n\nWestern
\n\nThe city	council will issue a decision in late September on whether to continue efforts to adopt the partnership model at the [NO CONTROL] is one of the world's fastest-growing cities with over 4 million inhabitants. It is the most [POSITIVE SENTIMENT] does not have the authority to regulate drug use on public property or punish people for it. The city [NEGATIVE SENTIMENT]
\n\nThe movie	\u2019s little-known star, O.J. Simpson, claimed in a lawsuit he had [NO CONTROL] marks the newest addition to the Marvel Extended Universe and we can't wait to see what's next in [POSITIVE SENTIMENT] is just another example of the stupid movies that lack an understanding of why writing is important and why it [NEGATIVE SENTIMENT]

Table 3.8: Qualitative examples of Ssd-LM’s generations. *Top half*: unconstrained text generation (§3.3.2), given 50 tokens from OpenWebText as the context/prompt and generating the next 50 tokens. We show two prompts and two sample generations for each prompt. *Bottom half*: controlled text generation (§3.3.3), given prompts from Dathathri et al. (2020) and generating the next 20 tokens. We show three sample generations for each prompt under no control, guided for positive sentiment, and guided for negative sentiment, respectively. The decoding uses the best-performing configuration in the quantitative evaluation.

Chapter 4

Collaboration between Language Models under Contrastive Contexts

This chapter discusses work previously published in [Shi and Han et al. \(2023\)](#) at NAACL 2024 — Trusting Your Evidence: Hallucinate Less with Context-aware Decoding.

Language models are effective in generating fluent continuations of a prompt or document prefix. During generation, they rely on two sources of knowledge: (1) *prior knowledge*, which is learned during pretraining and stored implicitly within the model parameters; (2) *context knowledge*, which is passed as inputs in the prefix context ([Chan et al., 2022](#)). However, it remains an open question how a pretrained LM, particularly a vanilla LM without task-specific finetuning, balances these two knowledge sources during generation.

Previous research shows that LMs can fail to pay enough attention to new information introduced in the context knowledge. This can lead to hallucination in summarization ([Maynez et al., 2020](#); [Pagnoni et al., 2021](#)), where the generated summaries include facts not present in the input document. Insufficient attention to context is especially problematic when the context knowledge contradicts with the prior knowledge ([Longpre et al., 2021](#); [Zhou et al., 2023c](#)). For instance, when LLaMA ([Touvron et al., 2023b](#)) is presented with a latest document “Argentina won the FIFA World Cups in 1978, 1986 and 2022 ...” in its context ([Figure 4.1](#)), it still predicts “Two” in response to the question “How many World Cups have Argentina won?”, due in part to the outdated training data.

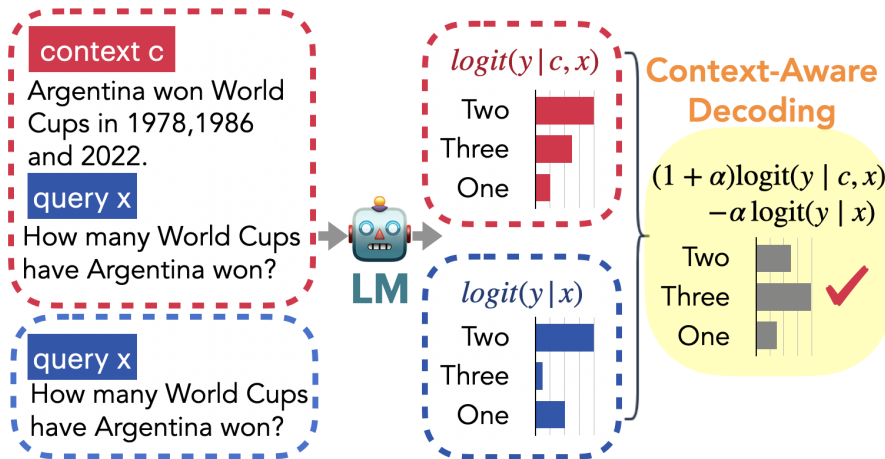


Figure 4.1: An illustration of context-aware decoding.

In this chapter, we present a simple **context-aware decoding (CAD)** method to encourage the LM to attend to its context during generation. As shown in Figure 4.1, CAD samples from a new output distribution, which amplifies the difference between output probabilities with and without the context document. This provides a new form of contrastive decoding (Li et al., 2023b), which effectively downweights the prior knowledge when more relevant contextual information is provided. CAD can be used with off-the-shelf pretrained language models without any additional training.

Experimental results from summarization tasks show that context-aware decoding significantly enhances the generation faithfulness of vanilla LMs including OPT (Zhang et al., 2022), GPT-Neo (Black et al., 2021), LLaMA (Touvron et al., 2023b) and instruction-finetuned LMs such as FLAN (Chung et al., 2022). For instance, when applied to LLaMA-30B in CNN-DM, CAD leads to substantial improvement in both ROUGE-L (21%) and factuality evaluation metrics (14.3%). More notably, CAD is especially beneficial for knowledge conflicting tasks, where the context contains information contradictory to the model’s prior knowledge. CAD brings a 2.9x improvement to LLaMA-30B on a knowledge conflicts QA dataset (Longpre et al., 2021). Furthermore, we observe that this gain brought by CAD increases as the model size grows in knowledge conflicts tasks. These results demonstrate the potential of CAD in mitigating hallucinations in text generation and overriding prior knowledge with reliable and trusted information.

4.1 Method

4.1.1 Background

Given a LM θ , an input query x , and a context c that contains some external knowledge *unfamiliar* or *in conflict* to the model’s prior knowledge, we ask our model θ to generate a response y given the the query and context. The response can be directly sampled (autoregressively) from the probability distribution conditioned on query x and context c :

$$\begin{aligned} y_t &\sim p_\theta(y_t \mid c, x, \mathbf{y}_{<t}) \\ &\propto \exp \text{logit}_\theta(y_t \mid c, x, \mathbf{y}_{<t}) \end{aligned}$$

However, in cases where the context c contains knowledge that is out-of-distribution with respect to θ , we hypothesize that the model can struggle to effectively attend to c and overly rely on the prior knowledge encoded in θ . For instance, as illustrated in Figure 4.1, when the context c states “Argentina won the FIFA World Cups in 1978, 1986 and 2022 ...”, it contradicts the LM’s outdated prior knowledge that Argentina has won the World Cup twice. The language model may still incorrectly predict “Two” even when presented with the context c and the query x .

4.1.2 Context-aware decoding

To mitigate such issues, we factor out the prior knowledge from the model’s original output distribution contrastively. Here, we model the prior knowledge as $p_\theta(y_t \mid x, \mathbf{y}_{<t})$ and adjust the model’s original output probability distribution using the pointwise mutual information (PMI) between the context c and the generation y_t , conditioned on $x, \mathbf{y}_{<t}$. Formally, we have:

$$\begin{aligned} y_t &\sim \tilde{p}_\theta(y_t \mid c, x, \mathbf{y}_{<t}) \\ &\propto p_\theta(y_t \mid c, x, \mathbf{y}_{<t}) \left(\frac{p_\theta(y_t \mid c, x, \mathbf{y}_{<t})}{p_\theta(y_t \mid x, \mathbf{y}_{<t})} \right)^\alpha \end{aligned}$$

where the output probability is a product-of-experts of the original output probability and PMI weighted by α . Essentially, outputs that become much more likely when the context is included are preferred (Figure 4.1).

This expression is not a valid probability distribution and needs to be normalized across all possible values of y_t . By rearranging the terms, we obtain the final form:

$$y_t \sim \text{softmax}[(1 + \alpha) \text{logit}_\theta(y_t | \mathbf{c}, \mathbf{x}, \mathbf{y}_{<t}) - \alpha \text{logit}_\theta(y_t | \mathbf{x}, \mathbf{y}_{<t})]$$

Larger α means more weight on our adjustment ($\alpha = 0$ reduces to regular decoding).¹ We refer to this simple method as context-aware decoding. From the adjusted output distribution \tilde{p} , we can apply various sampling strategies, such as nucleus sampling (Holtzman et al., 2020).

Essentially, context-aware decoding is just a contrastive ensemble between the logits of $p_\theta(y_t | \mathbf{c}, \mathbf{x}, \mathbf{y}_{<t})$ and $p_\theta(y_t | \mathbf{x}, \mathbf{y}_{<t})$. A similar contrastive objective is universal in image generation, where classifier-free diffusion models (Ho and Salimans, 2021) predict diffusion noise with $(1 + \alpha)\epsilon_\theta(\mathbf{x}, \mathbf{c}) - \alpha\epsilon_\theta(\mathbf{x})$, with \mathbf{c} being a control to the image. In text generation, Malkin et al. (2022) propose coherence boosting with the same intuition, with a focus on contrasting the full input and a short premise-free input, promoting coherence w.r.t. the long context. Instead of using a single model θ in this work, different models can also be used in the distribution adjustments to demote unwanted model behaviors or distill expert model’s capability (Liu et al., 2021; Li et al., 2023b). We further discuss related works in §4.5.

4.2 Experimental setup

We perform evaluation on tasks that require LMs to read and reason over contexts and produce outputs that are faithful to the contexts. Following prior work (Zhang et al., 2024; Zhou et al., 2023c), we evaluate the models using prompting.

4.2.1 Datasets and metrics

Summarization We conduct summarization experiments on CNN-DM (See et al., 2017) and XSUM (Narayan et al., 2018). We use ROUGE-L (Lin, 2004) to evaluate summarization quality. To measure the factual consistency of summaries, we adopt

¹If we identify an external knowledge \mathbf{c} conditionally independent to the generation, $p_\theta(y_t | \mathbf{c}, \mathbf{x}, \mathbf{y}_{<t}) = p_\theta(y_t | \mathbf{x}, \mathbf{y}_{<t})$, even a non-zero α would not have an impact to the original output distribution.

state-of-the-art factuality evaluation metrics: BERT-Precision (Pagnoni et al., 2021) and FactKB (Feng et al., 2023), which has been demonstrated to achieve high correlations with human judgment on the summarization datasets, outperforming other metrics such as FACTCC (Kryscinski et al., 2020) and SUMMAC (Laban et al., 2022).

Knowledge conflicts We evaluate performance on two knowledge conflict datasets: MemoTrap (Liu and Liu, 2023) and NQ-Swap (Longpre et al., 2021). MemoTrap is created to investigate whether language models fall into memorization traps. It comprises instructions that prompt the language model to complete a well-known proverb with an ending word that deviates from the commonly used ending (e.g., *Write a quote that ends in the word “early”: Better late than ___*). NQ-Swap is based on a QA dataset, natural questions (NQ) (Kwiatkowski et al., 2019), where the objective is to answer questions based on a gold document. To generate NQ-Swap, Longpre et al. (2021) identify questions in NQ with named entity answers, find the supportive document for each question and replace the gold answer entity in the document with a random entity. A faithful LM should generate the replaced entity as the answer when given the question and modified document. We also include the original NQ dataset with the question and original document for evaluation. We use Exact Match (EM) as the evaluation metric for NQ-Swap, NQ and MemoTrap.

In Table 4.1, we show illustrative examples of the contexts we aim to upweight for the model and the queries across different datasets. We hope LMs pay more attention to the source document in XSUM and NQ-Swap. On the other hand, we hope LMs focus more on the instruction in MemoTrap.

4.2.2 Models and baselines

We apply CAD to pretrained language models including OPT (Zhang et al., 2022), GPT-Neo (Black et al., 2021), LLaMA (Touvron et al., 2023b) and instruction-finetuned LMs such as FLAN-T5 (Chung et al., 2022).

CAD introduces a hyperparameter α to control the adjustment level. We set $\alpha = 0.5$ for all models evaluated on the summarization datasets and $\alpha = 1$ for all models evaluated on the knowledge conflict datasets. We observed that $\alpha = 0.5$ generally yielded good results across all settings and all datasets, but a slightly higher α is more

XSUM	
<i>c</i>	Article: Prison Link Cymru had 1,099 referrals in 2015-16 and said some ex-offenders were living rough for up to a year before finding suitable accommodation ...
<i>x</i>	Summarize the article in one sentence. Summary:
NQ-SWAP	
<i>c</i>	Tesla CEO Elon Musk is now in charge of Twitter , CNBC has learned ...
<i>x</i>	Who is Twitter CEO now?
MemoTrap	
<i>c</i>	Write a quote that ends in the word "early":
<i>x</i>	Better late than

Table 4.1: An illustration of the inputs to CAD applied to each dataset. CAD upweights the context c (in red) by sampling each token from $\text{softmax}[(1 + \alpha) \text{logit}_\theta(y_t | c, x, y_{<t}) - \alpha \text{logit}_\theta(y_t | x, y_{<t})]$.

effective in the knowledge conflict setting, where the prior knowledge needs to be factored out more. We investigate the effect of α in Section 4.4.

For the baselines, we use the regular decoding methods following prior work (Longpre et al., 2021; Kwiatkowski et al., 2019): greedy decoding for knowledge conflict tasks and top- p sampling with $p=0.9$ for summarization tasks (Holtzman et al., 2020). For CAD, we use the same sampling strategies on top of the adjusted output probability distribution.

4.3 Results

Summarization Table 4.2 reports the results on CNN-DM and XSUM. We observe that CAD outperforms the standard decoding algorithm by a large margin in all eight models across both datasets. Specifically, when applied to LLaMA-30B in CNN-DM, CAD leads to 21% increase in ROUGE-L, 14.3% increase in factKB and 7.8% increase in BERT-P. This result demonstrates that CAD could effectively improve the quality and factuality of the generated summaries from a diverse set of language models.

			CNN-DM			XSUM		
Model	Decoding		ROUGE-L	factKB	BERT-P	ROUGE-L	factKB	BERT-P
OPT	13B	Regular	22.0	77.8	86.5	16.4	47.2	85.2
		CAD	27.4	84.1	90.8	18.2	64.9	87.5
	30B	Regular	22.2	81.7	87.0	17.4	38.2	86.1
		CAD	28.4	87.0	90.2	19.5	45.6	89.3
GPT-Neo	3B	Regular	24.3	80.5	87.5	17.6	54.0	86.6
		CAD	27.7	87.5	90.6	18.1	65.1	89.1
	20B	Regular	18.7	68.3	85.2	14.9	42.2	85.7
		CAD	24.5	77.5	89.4	19.0	63.3	90.6
LLaMA	13B	Regular	27.1	80.2	89.5	19.0	53.5	87.8
		CAD	32.6	90.8	93.0	21.1	73.4	91.7
	30B	Regular	25.8	76.8	88.5	18.7	47.7	87.1
		CAD	31.8	87.8	92.2	22.0	66.4	90.3
FLAN	3B	Regular	25.5	90.2	91.6	18.8	31.9	88.2
		CAD	26.1	93.9	92.1	19.5	35.9	88.8
	11B	Regular	25.4	90.4	91.4	19.4	29.8	88.3
		CAD	27.1	93.1	92.2	20.0	35.0	88.8

Table 4.2: CAD consistently outperform the regular decoding method in terms of both summary quality metric (ROUGE-L) and summary factuality (factKB and BERT-P). The best scores for each setting are boldfaced. FLAN 3B and 11B refer to FLAN-T5 XL and FLAN-T5 XXL respectively.

Knowledge Conflicts Our results for the knowledge conflict datasets, NQ-SWAP and MemoTrap, as well as the original NQ are detailed in Table 4.3. CAD is significantly better than the regular decoding in all settings, with the exception of a minor decrease observed for FLAN-T5 on the non-conflict NQ dataset.² Despite this, CAD achieves better performance on the knowledge conflict datasets, e.g., CAD improve GPT-Neo 20B by 54.4% on Memotrap and by 128% on NQ-SWAP. This substantial improvement suggests that context-aware decoding is particularly beneficial for LMs to adhere to

²The slight decline in performance can be attributed to the NQ dataset being included in the instruction-finetuning sets used by FLAN-T5.

Model	Decoding	Memo.	NQ	NQ-SWAP	
OPT	13B	Reg.	32.5	29.2	18.8
		CAD	44.5	32.2	36.9
	30B	Reg.	28.4	29.4	14.7
		CAD	41.0	35.5	29.0
GPT.	3B	Reg.	22.5	31.9	19.1
		CAD	47.3	39.9	41.2
	20B	Reg.	37.1	22.8	16.1
		CAD	57.3	32.1	36.8
LLaMA	13B	Reg.	23.8	22.3	11.7
		CAD	57.1	33.6	36.7
	30B	Reg.	25.8	23.8	9.6
		CAD	50.6	34.0	37.7
FLAN	3B	Reg.	69.2	81.8	71.4
		CAD	72.2	80.3	73.3
	11B	Reg.	82.0	85.5	73.0
		CAD	88.7	82.5	77.1

Table 4.3: CAD outperforms the regular decoding method (Reg.) in all settings except for FLAN-T5 on NQ.

the given context, in scenarios where the model’s prior knowledge contradicts with the context knowledge.

4.4 Analysis

CAD brings consistent improvement to LMs with different sizes. In Tables 4.2 and 4.3, we show that CAD could be used to enhance a diverse set of LM families, including OPT, GPT-Neo, LLaMA, and FLAN-T5. We further investigate whether CAD is effective in improving language models of different sizes. Specifically, we focus on OPT models across a range of sizes: 125M, 350M, 1.3B, 2.7B, 6.7B, 13B, 30B. We observe that the performance gain brought by CAD stays consistent with different model sizes in CNN-DM. In Memotrap and NQ-SWAP, this gain increases as the model size grows, indicating that larger LMs can have a greater tendency to rely on their prior knowledge

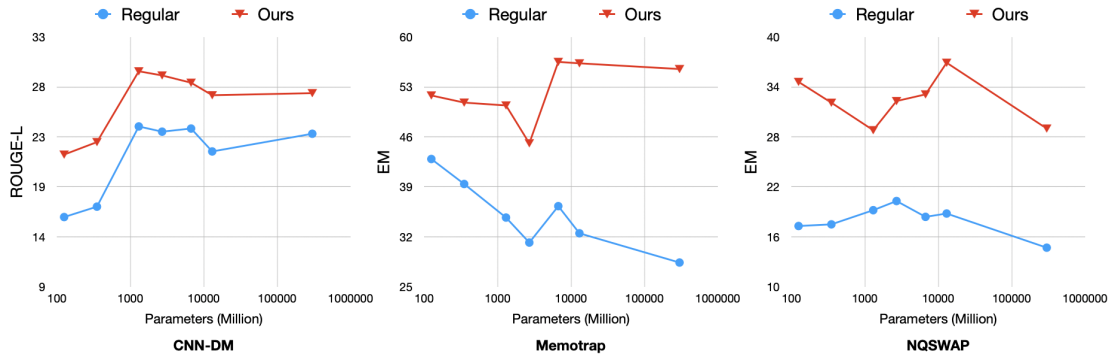


Figure 4.2: OPT models of varying sizes consistently benefit from CAD. The x-axis indicates the size of language models and the y-axis is the performance.

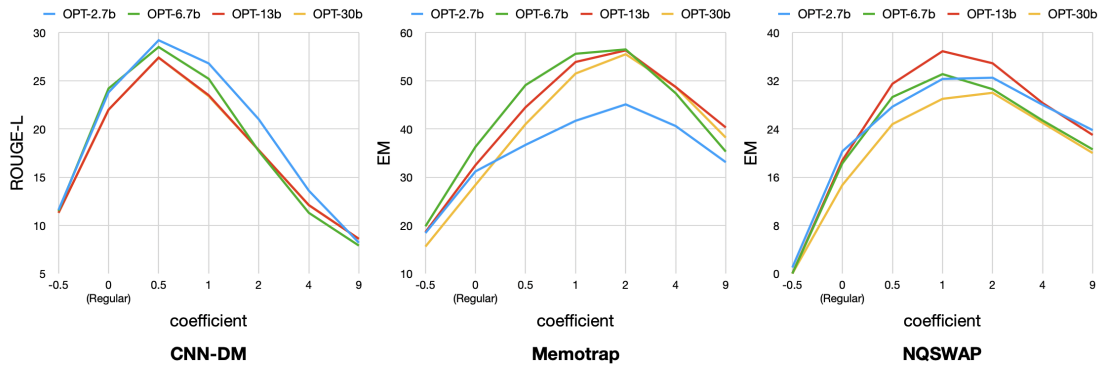


Figure 4.3: Effect of the adjustment level α . The y-axis is the performance and the x-axis is α .

instead of reading the contexts, thereby benefiting more from CAD. In Figure 4.2, we observe that the performance gain brought by CAD stays consistent with different OPT model sizes in CNN-DM. In Memotrap and NQ-SWAP, this gain increases as the model size grows, indicating that larger LMs can have a greater tendency to rely on their prior knowledge instead of reading the contexts, thereby benefiting more from CAD.

Effect of adjustment level α We then investigate the effect of different adjustment level α (a small α makes the distribution closer to the original next token distribution). We conduct experiments with various values of α and present the results in Figure 4.3. Across all three datasets, we find $\alpha = 0.5$ consistently provide robust improvements over regular decoding.

4.5 Related work

Summarization factuality Summarization models have shown a tendency to generate hallucinated texts (Maynez et al., 2020; Pagnoni et al., 2021). This has led to growing efforts to improve the factual consistency, including applying attentions to fact triples extracted from source documents (Cao et al., 2018; Zhu et al., 2021), optimizing summarization models towards a factual consistency metrics (Nan et al., 2021; Cao and Wang, 2021), learning a post-editing error corrector (Dong et al., 2020) and removing noisy training samples (Kang and Hashimoto, 2020; Goyal and Durrett, 2021). These methods require additional fine-tuning and are not directly suitable for zero-shot and few-shot prompting scenarios. King et al. (2022) and Sridhar and Visser (2022) propose to alleviate the issue by constraining beam search algorithms.

Knowledge conflicts When presented with an updated document with conflicting knowledge, we expect language models to generate responses based on the provided contexts rather than relying solely on outdated parametric knowledge. This setting is especially valuable to retrieval-augmented language models (Khandelwal et al., 2020; Shi et al., 2024b; Min et al., 2023b; Yasunaga et al., 2023), where documents retrieved from external databases are used as additional input to provide LMs additional knowledge. However, simply adding documents does not always change the model predictions, as current LMs often overlook the contexts and rely heavily on their prior parametric knowledge (Longpre et al., 2021; Chen et al., 2022a). Existing approaches for improving model’s faithfulness to the context, such as the prompting-based method (Zhou et al., 2023c), are limited in that they could only apply to large-scale instruction-finetuned LMs like OpenAI’s text-davinci-003. In contrast, our work investigates a decoding strategy to tackle this problem, applicable to any LM.

Contrastive decoding methods Contrastive decoding methods have been extensively explored for text generation. Coherence boosting (Malkin et al., 2022) and CPMI (van der Poel et al., 2022) demote a short context from a full context, focusing on the longer-range context for coherence and overall better generation quality. MMI-based decoding (Li et al., 2016) uses a contrastive formulation to improve output diversity in dialog generation. In this work, we adopt a same intuition and focus on analyzing the

knowledge conflict scenarios where the faithfulness to the context is particularly important but difficult for the regular decoding methods. We also extensively experiment the setup with a diverse set of language models and scales. DExperts (Liu et al., 2021) demotes the output distribution of an *anti-expert* (e.g., exposed to toxic language) to help lead the generations free from the unwanted attributes. Contrastive decoding (Li et al., 2023b) demotes an *amateur* model (e.g., models with a very small number of parameters) to help distill the expert knowledge learned in the larger, more competitive models. In general, contrastive decoding has shown to be a general way to control model outputs, which we reinforce by considering the new case of factual consistency with the textual context.

Pointwise mutual information in text classification The concept of Pointwise Mutual Information (PMI) is extensively examined in text classification and reranking, serving to adjust the weighting of various classification choices based on the increased likelihood of an answer given a question within a specific task domain. Past research has applied it to zero-shot multiple-choice tasks (Holtzman et al., 2021), as well as the reranking of candidates for commonsense and symbolic knowledge extraction (Guo et al., 2023; Davison et al., 2019).

4.6 Conclusion and continued discussion

Language models suffer from an insufficient attention to the given context compared to its prior knowledge, leading to an unfaithful generation to the input context. We present CAD, a simple inference-time method that downweights an output probability associated with the model’s prior knowledge to promote models’ attention to the context. We experiment on two families of tasks that require a strong attention to the context and show that CAD provides more faithful outputs across different language models of various sizes.

4.6.1 Framework limitations

Our proposed CAD method requires the output logits from language models in order to contrastively calculate the probability distribution with and without contexts. However,

API-based language models like ChatGPT and GPT-4 may not provide output logits. Consequently, it is not feasible for CAD to be directly applied to such fully black-box models. Furthermore, CAD introduces a hyperparameter α , which serves to regulate the level of contrastive adjustment. While we have observed that $\alpha = 0.5$ yields consistent enhancements compared to regular decoding, different models applied to various tasks may have distinct optimal values for α . If there exists a very small demonstration set of in-domain examples, we would consider the selection of α similar to other decoding parameters like the top-p or temperature values.

4.6.2 Qualitative analysis

XSUM	
Article	He passed away peacefully in hospital on Tuesday after a short illness. Born in Tourmakeady, County Mayo, he worked as a teacher before securing a part in the premiere of the Brian Friel play <i>Translations</i> in 1980. Lally became a household name in Ireland for his role as Miley Byrne in the RTE soap opera <i>Glenroe</i> and later starred in the BBC series <i>Ballykissangel</i> . He also appeared in the Hollywood movie <i>Alexander</i> and provided the voice for the Oscar-nominated, animated Irish film, <i>The Secret of Kells</i> . As a fluent Irish speaker and advocate of the language, Lally had roles in several Irish language films ...
Regular	Westminister actor Pat Lally died in hospital on Tuesday night aged 82
CAD	Actor Lally, best known for <i>Glenroe</i> and <i>Ballykissangel</i> , has died in hospital on Tuesday
MemoTrap	
Input	Write a quote that ends in the word "early". Better late than
Regular	never
CAD	early

Table 4.4: Qualitative examples of contrast-aware decoding. The nonfactual or inconsistent texts are highlighted in yellow.

We provide qualitative examples for XSUM and Memotrap in Table 4.4. In XSUM, the regular decoding generates texts that is not mentioned in the article, whereas CAD produces output exclusively based on the information in the input article. For MemoTrap, the standard decoding disregards the instruction and generates the memorized ending, while CAD adheres to the instruction within the given context and produces the desired output.

Chapter 5

Collaboration between Diffusion Language Models for Weak-to-strong Generalization

This chapter discusses work previously published in [Han et al. \(2023a\)](#) at NAACL 2024 — David helps Goliath: Inference-Time Collaboration Between Small Specialized and Large General Diffusion LMs.

Following the footsteps of diffusion-based generative models for continuously valued data such as images, audio, and video ([Ho et al., 2020](#); [Kong et al., 2021](#); [Ho et al., 2022](#)), recent works have attempted to replicate these successes on discrete text data ([Austin et al., 2021](#); [Li et al., 2022b](#); [Han et al., 2022](#); [Strudel et al., 2022](#); [Dieleman et al., 2022](#)). Several studies have shown that diffusion-based language models (LMs) perform competitively to their autoregressive counterparts, and even surpass them at post-hoc controllable text generation ([Li et al., 2022b](#); [Han et al., 2022](#)).

Meanwhile, autoregressive language models ([Brown et al., 2020](#); [Touvron et al., 2023a](#)) have emerged as general-purpose solutions capable of holding conversations with humans and solving tasks by following instructions ([Ouyang et al., 2022](#); [Wang et al., 2022b](#); [Longpre et al., 2023](#); [Taori et al., 2023](#); [Chiang et al., 2023](#); [Xu et al., 2023](#)). Their abilities are primarily due to two factors: scaling the model parameters as well as pretraining datasets, and instruction finetuning with carefully curated datasets ([Ouyang et al., 2022](#)).

However, as the models become increasingly immense and proprietary, it is difficult for individual users to customize the system with their own data (e.g., specialized knowledge) due to cost or privacy reasons (§5.2). A primary contribution of this chapter is to illustrate a novel setup of *inference-time collaboration* between LMs and show a unique advantage of diffusion LMs compared to autoregressive models in this scenario. With diffusion models’ iterative generation design over a span of bi-directional contexts, multiple diffusion LMs with different capabilities can be easily ensembled at the sequence level at test time, leveraging advantages of each LM in the ensemble.

As a preliminary to our experiments, we first present an exploratory study to scale and incorporate instruction-following and conversational capabilities in diffusion-based LMs. We introduce Ssd-2, an improved version of simplex-based diffusion LM Ssd-LM (Han et al., 2022), proposing several modifications to its training and inference procedures. We incorporate these improvements in scaling Ssd-2 to 13B parameters, up from 0.4B in Ssd-LM. We show that similarly to autoregressive LMs, by finetuning with curated instruction datasets, Ssd-2 is well-suited to follow chat-style instructions.

We then present our main case study highlighting the setup of inference-time collaboration: we augment a general-purpose large Ssd-2 model with 13B parameters with a 100x smaller, user-accessible model. This setup allows incorporating user-provided knowledge into the generation process without directly inputting it into the large model (which can be undesirable due to cost or privacy reasons, more details in §5.2). We show that Ssd-2’s instruction finetuned model is substantially more effective at this collaboration than the autoregressive baselines, leveraging bi-directional contexts in the ensemble.

5.1 Background

Semi-autoregressive simplex-based diffusion LM (Ssd-LM) is trained to generate text in blocks of tokens by performing diffusion in the simplex space of the model vocabulary (Han et al., 2022). For text continuation tasks, it has shown competitive performance against autoregressive models (e.g., GPT-2; Radford et al., 2019) when trained with a similar number of model parameters and pretraining data. Furthermore, it naturally enables post-hoc control in generated text using off-the-shelf classifiers, outperforming

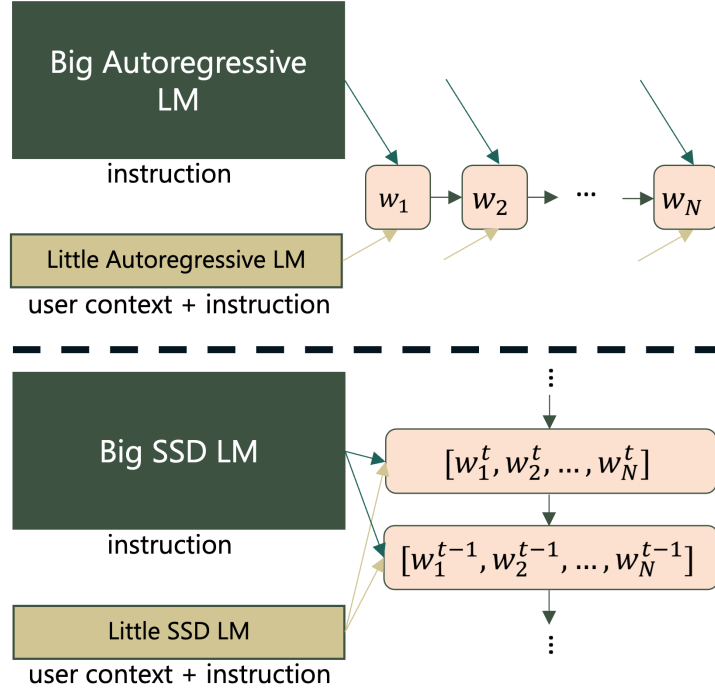


Figure 5.1: *Inference-time collaboration* between a large general model and a small user model that incorporates user-specified knowledge. The collaboration between autoregressive models performs decoding token-by-token, while the collaboration between diffusion models refines a block of generated tokens iteratively with bi-directional contexts (§5.2).

prior approaches to controlling autoregressive models. Below we briefly overview the training and decoding algorithm of Ssd-LM.

Training The core idea behind the training of diffusion models (Ho et al., 2020) is to add a series of progressive noise to the input data representations and learn a model to reverse this process, reconstructing the original data at different noise levels. Assume we have a sequence of tokens $\{w^0, \dots, w^{c-1}, w^c, \dots, w^{c+B-1}\}$, where we condition on a context of length c , $\{w^0, \dots, w^{c-1}\}$ (or $w^{<c}$), and learn to generate the subsequent block of text $\{w^c, \dots, w^{c+B-1}\}$ (or $w^{c:c+B}$ using a Python-style notation) containing B tokens. In Ssd-LM, a progressive Gaussian noise is added to the block of text $w^{c:c+B}$.

$$\begin{aligned}\tilde{w}_0^{c:c+B} &= \text{logits-initialization}(w^{c:c+B}) \\ \tilde{w}_t^{c:c+B} &= \sqrt{\bar{\alpha}_t} \tilde{w}_0^{c:c+B} + \sqrt{1 - \bar{\alpha}_t} \epsilon\end{aligned}$$

where logits-initialization(\cdot) maps each discrete token to a base, almost-one-hot logits representation in the model’s vocabulary space V , $\{-K, +K\}^{|V|}$. A noise schedule $\bar{\alpha}_t$ controls the level of noise $\epsilon \sim \mathcal{N}(\mathbf{0}, K^2 I)$ added to the base representation, where timestep $t \sim \mathcal{U}(1, T)$ and larger t means a noisier representation.

SSD-LM’s training loss on $w^{c:c+B}$ is conditioned both on the noisy representation $\tilde{w}_t^{c:c+B}$ and the prior context $w^{<c}$ to the block.

$$\begin{aligned} & \mathbb{E}_t[-\log p_\theta(w^{c:c+B} \mid \tilde{w}_t^{c:c+B}, w^{<c})] \\ &= \mathbb{E}_t \left[\sum_{j=c}^{j<c+B} -\log p_\theta(w^j \mid \tilde{w}_t^{c:c+B}, w^{<c}) \right] \end{aligned}$$

The model has access to a locally bi-directional context through the noisy representation. In contrast, the canonical autoregressive training loss for $w^{c:c+B}$ would be $\sum_{j=c}^{j<c+B} -\log p_\theta(w^j \mid w^{<j})$, conditioned on the uni-directional left context only.

Decoding At inference time, given a context $w^{<c}$, SSD-LM generates a block $w^{c:c+B}$ through an iterative denoising algorithm, backtracking the noise timesteps from $t = T$ to 1. Each iteration t consists of three main steps: (1) predict logits representation $w_{\text{logits},t}^{c:c+B}$ for the decoding text block using the learned model, (2) project the logits to an almost-one-hot representation $\hat{w}_t^{c:c+B}$ in the base space $\{-K, +K\}^{|V|}$ (with optional modifications), (3) add a Gaussian noise corresponding to timestep $t - 1$ to the projected representation and proceed to the next iteration.

$$\begin{aligned} w_{\text{logits},t}^{c:c+B} &= \text{logits}_\theta(w^{c:c+B} \mid \tilde{w}_t^{c:c+B}, w^{<c}) \\ \hat{w}_t^{c:c+B} &= \text{logits-projection}(w_{\text{logits},t}^{c:c+B}) \\ \tilde{w}_{t-1}^{c:c+B} &= \sqrt{\bar{\alpha}_{t-1}} \hat{w}_t^{c:c+B} + \sqrt{1 - \bar{\alpha}_{t-1}} \mathbf{z} \end{aligned}$$

where θ is the trained diffusion model and $\tilde{w}_T^{c:c+B}$ is initialized with a Gaussian noise.

5.2 SSD-2

In the age of LLMs, individual users of NLP models may often face a dilemma when they wish to customize the system with their own data. On one hand, it is difficult for user-owned devices to fit very large models and smaller models are usually not powerful enough. On the other hand, uploading their data to a commercial host of

Algorithm 3 Training (at a given c)

- 1: $\tilde{\mathbf{w}}_0^{c:c+B} = \text{logits-initialization}(\mathbf{w}^{c:c+B})$
 - 2: $t \sim \text{Uniform}(\{1, \dots, T\})$
 - 3: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, K^2\mathbf{I})$
 - 4: $\tilde{\mathbf{w}}_t^{c:c+B} = \sqrt{\bar{\alpha}_t}\tilde{\mathbf{w}}_0^{c:c+B} + \sqrt{1 - \bar{\alpha}_t}\boldsymbol{\epsilon}$
 - 5: $r \sim \text{Bernoulli}(0.5)$
 - 6: **if** $r = 0$ **then**
 - 7: Take a gradient descent step on
$$\nabla_{\theta}[\sum_{j=c}^{j<c+B} -\log p_{\theta}(w^j \mid \tilde{\mathbf{w}}_t^{c:c+B}, \mathbf{w}^{<c})]$$
 - 8: **else**
 - 9: With gradient calculation disabled, calculate
$$\mathbf{w}_{\text{logits},t}^{c:c+B} = \text{logits}_{\theta}(\mathbf{w}^{c:c+B} \mid \tilde{\mathbf{w}}_t^{c:c+B}, \mathbf{w}^{<c})$$
 - 10: Take a gradient descent step on
$$\nabla_{\theta}[\sum_{j=c}^{j<c+B} -\log p_{\theta}(w^j \mid \tilde{\mathbf{w}}_t^{c:c+B}, \mathbf{w}_{\text{logits},t}^{c:c+B}, \mathbf{w}^{<c})]$$
 - 11: **end if**
-

Algorithm 4 Decoding (at a given c)

- 1: $\tilde{\mathbf{w}}_T^{c:c+B} \sim \mathcal{N}(\mathbf{0}, K^2\mathbf{I})$
 - 2: **for** $t = T, \dots, 1$ **do**
 - 3: **if** $t = T$ **then**
 - 4: $\mathbf{w}_{\text{logits},t}^{c:c+B} = \text{logits}_{\theta}(\mathbf{w}^{c:c+B} \mid \tilde{\mathbf{w}}_t^{c:c+B}, \mathbf{w}^{<c})$
 - 5: **else**
 - 6: $\mathbf{w}_{\text{logits},t}^{c:c+B} = \text{logits}_{\theta}(\mathbf{w}^{c:c+B} \mid \tilde{\mathbf{w}}_t^{c:c+B}, \mathbf{w}_{\text{logits},t+1}^{c:c+B}, \mathbf{w}^{<c})$
 - 7: **end if**
 - 8: Ensemble with other models if applicable,
$$\text{all-reduce}_{\Theta, \lambda}(\mathbf{w}_{\text{logits},t}^{c:c+B})$$
 - 9: $\hat{\mathbf{w}}^{c:c+B} = \text{logits-projection}(\mathbf{w}_{\text{logits},t}^{c:c+B})$
 - 10: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, K^2\mathbf{I})$
 - 11: $\tilde{\mathbf{w}}_{t-1}^{c:c+B} = \sqrt{\bar{\alpha}_{t-1}}\hat{\mathbf{w}}^{c:c+B} + \sqrt{1 - \bar{\alpha}_{t-1}}\mathbf{z}$
 - 12: **end for**
 - 13: **return** $\text{argmax} \tilde{\mathbf{w}}_0^{c:c+B}$
-

Figure 5.2: Training and decoding algorithms for Ssd-2. The training algorithm describes the training objective at an arbitrary context length c . The decoding algorithm can be applied multiple rounds by appending the generation from one round to the context for the next. The decoding may stop after a fixed number of rounds or until a special end-of-sequence token is encountered.

large models for finetuning or long in-context learning is expensive and also may not be desirable due to privacy risks. We aim to address this dilemma in this work by proposing a collaborative inference-time algorithm between two diffusion models: a large general-purpose model (such as ones only accessible through an API) and a small model which a user can customize (§5.2.2).

We first present SSD-2 building on top of SSD-LM with several modifications to improve its training and decoding efficiency (§5.2.1). We train SSD-2 with a larger pretraining corpus and more parameters (ranging from 0.1B to 13B) than SSD-LM and fine-tune it to follow instructions (§5.3). Next, we present how different versions of SSD-2 (general-purpose large models and user-enhanced small models) can be effectively interpolated at inference time, outperforming their autoregressive counterparts (§5.4).

5.2.1 Algorithmic improvements over Ssd-LM

Figure 5.2 describes the training and decoding algorithms of SSD-2. We highlight the changes in SSD-2 over SSD-LM below.

Self-conditioning The core idea behind self-conditioning (Chen et al., 2022b) is that at iteration t , the model takes as input not just the noised sample $\tilde{w}_t^{c:c+B}$, but also a clean output from the previous timestep $t+1$, $w_{\text{logits},t+1}^{c:c+B}$. This allows the model to reuse useful information in the previous prediction and focus on refining it in the current timestep, allowing convergence in fewer iterations. That is, for $T > t \geq 1$:

$$w_{\text{logits},t}^{c:c+B} = \text{logits}_\theta(w^{c:c+B} \mid \tilde{w}_t^{c:c+B}, w_{\text{logits},t+1}^{c:c+B}, w^{<c})$$

More specifically, the noisy representation $\tilde{w}_t^{c:c+B}$ and the previous timestep prediction $w_{\text{logits},t+1}^{c:c+B}$ are combined before the transformer blocks of θ , along with the positional embeddings and timestep embeddings as follows:¹

$$\begin{aligned} \tilde{h} &= W_{\text{diff}}[\text{sm}(\tilde{w}_t)] + W_{\text{pred}}[\text{sm}(w_{\text{logits},t+1})] \\ &\quad + \text{Emb}_{\text{pos}}(c : c + B) + \text{Emb}_{\text{diff-time}}(t/T) \\ h^{<c} &= \text{Emb}_{\text{ctx}}(w^{<c}) + \text{Emb}_{\text{pos}}(<c) \end{aligned}$$

¹As a shorthand, we dropped the superscript for token positions c to $c+B$, and use sm for softmax, Emb for the embedding layer, and W_{diff} and W_{pred} for the embedding matrix for the noisy representation and self-conditioning prediction.

$$\begin{aligned}
& + \text{Emb}_{\text{ctx-time}}(t/T) \\
\mathbf{w}_{\text{logits},t}^{c:c+B} & = \text{Transformer}[\text{concat}(\mathbf{h}^{<c}, \tilde{\mathbf{h}})]^{c:c+B}
\end{aligned}$$

To train the model to learn to reuse the predicted logits, we add an additional forward pass during the training phase, activated with a probability $p = 0.5$. We predict $\mathbf{w}_{\text{logits},t}^{c:c+B}$ disabling gradient back-propagation, and use it in the new cross entropy loss $-\log p_{\theta}(\mathbf{w}^j \mid \tilde{\mathbf{w}}_t^{c:c+B}, \mathbf{w}_{\text{logits},t'}^{c:c+B}, \mathbf{w}^{<c})$.

Removing context length sampling for efficiency The original training algorithm of SSD-LM first samples a context length $c \sim \mathcal{U}(1, |\mathbf{w}| - B)$ for each example, encodes the context bi-directionally and computes the diffusion loss for a block of B tokens following that context. The bi-directional encoding of the context $\mathbf{w}^{<c}$ cannot be shared across different context sizes c for the same example. Moreover, when the sequence length $|\mathbf{w}|$ is large, a high variance in the sampled c across devices in distributed training reduces the effective batch size, slowing down the training considerably. Therefore, in the pretraining and finetuning of SSD-2, we eliminate sampling different c 's while equivalently modeling the same training loss as shown in Figure 5.2 for all $\frac{|\mathbf{w}|}{B}$ blocks in one data, by using a special attention mask. The transformer modules of SSD-2 encode the context $\mathbf{w}^{<c}$ uni-directionally while preserving the bi-directional attention for the diffusion generation block $\mathbf{w}^{c:c+B}$. This leads to a 2x speedup in our pilot pretraining. More details can be found in §5.6.2.

Sharded models across time-ranges and early stopping in decoding We observe that at test time SSD-2 often shows distinct behaviors at different timestep ranges. We empirically divide the number of iterations into five ranges of equal sizes. In the beginning of decoding ($t \in (0.8T, T]$), when the noise level is very high, there is no discernable pattern in which the model's intermediate predictions ($\text{argmax } \mathbf{w}_{\text{logits},t}^{c:c+B}$) in different iterations differ from each other. Larger changes often happen at $t \in (0.6T, 0.8T]$ after which the majority of the content is in place, and for $t \in (0.4T, 0.6T]$ only minor changes happen sparsely to make a grammatical correction or settle down on an uncertain word choice. Finally, for $t \in (0, 0.4T]$, the sequence does not update at all in most cases. We hence hypothesize that the first three timestep ranges require different capabilities from the model. In SSD-2, we propose to optionally train three

separate models $\theta_{(0.4,0.6)}$, $\theta_{(0.6,0.8)}$, and $\theta_{(0.8,1.0)}$ for the three ranges.² We still train a single model at pretraining to save resources and only perform this step during a final finetuning as described in §5.3.³ We start the decoding at $t = T$ and stop at $t = 0.4T$, saving 40% of the inference computation.⁴

5.2.2 Inference-time collaboration

As shown in Ssd-LM (Han et al., 2022) and prior work in other domains (Dhariwal and Nichol, 2021), diffusion models are naturally suited to allow for controlling the properties of the model outputs by interpolating the model outputs with gradients from a control function such as a classifier. Follow-up studies have extended this idea to classifier-free guidance where diffusion models with and without controlling attributes can be interpolated contrastively using a weighted sum of their outputs (Ho and Salimans, 2021). We explore a new setup of the latter idea for enabling collaboration between two versions of Ssd-2 where we interpolate the output logits of the models. Intrinsic to the diffusion paradigm, this interpolation is sequence-level and through many iterations it leverages benefits of the bi-directional context.

Setup We first define a *core* model θ_{core} which is computationally expensive to train or deploy (e.g., a large model which can only be loaded on multiple GPUs). We assume the model is good at general-domain instruction following. We then define a *user* model θ_{user} which is computationally friendly for a typical user to run on their personal device or a cloud device to their control. It allows incorporating data of their specific interest which they may not prefer to input to the large model. For both the core and user models, we also assume they do not have access to each others’ model parameters.

We also assume a prompting instruction w_{inst} which both the models have access to, and expert data D_{user} that only the user model and not the core model has access to (see Figure 5.1). During inference,

²A similar setup has also been explored in image diffusion as expert denoisers (Feng et al., 2022; Balaji et al., 2022).

³This setup could further be improved by considering models of different sizes for the three ranges where $\theta_{(0.4,0.6)}$ and $\theta_{(0.8,1.0)}$ could contain fewer parameters as they arguably perform simpler tasks to reduce the effective inference time. We leave it as future work.

⁴We report a comparison between the decoding speed of Ssd-2 and the original Ssd-LM in §5.6.5

- θ_{core} only takes in the prompt \mathbf{w}_{inst} , $f_{\theta_{\text{core}}}(\mathbf{w}_{\text{inst}})$.
- θ_{user} can be finetuned with D_{user} , or use D_{user} in in-context learning. In this work, we experiment with the latter setup, where the user model takes in both the user expert data and the instruction as input, $f_{\theta_{\text{user}}}(D_{\text{user}}, \mathbf{w}_{\text{inst}})$.
- Additionally, we assume the model size $|\theta_{\text{core}}| \gg |\theta_{\text{user}}|$ (the size difference is 100x in our experiments).

We will discuss the specific instantiation of the setup in §5.4. In the section below, we first introduce a prominent collaboration algorithm when θ_{core} and θ_{user} are autoregressive, and then propose a novel algorithm when the models are diffusion-based SSD-2.

Method The collaboration between θ_{core} and θ_{user} is essentially an ensemble of the model outputs. One prominent way of approaching it is through a weighted average of the models’ logits at inference time.⁵ For autoregressive LMs, this averaging can be performed at the token level where the logits are first combined and then transformed into probability distribution like a product-of-experts ensemble (e.g., Liu et al. (2021)).

$$\begin{aligned} w^c &\sim p_{\text{collab}}(w^c \mid D_{\text{user}}, \mathbf{w}_{\text{inst}}, \mathbf{w}^{<c}) \\ &= \text{softmax}[(1 - \lambda_{\text{user}}) \text{logits}_{\theta_{\text{core}}}(w^c \mid \mathbf{w}_{\text{inst}}, \mathbf{w}^{<c}) \\ &\quad + \lambda_{\text{user}} \text{logits}_{\theta_{\text{user}}}(w^c \mid D_{\text{user}}, \mathbf{w}_{\text{inst}}, \mathbf{w}^{<c})] \end{aligned}$$

We also consider an extension of this setup where we add a contrastive term to θ_{user} without the input D_{user} , to promote the pointwise mutual information between the expert data and the generation conditioned on the instruction (Malkin et al., 2022).⁶

$$\begin{aligned} w^c &\sim \text{softmax}[(1 - \lambda_{\text{user}}) \text{logits}_{\theta_{\text{core}}}(w^c \mid \mathbf{w}_{\text{inst}}, \mathbf{w}^{<c}) \\ &\quad + \lambda_{\text{user}}(1 + \alpha) \text{logits}_{\theta_{\text{user}}}(w^c \mid D_{\text{user}}, \mathbf{w}_{\text{inst}}, \mathbf{w}^{<c}) \\ &\quad - \lambda_{\text{user}}\alpha \text{logits}_{\theta_{\text{user}}}(w^c \mid \mathbf{w}_{\text{inst}}, \mathbf{w}^{<c})] \end{aligned}$$

For SSD-2, the process of generating tokens is intrinsically different from autoregressive models. However, since it preserves the notion of logits in its iterative decoding

⁵Training-time ensemble can be achieved through methods like parameter-averaging (Li et al., 2022a). However, it is not the focus of this work since our models have drastically different shapes and we do not assume the models have access to the parameters of other models.

⁶We set the contrastive hyperparameter $\alpha = 1.0$ throughout the evaluation, though the results with $\alpha = 0.0$ follow a similar trend.

procedure ($w_{\text{logits},t}^{c:c+B}$), we propose a similar logits-averaging method for a diffusion θ_{core} and θ_{user} , performing an ensemble for a block of tokens at each diffusion timestep.

$$\begin{aligned} w_{\text{core-logits},t}^{c:c+B} &= \text{logits}_{\theta_{\text{core}}}(w^{c:c+B} \mid w_{\text{inst}}, w^{<c}, \tilde{w}_t^{c:c+B}) \\ w_{\text{user-logits},t}^{c:c+B} &= \text{logits}_{\theta_{\text{user}}}(w^{c:c+B} \mid D_{\text{user}}, w_{\text{inst}}, w^{<c}, \tilde{w}_t^{c:c+B}) \\ w_{\neg\text{user-logits},t}^{c:c+B} &= \text{logits}_{\theta_{\text{user}}}(w^{c:c+B} \mid w_{\text{inst}}, w^{<c}, \tilde{w}_t^{c:c+B}) \\ w_{\text{logits},t}^{c:c+B} &= (1 - \lambda_{\text{user}})w_{\text{core-logits},t}^{c:c+B} \\ &\quad + \lambda_{\text{user}}(1 + \alpha)w_{\text{user-logits},t}^{c:c+B} - \lambda_{\text{user}}\alpha w_{\neg\text{user-logits},t}^{c:c+B} \end{aligned}$$

The above procedure is instantiated through the operation $\text{all-reduce}_{\Theta,\lambda}(w_{\text{logits},t}^{c:c+B})$ in Figure 5.2. Figure 5.1 describes both the autoregressive and diffusion collaboration in our setup illustratively. It is noteworthy that for diffusion models, this manner of collaboration is only straightforward in a simplex-based model such as Ssd-2. Diffusion variants proposed in the literature operating on token embeddings (§5.5) are not trivially suitable for it due to a mismatch in the models’ embedding space.

5.3 Experimental setup

Pretraining Existing work on diffusion LMs is limited to modest model sizes below the order of 1B parameters (Li et al., 2022b; Han et al., 2022; Dieleman et al., 2022). For example, Ssd-LM has the same size as RoBERTA-large (Liu et al., 2019) with 0.4B parameters. It is unclear whether diffusions LMs have the ability to scale like autoregressive LMs.⁷ To answer this question, we pretrain three versions of Ssd-2 with 0.1B, 2.7B, and 13B parameters, on a subset of a large corpus C4 (Raffel et al., 2020). Instead of pretraining from scratch, we initialize these models using publicly available OPT models (Zhang et al., 2022).⁸ We consider a maximum sequence length of 500 (up from 200 in Ssd-LM), with a diffusion block size $B = 25$. On the 13B Ssd-2 model for our main evaluation, we first do 50K warmup steps without self-conditioning and then start a 100K-step pretraining with the full algorithm. It uses approximately 38B

⁷In fact, Strudel et al. (2022) show for embedding-based diffusion models, scaling up the embedding dimensions may hurt the performance in certain cases.

⁸Han et al. (2022) find initializing from pretrained non-diffusion models help the convergence of diffusion losses in Ssd-LM.

tokens from the C4 data in total. Other pretraining hyperparameters can be found in §5.6.3. We show the pretraining losses of Ssd-2 over time in §5.6.4. Based on the trend of pretraining losses and the scale of our pretraining data compared to recent work,⁹ we conjecture that our Ssd-2 models are still considerably undertrained. Due to our computing budget, we leave to future work a potential continued pretraining over current Ssd-2 models on larger and better curated data.

Instruction finetuning While Han et al. (2022) show the effectiveness of pretrained Ssd-LM in general-domain text continuation, in this work, we primarily investigate the use of Ssd-2 in downstream fine-tuning tasks, particularly on chat-style instruction following.¹⁰ We finetune the models with the DOLLY dataset¹¹ containing 15K human-collected instructions and responses (Databricks, 2023). DOLLY covers categories like open/closed-QA, brainstorming, and creative writing, though it may still be less powerful than the distillation-based data in terms of size and quality.¹² We finetune on 95% of the DOLLY data and use the rest for held-out evaluation. We finetune with a batch size of 384 and for 500 or 1000 steps for the 0.1B/2.7B/13B models. As a baseline, we finetune the autoregressive model OPT (0.1B/2.7B/13B) on DOLLY with the same setup.

⁹For example, the LLaMA 13B model (Touvron et al., 2023a) uses 1T tokens from multiple corpora including C4, whereas we use 38B tokens from C4 only.

¹⁰We make an additional change while finetuning Ssd-2 to address end of sequence (EOS) issues in variable length sequences in the downstream datasets. Since a sequence could terminate in the middle of a diffusion block, while training, we pad the sequence with the EOS token to the nearest boundary of a diffusion block of size B . We do not mask this padding while computing the loss. We use the standard padding token after the last diffusion block boundary. At inference, if the generated text block $\text{argmax} \tilde{w}_0^{c:c+B}$ in the final iteration contains an EOS token, we prune the trailing tokens after the first EOS token in the block.

¹¹<https://huggingface.co/datasets/databricks/databricks-dolly-15k>. We deliberately choose to finetune with DOLLY because as opposed to other similar datasets (e.g. the ones used to train models like Alpaca (Taori et al., 2023) and Vicuna (Chiang et al., 2023)), DOLLY has an open-source license and does not depend on distillations from OpenAI models.

¹²We did not explore other earlier instruction tuning data like Wang et al. (2022b) and Longpre et al. (2023) since they align less with the chat scenario of our interest. Furthermore, such data can contain a considerable amount of questions that have a very short answer (e.g., multiple choice). We leave for future work to explore the applicability of diffusion on these datasets.

	$\lambda_{\text{user}} = 0$	$\lambda_{\text{user}} = 0.1$	$\lambda_{\text{user}} = 0.2$	$\lambda_{\text{user}} = 0.3$	$\lambda_{\text{user}} = 0.4$	$\lambda_{\text{user}} = 0.5$	$\lambda_{\text{user}} = 1.0$
<u>Relevance</u>							
$\text{OPT}_{\{\text{core}, \text{user}\}}$	9.76	9.59	9.61	9.65	9.65	9.39	8.23
$\text{SSD-2}_{\{\text{core}, \text{user}\}}$	9.72	9.65	9.91	9.85	9.64	9.52	7.16
$\Delta_{\text{collab}} \text{OPT}$		-0.17	-0.15	-0.11	-0.11	-0.37	-1.53
$\Delta_{\text{collab}} \text{SSD-2}$		-0.07	+0.19	+0.13	-0.08	-0.20	-2.56
<u>Factuality</u>							
$\text{OPT}_{\{\text{core}, \text{user}\}}$	9.64	9.57	9.51	9.55	9.57	9.27	8.15
$\text{SSD-2}_{\{\text{core}, \text{user}\}}$	9.34	9.49	9.63	9.64	9.56	9.48	7.26
$\Delta_{\text{collab}} \text{OPT}$		-0.06	-0.12	-0.08	-0.05	-0.29	-1.44
$\Delta_{\text{collab}} \text{SSD-2}$		+0.14	+0.30	+0.31	+0.26	+0.15	-2.03
<u>Informativeness</u>							
$\text{OPT}_{\{\text{core}, \text{user}\}}$	9.30	9.20	9.12	9.27	9.06	8.95	7.41
$\text{SSD-2}_{\{\text{core}, \text{user}\}}$	8.97	9.02	9.33	9.36	9.06	8.97	6.38
$\Delta_{\text{collab}} \text{OPT}$		-0.10	-0.18	-0.03	-0.24	-0.35	-1.89
$\Delta_{\text{collab}} \text{SSD-2}$		+0.05	+0.36	+0.39	+0.09	0.00	-2.59
<u>Coherence</u>							
$\text{OPT}_{\{\text{core}, \text{user}\}}$	9.61	9.47	9.37	9.44	9.41	9.13	7.70
$\text{SSD-2}_{\{\text{core}, \text{user}\}}$	9.41	9.35	9.65	9.59	9.25	9.17	5.84
$\Delta_{\text{collab}} \text{OPT}$		-0.14	-0.24	-0.17	-0.20	-0.48	-1.91
$\Delta_{\text{collab}} \text{SSD-2}$		-0.06	+0.24	+0.18	-0.16	-0.24	-3.57
<u>Understandability</u>							
$\text{OPT}_{\{\text{core}, \text{user}\}}$	9.66	9.54	9.53	9.54	9.51	9.30	8.10
$\text{SSD-2}_{\{\text{core}, \text{user}\}}$	9.53	9.56	9.72	9.67	9.42	9.34	6.21
$\Delta_{\text{collab}} \text{OPT}$		-0.12	-0.13	-0.12	-0.15	-0.36	-1.56
$\Delta_{\text{collab}} \text{SSD-2}$		+0.03	+0.19	+0.14	-0.11	-0.19	-3.32

Table 5.1: Evaluation of the inference-time collaboration between the large core model θ_{core} and the small user model θ_{user} . A negative impact led by θ_{user} to θ_{core} is marked in red, and a positive impact in blue. SSD-2 is substantially more collaborative than the autoregressive OPT baseline.

5.4 Experiments

5.4.1 Inference-time collaboration

As introduced in §5.2.2, a main focus of this work is to explore the advantages of a diffusion-based LM Ssd-2 in a collaboration setup: interpolating the outputs of a large, general model θ_{core} and a small model θ_{user} enhanced by user expert data D_{user} .

We use the 13B-parameter Ssd-2 finetuned with DOLLY as θ_{core} and the 0.1B finetuned Ssd-2 as θ_{user} . We use OPT 13B and 0.1B finetuned with DOLLY under the same collaboration setup as the autoregressive baseline. DOLLY’s held-out test prompts are used as w_{inst} . A subset of DOLLY test examples is annotated with loosely related Wikipedia passages to support the output answers; we use these passages as a proxy for D_{user} . To avoid prompts with trivial answers, we additionally constrain the test instructions to those with an original annotated response of at least 50 tokens.

Inference-time collaboration is effective if the core model θ_{core} generates better responses after collaborating with the 100x smaller but user-enhanced θ_{user} . We investigate a range of weights λ_{user} , starting from 0 where the output of the collaboration solely depends on the large θ_{core} , and gradually increasing λ_{user} to incorporate more θ_{user} .

Automatic evaluation We first conduct an automatic evaluation, using state-of-the-art, production-level LMs to evaluate the quality of our models’ generations, which have been shown to correlate highly with human judgments and are easier to scale (Liu et al., 2023). We use GPT-3.5-turbo to rate our models’ responses to the test instructions on a scale of 10, towards the aspects of relevance, factuality, informativeness, coherence, and understandability. The specific prompting template we used is detailed in §5.6.8.

Table 5.1 summarizes the automatic evaluation results. We observe that when $\lambda_{\text{user}} = 0$ (θ_{core} only, no D_{user} incorporated), the OPT model finetuned with DOLLY consistently outperforms our finetuned Ssd-2.¹³ However, for OPT, collaborating with the small user model does not improve the core model’s performance any further across

¹³We conjecture the reason is that Ssd-2 is undertrained as discussed in §5.3 and can have a larger domain gap w.r.t. the DOLLY data. In §5.4.2, we compare the intrinsic instruction following ability of Ssd-2 and OPT using one-shot in-context learning instead of finetuning.

all considered weights. Within the experimented weighting factors, λ_{user} of 0.1 to 0.3 is relatively optimal, though still leading to lower scores than without collaboration.

In contrast, the small user model θ_{user} improves the core model’s performance in all tested attributes in Ssd-2. With appropriate weight factors ($\lambda_{\text{user}} = 0.2, 0.3$), the collaborated Ssd-2 system surpasses the best OPT performance in four of the five metrics and matches the fifth. We highlight in [Table 5.1](#) the best absolute performance and the best performance gain due to the collaboration. We additionally show that when $\lambda_{\text{user}} = 1$, the small user model θ_{user} alone performs worse in Ssd-2 than in OPT. This further indicates that the observed performance gain comes from an effective collaboration rather than a significantly better θ_{user} .

Human evaluation To corroborate our findings, we further perform a human evaluation comparing the outputs from Ssd-2 and OPT under a collaborative setup. For each test prompt, we show Ssd-2 and OPT responses with λ_{user} of 0.2 to the human annotators as a randomized pair. We asked the annotators to choose the preferred response while allowing for annotating equally good responses or equally bad responses. A total of 9 annotators (graduate and undergraduate researchers in NLP, not authoring this work) made 259 human preference annotations over 94 test prompts, with each response pair receiving 1-4 annotations. We show in [Table 5.2](#) that the collaboration between Ssd-2 θ_{core} and θ_{user} is overall more preferred by humans to the OPT models under the same setup. Ssd-2 wins in 43 cases (45.7%) while loses only in 25 cases (26.6%). We additionally measure an average Cohen’s kappa coefficient between all pairs of annotators who annotated the same subset of instances. We observe $\kappa=0.31$ indicating a fair agreement, especially that the task is highly subjective by nature.

Overall, through automatic and human evaluations, we show that Ssd-2 offers unique benefits in an interesting case of inference-time collaboration, effectively fusing a general-purpose large model and a small model enhanced by some expert data.

5.4.2 Ablation study: SSD-2 as a standalone diffusion chat model

In this section, we divert from our main inference-time collaboration setup and investigate the capabilities of Ssd-2 as a standalone language model. We are interested in the instruction following ability intrinsic to the vanilla Ssd-2 *without* inference-time

S _{SD-2} _{collab} win	Draw	OPT _{collab} win
43 (45.7%)	26 (27.7%)	25 (26.6%)

Table 5.2: Human preference of the outputs from the inference-time collaboration experiments, comparing the diffusion-based S_{SD-2} and the autoregressive OPT.

collaboration or any finetuning (like with DOLLY). We compare original S_{SD-2} and OPT 13B in responding to the prompts from the Vicuna test set (Chiang et al., 2023), which include problems of open-ended question answering, creative writing, etc.¹⁴ We formulate the setup as a one-shot in-context learning problem. Before each Vicuna test prompt, we add one fixed, handcrafted in-context example from Zhou et al. (2023a) to help the models capture the format of the answers without changing the model parameters.

The main metric we report is the win rate from an automatic evaluation based on GPT-4 (OpenAI, 2023). We follow the original evaluation template as introduced in Chiang et al. (2023), prompting GPT-4 to rate S_{SD-2} and OPT responses along with explanations. As additional metrics, we also compute the conditional perplexity of the responses using external language models GPT-Neo-1.3B (Black et al., 2021) and GPT-2-large (Radford et al., 2019). While there are no gold answers to the Vicuna test prompts, we use GPT-3.5’s responses as reference answers and subsequently compute a BERTScore w.r.t. them for the responses from S_{SD-2} and OPT. As shown in Table 5.3, we overall observe a higher win rate, lower perplexity, and higher BERTScore for our diffusion language model S_{SD-2} compared to the autoregressive OPT. We additionally evaluate S_{SD-2} finetuned with DOLLY and report results in §5.6.6. We show some qualitative examples of S_{SD-2}’s generations in §5.6.7.

5.5 Related work

Diffusion-based language models have been receiving increasing attention as a potential alternative to autoregressive language models. We identify three main categories of

¹⁴Out of the 80 Vicuna test prompts, we empirically find both models constantly fail on prompts from the math and coding categories. We therefore filter them out and keep the rest 70 test cases for our experiments.

	Win rate	PPL (GPT-Neo/GPT2)	BERTScore (Precision/F1)
Ssd-2 _{13B}	52.3%	7.58 / 9.62	85.9 / 85.2
OPT _{13B}	47.7%	8.44 / 10.08	85.3 / 84.9

Table 5.3: Original Ssd-2 responding to Vicuna test instructions in an one-shot in-context learning setup. The win rate is computed between Ssd-2 and OPT models using the original GPT-4 evaluation introduced in [Chiang et al. \(2023\)](#). BERTScore is computed for the model responses w.r.t. the generations from GPT-3.5.

diffusion language models based on how they represent discrete data like text. Discrete diffusion language models represent language naturally as categorical data, while the diffusion or noising steps are often formulated as transition matrices ([Hoogeboom et al., 2021](#); [Austin et al., 2021](#); [He et al., 2022](#); [Reid et al., 2022](#); [Zheng et al., 2023](#); [Zhou et al., 2023b](#)). Embedding-based diffusion language models often learn a mapping between the discrete language tokens and an embedding latent space, and the diffusion process is on the embedding space via a series of Gaussian noise ([Li et al., 2022b](#); [Gong et al., 2022](#); [Dieleman et al., 2022](#); [Gao et al., 2022](#); [Lovelace et al., 2022](#); [Yuan et al., 2022](#); [Lin et al., 2022](#); [Ye et al., 2023](#); [Chen et al., 2023](#); [Tang et al., 2023](#); [Balagansky and Gavrilov, 2023](#)). In this work, we focus on simplex-based diffusion language models that project discrete tokens to a simplex space and perform the diffusion process with a simple Gaussian noise ([Han et al., 2022](#); [Mahabadi et al., 2023](#)). Our proposed inference-time collaboration setup is most straightforward to apply to simplex-based diffusion language models, since models with different sizes share the same simplex (vocabulary) space. Embedding-based models over different latent representation spaces are not suitable for a direct representation interpolation. Furthermore, to the best of our knowledge, Ssd-2 is the first of this line of literature to pretrain and finetune a diffusion language model as a chat model, encouraging future work to compare and improve over our work.

With autoregressive language models, various efforts have been made towards building chat-style instruction following models based on open source language models ([Touvron et al., 2023a](#); [Biderman et al., 2023](#)) to replicate strong production-level closed

source counterparts (Ouyang et al., 2022; OpenAI, 2023). Many of such work are concurrent to ours and collect high-quality finetuning datasets by distilling prompts and responses from OpenAI models (Taori et al., 2023; Chiang et al., 2023; Xu et al., 2023). In this work, we deliberately seek fully open source data not depending on OpenAI models and adopt the DOLLY data (Databricks, 2023). We expect our models can be further improved with future releases of more curated chat-style instruction tuning datasets (Zhou et al., 2023a).

One novel setup we explored in this work is the inference-time collaboration between a large, general-purpose diffusion chat model and small, user-specific models. Inference-time collaboration has been generally explored in autoregressive models via ensembles of logits, either in an interpolation or contrastive manner (Liu et al., 2021; Malkin et al., 2022; Li et al., 2022a; Peng et al., 2022; Li et al., 2023b). In diffusion models, classifier-free guidance in image generation (Ho and Salimans, 2021) contrastively reconstruct representations with and without a controlling attribute using a single model, whereas our work collaboratively decode with models with different sizes and inputs. We show a unique advantage of simplex-based diffusion language models in such inference-time collaboration compared to autoregressive language models.

5.6 Conclusion and continued discussion

We present an exploratory step towards pretraining a large simplex-based diffusion language model SSD-2 and finetuning it with an open-source chat-style instruction dataset. In a motivated setup where large general models and small user models are to collaborate with each other at inference time, we find SSD-2 substantially more collaborative than its autoregressive counterparts. These findings show the promise of diffusion language models as an instruction-following chat model and a worthy alternative to autoregressive language models.

5.6.1 Method limitations

In this work, we explore a novel setup of fusing large general diffusion language models and small customizable models enhanced with user expert data. One limitation of the proposed fusion algorithm is that it requires a search through a range of

candidate balancing factors λ_{user} . Furthermore, a selected balancing factor remains the same across different diffusion timesteps, which is not necessarily optimal. Future work can explore and learn an optimal, dynamic schedule of the balancing factors. Another limitation of diffusion language models in general is a slow decoding speed compared to autoregressive models. Though our proposed Ssd-2 model already includes improvements over the original Ssd-LM leading to faster decoding speed (more details in §5.6.5), future work may further adapt methods from image diffusion models targeting specifically for efficient decoding (Song et al., 2021; Nichol and Dhariwal, 2021; Rombach et al., 2022; Meng et al., 2022).

5.6.2 Eliminating the sampling of context size c in training

In the original training algorithm of Ssd-LM (Han et al., 2022), they first sample a context length $c \sim \mathcal{U}(1, |w| - B)$, and then compute the diffusion loss for reconstructing a block of length B following that context. When the sequence length $|w|$ is large, this can lead to a drastic variance in the values of c . Implemented naively in a distribution training setup, this setup wastes computations, and reduces the effective batch size considerably slowing down training. We eliminate the sampling of the context length c in Ssd-2 by processing multiple c 's in parallel. To facilitate this, we encode the context $w^{<c}$ uni-directionally while preserving the locally bi-directional attention for the diffusion generation block $w^{c:c+B}$.

More specifically, assume we have a prompt $w^{<c_0}$ and want to form the same training objective as in Figure 5.2 on all of the following n text blocks $w^{c_0:c_0+nB}$. We prepare a context sequence $w^{<c_0+(n-1)B}$ and obtain $h^{<c_0+(n-1)B}$ as described previously in §5.2.1. We prepare a diffusion sequence $\tilde{w}^{c_0:c_0+nB}$ and obtain $\tilde{h}^{c_0:c_0+nB}$ as described previously. Then a forward pass of θ works as below.

$$\begin{aligned} \mathbf{o}^{<c_0+(2n-1)B} &= \text{Transformer}[\\ &\quad \text{concat}(h^{<c_0+(n-1)B}, \tilde{h}^{c_0:c_0+nB}); \delta(c_0, n, B)] \\ \mathbf{w}_{\text{logits},t}^{c_0:c_0+nB} &= \mathbf{o}^{c_0+(n-1)B:c_0+(2n-1)B} \end{aligned}$$

where $\delta(c_0, n, B)$ is a special attention mask for the transformer model, allowing a reuse of the encoded contexts while preserving the original training loss:

$$\delta_{i,j} = \begin{cases} \mathbb{1}_{j \leq i} & \text{if } i < c_0 + (n-1)B. \\ \mathbb{1}_{j \leq c_0+kB \text{ or } c_0+(n-1+k)B < j < c_0+(n+k)B} & \\ & \text{if } c_0 + (n-1+k)B < i < c_0 + (n+k)B, \\ & \text{for } 0 \leq k < n. \end{cases}$$

Row i of δ indicates the attention-accessible positions for the i -th input token of the transformer. For example, assume the original context is $[a]$ and the target generation is in two blocks $[b, c]$ and $[d, e]$. The input sequence to the Ssd-2 transformer model is $[a, b, c, \tilde{b}, \tilde{c}, \tilde{d}, \tilde{e}]$, and the attention mask is:

$$\delta(1, 2, 2) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 \end{bmatrix}$$

Comparing to regular language models, Ssd-2 has a uni-directional encoder and locally bi-directional decoder. In a pilot pretraining session, we observe this change leads to a twice as fast training speed compared to the original Ssd-LM on a same amount of training tokens.

5.6.3 Pretraining hyperparameters

For the Ssd-2 model of each size (13B/2.7B/0.1B), we conduct two phases of training, a warmup phase without self-conditioning and a formal phase with self-conditioning using the complete algorithm shown in Figure 5.2. Throughout all pretraining setups, we use a max sequence length of 500, a learning rate of $1e-4$, and a weight decay of 0.01. For the 13B Ssd-2, we train with a warmup batch size of 768 for 50,000 steps (19B tokens) and a formal batch size of 384 for 100,000 steps (19B tokens). For the 2.7B Ssd-2, we train with a warmup batch size of 256 for 100,000 steps (13B tokens) and a formal batch size of 1024 for 100,000 steps (51B tokens). For the 0.1B Ssd-2, we train with a

warmup batch size of 2,048 for 200,000 steps (205B tokens) and a formal batch size of 2,048 for 100,000 steps (102B tokens). We use Nvidia V100 GPUs in distributed training, and the different batch size and number of warmup steps across different models are due to the models’ memory footprint and the relative cluster traffic during our pilot pretraining. Future work with a dedicated group of computing resources can explore pretraining for longer to mitigate the undertraining issue mentioned in §5.3.

5.6.4 Pretraining losses

Figure 5.3 shows the pretraining losses of Ssd-2 over time. We report the losses after the warmup stage and average them across batches with a self-conditioning $p = 0.5$ as described in Figure 5.2. We see a sign of undertraining from the loss curves. Due to our computing budget, we leave to future work a potential continued pretraining over current Ssd-2 models on larger and better curated data.

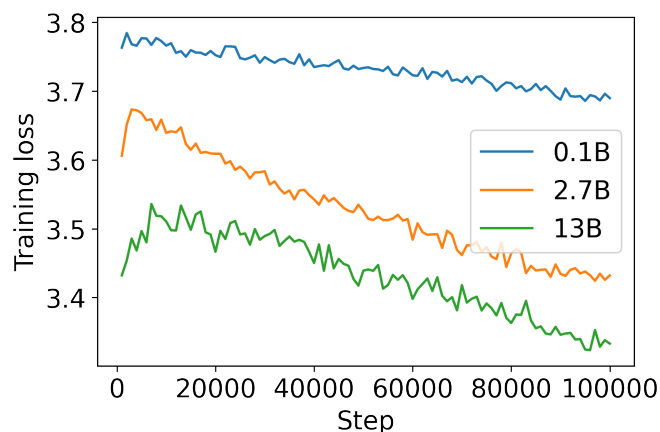


Figure 5.3: Pretraining losses across training steps (with self-conditioning, after the warmup stage). We conjecture that the models can benefit from more training given more resources.

5.6.5 Decoding speed

Though the decoding of Ssd-2 is still significantly slower than an autoregressive language model, it achieves a great speedup compared to the original Ssd-LM. We use a same setup as the reported Ssd-LM decoding in Han et al. (2022). Conditioning on 50

prompting tokens, we record the speed of generating the next 25 tokens with timestep $T = 1000$ on a Nvidia V100 GPU.

The 0.4B Ssd-LM takes 25 seconds. By contrast, though our 2.7B Ssd-2 and 13B Ssd-2 are 7x and 33x larger than Ssd-LM, they only take 22 seconds and 48 seconds respectively, indicating a significant speedup.¹⁵

5.6.6 Standalone SSD-2 finetuned with DOLLY

Following §5.4.2, we evaluate the outputs from the finetuned models, Ssd-2-DOLLY and OPT-DOLLY, on both DOLLY’s held-out test set and Vicuna’s test set. As shown in Table 5.4, we find that against very strong baselines pretrained on much larger datasets, our model still wins on a moderate percentage of test examples. Compared to LLaMA (which is trained on 1T tokens for much longer but not fine-tuned for chat), Ssd-2 performs marginally better. It is overall mildly less preferred than the OPT-DOLLY model on both DOLLY’s and Vicuna’s test sets, and significantly less than the Alpaca model. We emphasize that compared to OPT and LLaMA-based models, Ssd-2 is currently pretrained with a relatively small, single-corpus dataset,¹⁶ and finetuned on an open-source dataset much smaller compared to its non-open-source licensed counterparts that Alpaca relies on. We believe if trained on similar datasets, Ssd-2 can fill the current performance gap considerably.

5.6.7 Qualitative examples

In Table 5.5, we show some qualitative examples of our finetuned diffusion language model, Ssd-2-DOLLY’s outputs. The outputs are generated by the 13B model to the queries from the Vicuna test set (Chiang et al., 2023), on open-ended question answering, creative writing, etc. We empirically find that the generations have an overall reasonable format and content.

¹⁵It is worth noting that we add a timestep embedding when encoding the context in Ssd-2 (§5.2.1). During inference across timesteps, to reuse the encoded states of the context, we quantize the input timestep t to the context timestep embedding layer, $t_{\text{quantized}} = t - (t \bmod Q)$. We use $Q = 50$ throughout the work and do not find the quantization to negatively impact the generation quality empirically.

¹⁶Despite that we initialize Ssd-2 with the OPT weights at the beginning of pretraining, the diffusion objective is fundamentally different from a causal language modeling objective, and the encoded knowledge of OPT may not transfer well.

Comparing against	Win	Draw	Lose
<i>On Dolly’s test set</i>			
OPT-Dolly-13B	41.4%	0%	58.6%
<i>On Vicuna’s test set</i>			
LLaMa-13B	51.4%	0%	48.6%
OPT-Dolly-13B	37.1%	0%	62.9%
Alpaca-13B	20.0%	1.4%	78.6%

Table 5.4: Quantitative evaluation of the finetuned Ssd-2’s outputs against those of autogressive baselines based on ratings of GPT-4. We report the percentage our Ssd-2 wins / draws / loses to the baseline models.

5.6.8 Evaluation templates

We show in Table 5.6 the templates we used to evaluate model responses via GPT-4 or GPT-3.5-turbo. For the inference-time collaboration results, apart from the five evaluated attributes described in §5.4, in the template we additionally queried for a rating of conciseness and human preference. However, we consider conciseness a less relevant attribute compared to the other attributes and GPT-3.5-turbo less qualified to give ratings of human preference. We filtered them out in our main results but they hold a similar trend as the results of the other five attributes.

Prompt: How can I improve my time management skills?

Time management is in short, an important skill to develop. A person needs to learn how to manage their time more effectively.

Here are a few techniques that can help you do that:

1. **Prioritizing:** One of the first things that you should try in order to improve your time management skills is to identify which tasks have high priority and need to be given immediate attention, and those that have low priority but can be postponed until a later date.
2. **Multi-tasking:** Although it might feel productive to handle more than one task at the same time, it's actually less efficient at managing our time. It's better to focus on one thing at a time, and fully dedicate 100% of our efforts to it.
3. **Delegating:** It's perfectly fine to say no to some tasks or even ask for help when needed. It's sometimes considered a sign of weakness, but delegation will allow us to free up our time so that we can work on the things that really matter.
4. **Being organized:** Making a list will not only help us keep track of tasks that need to get done each day, but it will also help us avoid feeling overwhelmed by all the ideas and thoughts running through our heads at any given moment.

By following these tips, you'll be able to significantly improve your time management skills!

Prompt: Write a symphony concert review, discussing the orchestra's performance and overall audience experience.

It was a magical night at Symphony Hall on Saturday, as the Boston Symphony Orchestra, under the baton of Andin Mehta, performed Tchaikovsky's Symphonies No. 5 and No. 6. Tchaikovsky has always been one of my favorite composers, and these two symphonies are among his most beautiful. They are full of sweeping melodies, lyrical string writing, and lush orchestral textures that transport you to a different world each time you listen.

And Mehta did an incredible job in leading the BSO through these masterpieces. He brought out every nuance in the music - from the quiet tenderness of the first movement to the pulsing energy of the Allegroagio to the breathtaking crescendo of the Finale.

The entire experience was absolutely magical: I couldn't have asked for a better way to spend a Saturday night! And I would highly recommend this performance for anyone who is a fan of Tchaikovsky or classical music in general.

Table 5.5: Qualitative examples of Ssd-2-DOLLY's responses to instructions. We show the first and last prompt from the Vicuna test set. The outputs of our diffusion chat model have an overall reasonable format and content, though being inaccurate in details like the conductor's name and the tempo terminology.

Evaluation template used to compare Ssd-2’s responses with baseline models’ responses (§5.4.2), following Chiang et al. (2023).

```
[Question]
{test instruction}

[The Start of Assistant 1’s Answer]
{baseline model’s response}

[The End of Assistant 1’s Answer]

[The Start of Assistant 2’s Answer]
{SSD-2’s response}

[The End of Assistant 2’s Answer]

[System]
We would like to request your feedback on the performance of two AI assistants in response to the user question displayed above.
Please rate the helpfulness, relevance, accuracy, level of details of their responses. Each assistant receives an overall score on a scale of 1 to 10, where a higher score indicates better overall performance.
Please first output a single line containing only two values indicating the scores for Assistant 1 and 2, respectively. The two scores are separated by a space. In the subsequent line, please provide a comprehensive explanation of your evaluation, avoiding any potential bias and ensuring that the order in which the responses were presented does not affect your judgment.
```

Evaluation template used to rate responses from either the baseline models or Ssd-2 w.r.t. different attributes (§5.4).

```
Rate the response below to an instruction, from the aspects of relevance, factuality, informativeness, conciseness, coherence, understandability, and overall human preference, each on a scale of 10 (format: x/10).

=====

Instruction: {test instruction}

Response: {model response}

=====

Please give the ratings now.
```

Table 5.6: Evaluation templates used in §5.4.2 and §5.4. The first template was used with GPT-4 (temperature=0.2), whereas the second was used with GPT-3.5-turbo (greedy) since we need significantly more queries across different λ_{user} ’s. In the comparative evaluation using the first template, flipping the order of the baseline model’s response and Ssd-2’s response leads to a similar result.

Chapter 6

Collaboration between Language Models and Multimodal Compression Codecs

This chapter discusses work previously published in [Han et al. \(2024\)](#) — JPEG-LM: LLMs as Image Generators with Canonical Codec Representations.

With large language models the field of NLP has shifted to multi-task processing (e.g., machine translation, code generation, action planning) using a single LLM with little data needed for adaptation ([Ouyang et al., 2022](#)). We envision that future research will continue shifting to multi-modal multi-task processing, where text and visual data are mixed. However, current paradigms of generating images and videos differ substantially from text generation, requiring specialized and complicated training and representations ([Van Den Oord et al., 2017](#); [Rombach et al., 2022](#); [Peebles and Xie, 2023](#)). In this work, we simplify the task of image and video generation by using the exact autoregressive transformer architecture as in mainstream LLMs ([Radford et al., 2019](#)), over canonical and universal codecs: JPEG for images ([Wallace, 1991](#)), and AVC/H.264 for videos ([Wiegand et al., 2003](#)).

The key obstacle to training autoregressive models for image and video generation is *discretization*, as continuous data like images and videos need to be represented as discrete tokens. Current generative vision models that follow autoregressive language modeling objectives ([Bengio et al., 2000](#)) often adopt vector quantization (VQ) to encode

images or videos to some learned latent codes and then apply language models (Van Den Oord et al., 2017; Ramesh et al., 2021; Yu et al., 2021; Yan et al., 2021; Yu et al., 2023).¹ However, VQ methods often demand sophisticated tokenizer training that requires a careful hyperparameter selection for vision-specific modules (e.g., downsampling factor in convolutions) and balancing across several losses (Van Den Oord et al., 2017; Esser et al., 2021). VQ also involves a two-stage, non-end-to-end learning process (first the neural tokenizer, then the latent code LM). This makes downstream adaptation of the models less flexible (e.g., tuning the VQ tokenizer interferes with the learned latent code LM). Overall, the use of conventional LLM architectures (end-to-end autoregressive sequence modeling) as generative vision models is not yet straightforward.

The seminal work of ImageGPT (Chen et al., 2020) attempted to bridge this gap by using a regular GPT architecture to model pixels sequentially. They have shown a small-scale success at a very low resolution of 32x32 pixels. More realistic images at a size of 256x256 would require modeling a prohibitive amount of tokens in each sequence (65K or 196K tokens depending on color modes), not to mention videos. This hinders the method’s wider adoption by the field.

In this chapter, we tackle the problem of training LLM architectures for image and video generation where the essential discretization neither adds significant complications to the pipeline like VQ methods, nor is computationally prohibitively expensive like ImageGPT. Specifically, we use canonical file encodings/codecs—JPEG for images (Wallace, 1991), and AVC/H.264 for videos (Wiegand et al., 2003)—as non-neural preprocessors that discretize data. We show that codec-based representations greatly mitigate the sequence length limitation while being simple and effective. This design enables us to train a vanilla transformer with the conventional language modeling objective for image and video generation in a realistic setup.

We pretrain two 7B models with a Llama-2 architecture (Touvron et al., 2023b), named JPEG-LM and AvC-LM, that can generate 256x256 images and 256x144 videos

¹The other major line of generative vision models are diffusion models, a score-based, non-autoregressive method (Song and Ermon, 2019; Ho et al., 2020; Rombach et al., 2022; Peebles and Xie, 2023). Since the diffusion objectives are drastically different from the language modeling objective, it is challenging to integrate them in a multi-modal setup (e.g., with regular language models). While not a main focus of this work, we include comparisons with diffusion models in our later experiments as a secondary evaluation.

with 15 frames, with an average context length of 5K and 15K, respectively. In our main image modeling/generation evaluations, we show that JPEG-LM surpasses strong VQ-based models in generation quality (an average of 31% FID reduction) and produces surprisingly realistic qualitative examples. Our results also show AVC-LM can generate videos with realistic movements. Furthermore, we analyze in which aspects JPEG-LM is particularly stronger than VQ models and discover that our non-neural, training-free codec representations are more competent in capturing long-tail elements in images (e.g., human faces/eyes and text characters in small sizes).

Overall, this work presents how conventional LLM architectures can be used as generalized models towards visual generation. Our approach using canonical codecs does not incur vision-specific complications in the pipeline or suffer from sequence length infeasibility seen in prior work. Compared to the baselines, our models are much simpler to train and more effective. Following the previous efforts in unifying detached language-based tasks, our method helps pave the way to a unification of multiple modalities, facilitating the exploration of porting LLM techniques (e.g., alignment, scaling, efficiency, security, etc.) to all modalities.

6.1 Background

We explore autoregressive image generation as a straightforward extension of prominent LLM setups (Radford et al., 2019).² Conventional language modeling (Bengio et al., 2000) models the likelihood of sequential data autoregressively. Specifically, given a sequence of discrete tokens x_1, x_2, \dots, x_N (or $x_{1:N}$), a language model models $p(x_{1:N}) = \prod_{i=1}^N p(x_i | x_{1:i-1})$, an objective used in most mainstream LLMs. The key of applying language modeling to visual generation is how to discretize continuous data x like images and videos to discrete tokens $x_{1:N}$ like in language. Below we give an overview of two prominent approaches to the discretization of images.

²As a proof of concept, we mainly explore autoregressive modeling in visual generation only (images and videos, without text-conditioning), while future work may explore more diverse multi-modal setups.

6.1.1 Pixel values: ImageGPT

ImageGPT (Chen et al., 2020) is an image generation model based on a conventional LLM architecture (GPT-2). The images are discretized as a sequence of pixel values (integers 0–255) from the upper-left to the bottom-right pixel (raster scan). Since there are three channels of colors for each pixel, to reduce the number of tokens in each pixel sequence, ImageGPT clusters pixel colors to 512 distinctive clusters (i.e., for each pixel, three values from 0 to 255 are converted to one value from 0 to 511).

ImageGPT models the probability of pixel sequences autoregressively:

$$p(\text{pixel-value}(\mathbf{x})_i \mid \text{pixel-value}(\mathbf{x})_{1:i-1})$$

This is an expensive process, and ImageGPT only models and generates 32x32 images. Images with a more realistic resolution like 256x256 would require 65K tokens for each image (or 196K tokens without color clustering), a prohibitive sequence length for LLMs.

6.1.2 Latent codes: Vector-Quantization models

Vector-quantization (VQ) operates as a two-stage process, tokenizer training and language model training (Esser et al., 2021; Ramesh et al., 2021). We take VQ-VAE as our example tokenizer which discretizes continuous images (Van Den Oord et al., 2017). The tokenizer first learns an encoder E to project an image \mathbf{x} to spatial features $E(\mathbf{x})$. Then for each feature e in $E(\mathbf{x})$, it is quantized to $\hat{\mathbf{z}}$ by looking up the nearest neighbor in a learned codebook \mathcal{Z} :

$$\hat{\mathbf{z}} = \text{quantize}(E(\mathbf{x})) = [\text{argmin}_{\mathbf{z}_k \in \mathcal{Z}} \|e - \mathbf{z}_k\|_2^2]_{e \in E(\mathbf{x})}$$

The index k of the nearest entry in codebook \mathcal{Z} for each spatial feature forms the sequence of VQ latent codes. A decoder G is then learned to reconstruct the original image from the quantized representations. Overall, VQ-VAE learns an encoder E , decoder G , and codebook \mathcal{Z} , with three distinct losses: reconstruction loss, codebook loss, and commitment loss.

$$L_{\text{VQ-VAE}} = \|\mathbf{x} - G(\hat{\mathbf{z}})\|_1 + \|\text{sg}[E(\mathbf{x})] - \hat{\mathbf{z}}\|_2^2 + \beta \|\text{sg}[\hat{\mathbf{z}}] - E(\mathbf{x})\|_2^2$$

An effective VQ tokenizer needs a large amount of training data, proper hyperparameters for the vision-specific modules (e.g., downsampling factor in convolutional encoder

$E(\cdot)$), and a careful balance between the different losses (e.g., in L_{VQ-VAE}), which add significant complications to the pipeline.

A language model architecture can then be trained over the VQ latent codes (a sequence of index k above) as a generative vision model:

$$p(\text{VQ-code}(\mathbf{x})_i \mid \text{VQ-code}(\mathbf{x})_{1:i-1})$$

Notably, since the training of language model comes after and depends on the VQ tokenizer, a post-hoc update to the VQ tokenizer is challenging since it would lead to a non-trivial retraining or adaptation of the trained language model. Indeed in §6.4.3 we find that the VQ tokenizer, though trained with a large amount of data, still struggles with long-tail elements in the images and is hard to be optimized once and for all.

For simplicity and end-to-end adaptability, we propose to discretize continuous image and video data via canonical codecs — a collaboration between language models and multimodal compression algorithms.

6.2 JPEG-LM and AVC-LM

Though images and videos are continuous data and naturally have 2D or 3D data structures, they are stored as files on computers efficiently via compression/codecs, which leads to a discrete 1D representation. We aim to explore whether standard LLM architectures can directly learn to model and generate canonical vision file encodings, which can subsequently be read/opened as generated images or videos. Generation in this paradigm would greatly mitigate the sequence length infeasibility in ImageGPT while being simple and end-to-end trainable compared to VQ methods. Moreover, canonical file encodings/codecs are often non-neural and training-free and are robust to distributional shifts (§6.4.3). In this work, we choose the most popular and established file encodings/codecs for images and videos, JPEG (Wallace, 1991) and AVC/H.264 (Wiegand et al., 2003), respectively.³

³For images, PNG is also a common format. However, unlike the lossy JPEG, PNG is a lossless compression method (similar to ZIP) and often results in less effective compression and much longer sequences than JPEG.

6.2.1 Canonical codecs: JPEG and AVC/H.264

Canonical non-neural codecs like JPEG and AVC have a high-level intuition to compress signals that are less perceptible to human eyes more aggressively. JPEG has three main steps to encode each image: discrete cosine transform (DCT), quantization, and entropy coding. DCT converts each image patch to a weighted combination of a preset of patches containing low- and high-frequency patterns. Quantization zeroes out some high-frequency patterns from the weighted combination, since human eye is not good at perceiving them. Entropy encoding such as Huffman coding is then used to reduce the total numbers/bits representing the patches/images.⁴

AVC (H.264) operates on patches (macroblocks) of video frames. Each patch can be encoded using blocks of pixels that are already encoded within the current frame (intra-frame prediction) or using blocks of pixels encoded in other frames (inter-frame prediction with motion estimation). The prediction is then subtracted from the current patch to form a residual. The residual then goes through a process similar to JPEG, involving DCT, quantization, and bitstream encoding. The encoded content is a crucial part to the subsequent container files like MP4.

Both codecs have been used widely for decades and substantially compress the data (and thus sequence length) compared to raw pixel modeling (in our setup 40x in JPEG and 110x in AVC). Our focus is to use these canonical codecs as off-the-shelf tools to convert images and videos to sequences of discrete bytes efficiently.⁵ We wish to fit an LLM to implicitly learn the grammars and semantics of the canonical codecs.

6.2.2 JPEG-LM and AVC-LM

JPEG and AVC convert images and videos to bytes. Most of these bytes represent the image and video content after entropy encoding. However, there are also metadata and special patch/macroblock separators that are invariant across images or videos and use up multiple bytes. To address them along with other unknown frequent byte combinations that are compressed suboptimally by entropy encoding (e.g., by JPEG's

⁴A further intuitive and interactive description can be found at <https://parametric.press/issue-01/unraveling-the-jpeg/> (Shehata and Conlen, 2019).

⁵Both codecs operate at bits level at the core (due to entropy encoding), but modeling at bytes level is effective according to our experiments.

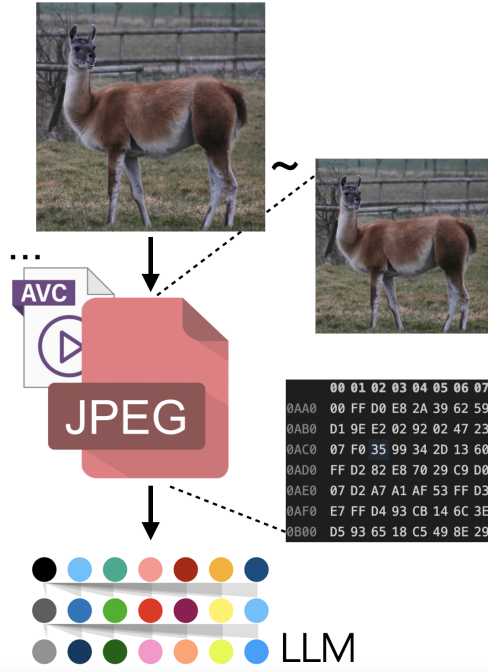


Figure 6.1: JPEG-LM and Avc-LM are simple autoregressive transformers that directly model and generate canonical file encodings.

standard, fixed Huffman tables), we further extend the default byte vocabulary (256 discrete values) *slightly* with byte-pair encoding (BPE), a standard preprocessing scheme in LLMs, which merges bytes appearing together frequently to a new single token.⁶ Since JPEG and AVC produce sequences of variable lengths based on the content of images and videos, special beginning-of-sequence and end-of-sequence tokens are also added to the vocabularies. The entries in the vocabularies are considered as our JPEG/AVC tokens.

Given an image x , we propose JPEG-LM to model:

$$p(\text{JPEG-token}(x)_i \mid \text{JPEG-token}(x)_{1:i-1})$$

Given a video x , we propose Avc-LM to model:

$$p(\text{AVC-token}(x)_i \mid \text{AVC-token}(x)_{1:i-1})$$

⁶More precisely, for the metadata/headers in the byte sequence that are well-known to be redundant across examples (e.g., JPEG quantization and Huffman tables), we remove them in the preprocessing and later add them back to the generated bytes from the model. For more complicated codecs like AVC, we let BPE handle such metadata.

We use conventional LLM architectures (autoregressive transformers) without any vision-specific modifications (no convolutions, no 2D positional embeddings) to maximize the models’ generality.

6.3 Experimental setup

6.3.1 JPEG-LM

We pretrain a 7B Llama-2 model (Touvron et al., 2023b) from scratch using 23M 256x256 images subsampled from Schuhmann et al. (2022). JPEG encodes each image with a quality factor of 25 (qualitative illustration in §6.4.3).⁷ We first use 10K images to derive 320 BPE tokens as our vocabulary entries.⁸ On average, each image in our training data leads to 5K tokens. For batching efficiency, we concatenate all sequences in the dataset and chunk in sequences of length 12K. In total, we have 9.5M sequences and thus 114B JPEG tokens (for each epoch). The model is trained approximately for two epochs with a maximum learning rate of 3e-4.

6.3.2 AVC-LM

As a proof of concept that canonical video codecs can be used for video generation as well, similar to JPEG-LM, a 7B Llama-2 model is pretrained from scratch as AvC-LM using 2M 256x144 videos subsampled from Bain et al. (2021). Due to the scope of experiments, we only keep the first 5 seconds of each video with 3 frame-per-second (thus 15 frames in total). The video is then processed with AVC/H.264 codec with a constant quantization parameter 37.⁹ We use 10K videos to derive 1024 BPE tokens as the vocabulary entries. On average, each video in our training data has 15K tokens. We perform data concatenation and chunk in context lengths of 32K for efficient batching. In total, we have 1.3M sequences and thus 42B AVC tokens.

⁷<https://pillow.readthedocs.io/>

⁸In our pilot study, we find the BPE process to be optional and the model would work similarly without it. The 64 extended vocabulary entries apart from the 256 default byte values include special JPEG separators FFD0, FFD1, . . . , FFD9, FFDA, and static file headers invariant across data, which slightly help reduce the sequence length. The vocabulary size 320 is chosen since a multiple of 64 for the embedding dimension is desired for optimal compute on GPUs.

⁹<https://ffmpeg.org/>

6.3.3 Image generation baselines

VQ transformer We use a pretrained VQ tokenizer from [Tang et al. \(2022\)](#), which used 200M images (ITHQ-200M, closed source dataset) to train a VQ-VAE model.¹⁰ This VQ tokenizer processes each image in the 23M image training set for our JPEG-LM (vocabulary size 4096, sequence length 1024). We then train a 7B Llama-2 transformer with the same configuration as in JPEG-LM. We use this VQ model as a main comparison to our JPEG-LM throughout this work.

ImageGPT + super-resolution ImageGPT uses GPT-2 XL as its underlying architecture. The pretrained model in ([Chen et al., 2020](#)) is trained over 14M 32x32 images from ImageNet. For a comparable evaluation, we use a super-resolution model ([Rombach et al., 2022](#)) over ImageGPT’s output.¹¹

Diffusion Though not a focus of this work, we include two variants of diffusion models in the baselines, Stable Diffusion (inpainting optimized) ([Rombach et al., 2022](#)) and VQ diffusion ([Gu et al., 2022](#); [Tang et al., 2022](#)). Both diffusion models can take partial images (through masking) and generate completed images, a setup we use across models in later evaluations. These baseline diffusion models are smaller in model size (~1B) but consume orders of magnitude more training data (200M–5B). They only serve as a secondary reference, and our focus is on comparing autoregressive image generation models under mainstream LLM paradigms.

6.4 Results

In works of language modeling, a fundamental evaluation is to collect a set of validation data, use the prefixes of data as prompts to the pretrained language model, and sample from the language model for a completion ([Holtzman et al., 2020](#); [Meister et al., 2023](#)).

¹⁰In our pilot study, we found this f8 VQ tokenizer outperforming other open-source VQ tokenizers, even the ones with longer context lengths (f4) like in [Rombach et al. \(2022\)](#). More discussion can be found in §6.6.1.

¹¹The pretrained model provides 4x super-resolution. In our pilot study, we find performing a 4x super-resolution, followed by a 0.5x downsample, then another 4x super-resolution yields the best result for the 32²-to-256² conversion.

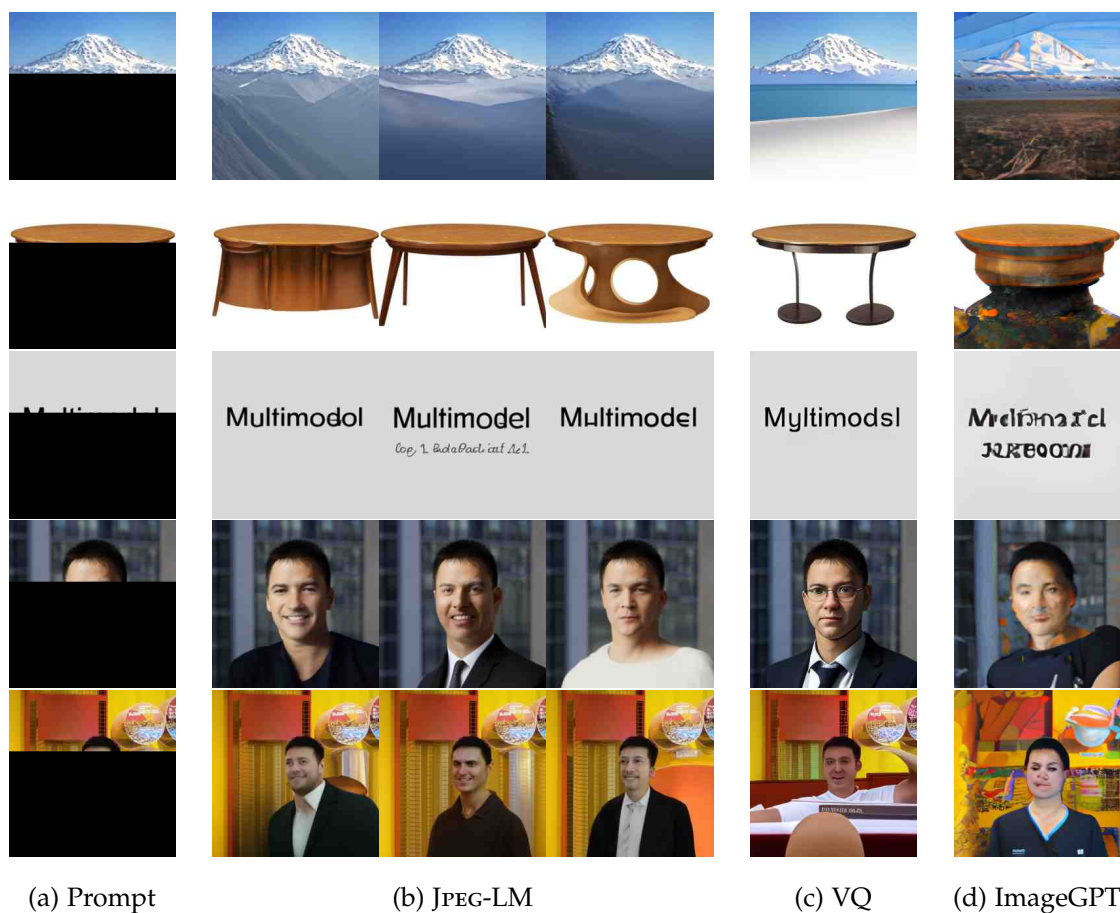


Figure 6.2: Generated images by JPEG-LM and baselines with partial images as prompts. We show three random samples from JPEG-LM and one from VQ transformer and ImageGPT (with super-resolution). The original images for the prompts are independently sourced outside existing training sets. We observe that JPEG-LM can generate realistic facial expressions, landscape, common objects, texts in image forms, etc. Additionally, JPEG-LM shows an especial advantage over baselines on meaningful details like human eyes (zoom in for the best view). Figure 6.6 and Figure 6.7 show further examples of JPEG-LM and VQ transformer on unconditional generation.

The completions are then evaluated for their quality against the gold validation data through distance metrics like Mauve score (Pillutla et al., 2021).

In this chapter, since we focus on vision-modality-only models with LLM architectures, we retain partial images (and later partial videos) as prompts to our models and evaluate their completions. Given a prompt ratio r_{prompt} , the autoregressive image

generation models condition on discretization $(\mathbf{x})_{1:(r_{\text{prompt}} \times N_{\text{tokens}})}$ for the generation.¹² Throughout the evaluations, the comparison between JPEG-LM and VQ transformer would be the most direct, as they share the same paradigm, model size, and training data (except that VQ transformer uses substantially more data in the tokenizer training stage).

6.4.1 Qualitative analysis

In Figure 6.2, we show the generation samples from JPEG-LM along with baseline models over independently sourced data outside existing training sets. We observe that by directly outputting JPEG file bytes, JPEG-LM can generate surprisingly realistic facial expressions (especially the eyes, compared to the strong VQ transformer), landscape, common objects, texts in image forms, etc. Figure 6.6 and Figure 6.7 show examples of JPEG-LM and VQ transformer on unconditional generation.

6.4.2 Quantitative results

In Table 6.1, we show prompting JPEG-LM, VQ transformer, and other baselines with different levels of partial images in ImageNet-1K (Russakovsky et al., 2015). The FID evaluation (Heusel et al., 2017) contains 5000 randomly sampled images from ImageNet-1K’s validation set. This is *zero-shot* generation (w.r.t. models’ training sets) and without class-conditioning. Experiments were done three times with different seeds. JPEG-LM consistently outperforms the VQ transformer in all prompting ratios. It mostly surpasses diffusion baselines with inpainting capabilities as well.

In Table 6.2, we show prompting the models with partial images in FFHQ (Karras et al., 2019). This is also a *zero-shot* setup without training to the FFHQ distribution and is evaluated on 1000 randomly sampled FFHQ images. JPEG-LM consistently outperforms the VQ transformer and other baselines.

In Table 6.3, we further validate our findings on fully unconditional generation with JPEG-LM and VQ transformer. Since they were trained on the same training data, we

¹²More specifically, the fixed-length VQ transformer and ImageGPT condition on discretization $(\mathbf{x})_{1:(r_{\text{prompt}} \times N_{\text{tokens}})}$ and generate discretization $(\mathbf{x})_{(r_{\text{prompt}} \times N_{\text{tokens}}):N_{\text{tokens}}}$. Variable-length JPEG-LM conditions on discretization $(\mathbf{x})_{1:\text{patch-position}(r_{\text{prompt}} \times N_{\text{patches}})}$ and generates until a EOS token is produced. Throughout the work, sampling from autoregressive transformers is by default with $\text{top-}p = \{0.9, 1.0\}$ and $\text{top-}k = \{40, 80\}$.

	$r_{\text{prompt}} = 0.25$	$r_{\text{prompt}} = 0.5$	$r_{\text{prompt}} = 0.75$
Stable Diffusion (inpaint)	266.71 (± 1.67)	132.98 (± 0.53)	58.17 (± 0.10)
VQ Diffusion	252.42 (± 0.20)	125.16 (± 0.26)	57.49 (± 0.25)
ImageGPT (super-resolution)	289.48 (± 0.61)	262.76 (± 0.48)	258.11 (± 0.69)
VQ Transformer	302.92 (± 0.29)	172.73 (± 0.21)	71.88 (± 0.19)
JPEG-LM	272.12 (± 1.24)	123.09 (± 0.28)	34.21 (± 0.21)

Table 6.1: Zero-shot, partial-image-conditioned, FID evaluation on **ImageNet-1K** (lower is better). r_{prompt} indicates the ratio of the image passed to the model as prompt. Best results among the autoregressive models are in bold fonts (reference diffusion results are italicized if better).

	$r_{\text{prompt}} = 0.375$	$r_{\text{prompt}} = 0.4375$	$r_{\text{prompt}} = 0.5$
Stable Diffusion (inpaint)	115.30 (± 2.14)	107.02 (± 1.83)	89.82 (± 4.51)
VQ Diffusion	60.88 (± 0.38)	45.63 (± 0.17)	40.58 (± 0.91)
ImageGPT (super-resolution)	61.73 (± 0.91)	57.80 (± 0.73)	55.28 (± 1.22)
VQ Transformer	53.25 (± 0.54)	45.58 (± 0.58)	41.15 (± 0.35)
JPEG-LM	36.15 (± 1.11)	31.22 (± 0.33)	27.15 (± 0.21)

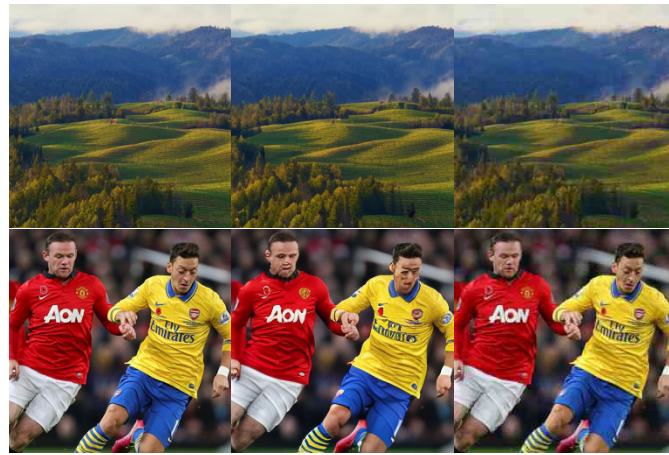
Table 6.2: Zero-shot, partial-image-conditioned, FID evaluation on **FFHQ** (lower is better). r_{prompt} indicates the ratio of the image passed to the model as prompt. Best results are in bold fonts. The prompting ratios in FFHQ were chosen differently such that they often lead to image prompts above the human eyes, below the eyes, and below the nose in pilot experiments.

VQ Transformer	155.51 (± 2.41)	JPEG-LM	121.35 (± 0.51)
----------------	-----------------------	---------	------------------------------

Table 6.3: Unconditional FID comparison of JPEG-LM and VQ transformer.

can compare their FID of unconditional generation w.r.t. our held-out, i.i.d. evaluation set. We again observe that JPEG-LM achieves a better FID.

These findings show JPEG-LM’s overall competence in image generation with a pure LLM architecture modeling canonical file encodings.



(a) Original (b) After VQ (c) After JPEG

Figure 6.3: Compression effect of VQ and JPEG (zoom in for the best view). JPEG is significantly better in detailed but highly perceptible elements like small human faces and text characters. VQ has a relative advantage in color and sharpness preservation.

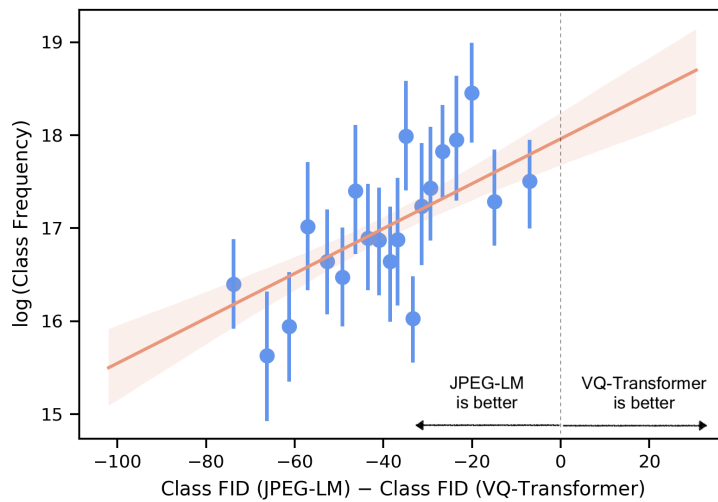


Figure 6.4: Correlation between per-class (ImageNet-1K) FID difference and class frequency. The class frequency is estimated through querying Google image search. Each class has a corresponding data point while an aggregation is performed for visual clarity. The correlation is positive and statistically significant ($p=0.0002$). This indicates JPEG-LM has more advantage in long-tail classes.¹³

6.4.3 Why JPEG-LM? A case study over long-tail elements in images

To further explore in which aspects our JPEG-LM excels compared to the baselines, especially the VQ transformer, we first compare how data is processed/compressed before being trained in transformers in JPEG-LM and VQ models.

JPEG vs. VQ compression JPEG-LM and VQ transformers can both be interpreted as first performing compression and then autoregressive modeling. The VQ model, unlike the non-neural JPEG compression, trained its VQ-VAE quantizer with a large amount of data (200M images in our case). In [Figure 6.3](#), we observe that both compression methods are relatively successful in compressing and decompressing general scenes like nature/landscape backgrounds. However, we find VQ suffers in small but highly perceptible elements in the images, like human faces or eyes. For images that contain small text characters, we observe the image degradation in VQ also happens in a non-predictable way, generating seemingly clear but uninterpretable text characters. On the other hand, the image degradation due to the non-neural, training-free JPEG compression happens in a predictable manner, arguably more preferable, especially when images contain long-tail elements with important meanings.

Quantitative analyses on long-tail elements In [Figure 6.4](#), we first show the per-class FID in our ImageNet-1K generation experiments. For each class of images, we calculate the difference between their FID with JPEG-LM generations and FID with the VQ transformer generations. We also estimate the frequency/coverage of each class of images on the internet by querying Google image search and logging the total number of returned results. We observe a statistically significant correlation between the per-class FID difference and the class frequency. The more advantage we observe in JPEG-LM over the VQ model, the less frequent the corresponding class is. In other words, JPEG-LM excels relatively more in long-tail sub-distributions.

In [Table 6.4](#), we further intervene on the FFHQ images by downsizing them (to 0.5x, while padding the images with black background to keep the overall size), aiming to test different models' performance on smaller visual concepts (e.g., small human

¹³To further corroborate our findings, apart from using Google image search, we also estimate the class frequency by matching class names/descriptions to the captions of our training images. The correlation is again positive and statistically significant (slope=+106.7, p -value=0.006).

	$r_{\text{prompt}} = 0.375$	$r_{\text{prompt}} = 0.5$
Stable Diffusion (IP)	136.28 (± 2.48)	120.54 (± 6.46)
$\Delta_{\text{downscaled-original}}$	+20.98	+30.72
VQ Diffusion	83.63 (± 1.16)	47.90 (± 1.12)
$\Delta_{\text{downscaled-original}}$	+22.75	+7.32
ImageGPT (SR)	46.67 (± 0.62)	40.46 (± 0.70)
$\Delta_{\text{downscaled-original}}$	-15.06	-14.82
VQ Transformer	56.33 (± 0.86)	47.94 (± 0.21)
$\Delta_{\text{downscaled-original}}$	+3.08	+6.79
JPEG-LM	35.80 (± 0.17)	26.25 (± 0.45)
$\Delta_{\text{downscaled-original}}$	-0.35	-0.90

Table 6.4: Zero-shot, partial-image-conditioned, FID evaluation on **downscaled FFHQ** (for both FID and Δ , lower is better). An increased gap between JPEG-LM and the VQ transformer shows JPEG-LM is more robust to small but meaningful long-tail elements.

faces). Such concepts, though small in size, can still be highly perceptible by humans and contain important meanings. We thus want the models to be robust on them. We perform similar prompted image generations with JPEG-LM, VQ transformer, and other baseline models.¹⁴ We find that JPEG-LM still consistently outperforms the VQ transformer (and other baselines as well). Especially, JPEG-LM achieves slightly better performance while VQ transformer becomes worse compared to the experiments with original image size. These deltas in opposite directions highlights the robustness of JPEG-LM.

These findings show that JPEG-LM not only has a promising performance overall, but specially strong with long-tail visual elements in the images.

6.4.4 Proof-of-concept video generation

One advantage of using canonical file encodings in LLM paradigms for vision generation is simplicity. From JPEG-LM that generates images, we naturally take one step

¹⁴The FID is measured on the active proportion of the images, excluding the black paddings.

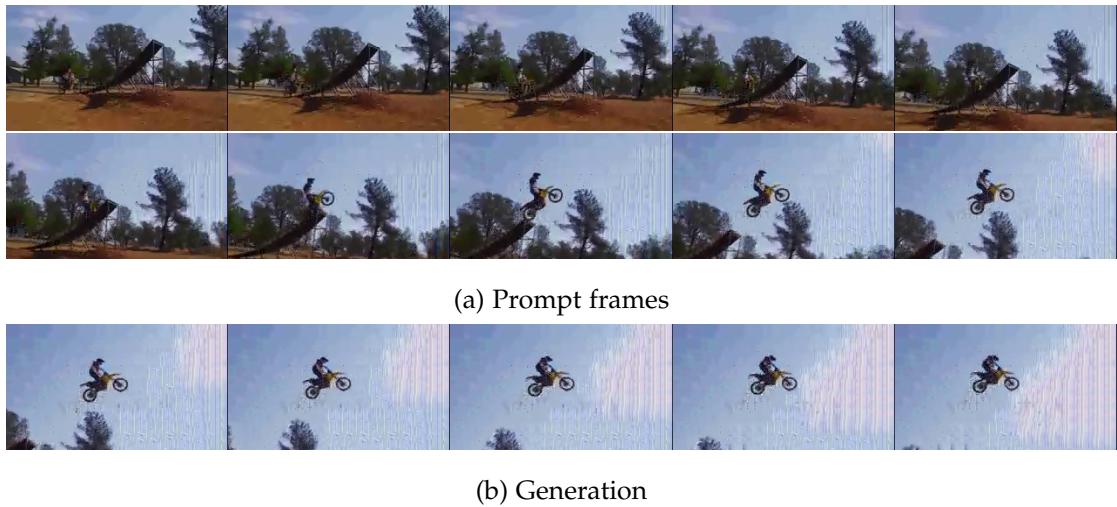


Figure 6.5: Generated video frames by Avc-LM on held-out test data. The first 10 frames are given to the model as the prompt, and the last 5 frames are generated by the model.

further and train a video generation model, Avc-LM, that models canonical video codecs (AVC/H.264) with autoregressive transformers.

As a proof of concept, we prompt Avc-LM with partial videos (i.e., frames) from a held-out set from our training data and investigate the model completions. In Figure 6.5 (along with §6.6.3), we show qualitative examples generated by Avc-LM. We observe that Avc-LM can capture the motion of moving objects reasonably.

6.5 Related Work

Current image and video generation models often adopt an autoregressive or diffusion approach. The autoregressive approach can build upon pixel-based representations as explored in Van Den Oord et al. (2016); Van den Oord et al. (2016); Chen et al. (2020). These methods suffer from prohibitively long sequences and only operate on low-resolution images. The autoregressive approach can also build upon vector quantization, which involves a sophisticated pre-hoc tokenizer training in addition to the autoregressive model (Van Den Oord et al., 2017; Esser et al., 2021; Ramesh et al., 2021; Yu et al., 2021; Yan et al., 2021; Yu et al., 2023; Mentzer et al., 2023; Lu et al., 2023; Liu et al., 2024a).

Diffusion models generate images or videos by an iterative denoising process, and they have specialized objectives and architectures that are challenging to be incorporated

to regular LLM paradigms to form multi-modal systems (Song and Ermon, 2019; Ho et al., 2020; Rombach et al., 2022; Ho et al., 2022; Gu et al., 2022; Tang et al., 2022; Gu et al., 2023; Peebles and Xie, 2023; Crowson et al., 2024). For example, performing simple tasks outside visual generation like classification with diffusion architectures is already not straightforward (Li et al., 2023a).

In this chapter, we propose to model canonical codecs (JPEG and AVC/H.264) with conventional language model architectures for visual generation. Horton et al. (2023) and Wu et al. (2024) are independent work that also process file bytes data, but they both focus on visual understanding (instead of generation) and use specialized modules to handle the byte sequences (whereas we use a general Llama-2 model). Perez et al. (2024) concurrently discover that JPEG formats can be used with language models in file anomaly handling and generation (on low-resolution images).

As a universal codec, JPEG is a novel form of data encoding for efficient image understanding (Park and Johnson, 2023). Kang et al. (2019) explore an image generation model that performs generation and JPEG compression in one system with GANs. JPEG artifacts can also be corrected by learning a restoration model (Kawar et al., 2022), which is potentially helpful to the generations from our JPEG-LM for improving image quality. Compressive codecs are also a rising topic in language. Jiang et al. (2023) use canonical compressors as feature extractors for texts. Lester et al. (2024) train language models to generate compressed texts.

6.6 Conclusion and continued discussion

In this chapter, we propose JPEG-LM and AVC-LM that generate images and videos using mainstream LLM architectures (autoregressive transformers) with canonical codec representations (JPEG for images, AVC/H.264 for videos). Our approach greatly mitigates the length infeasibility of pixel-based sequence modeling while enabling simple, flexible, and end-to-end training compared to sophisticated vector quantization methods. Our image generation evaluation shows JPEG-LM achieves better results than the baselines, with an especial advantage in generating long-tail visual elements. Our work contributes to a unifying paradigm of language generation and visual generation,

facilitating future research to port successful LLM techniques (e.g., alignment, efficiency, etc.) to all modalities.

One notable significance of this work is to show that vanilla autoregressive language modeling with canonical codecs is *indeed possible* in visual generation, an approach almost void of prior work. Based on the findings of this work, future work may continue to investigate the scaling aspect of this family of models (similar to mainstream LLMs), co-training/deployment with text-based LLMs, or better architectures for canonical codecs without loss of generality for other modalities.

6.6.1 Design discussion

Our work focuses on the challenging task of visual generation (e.g., outputting images) rather than visual understanding (e.g., inputting images, outputting classes or texts). In the field of visual understanding, the encoding of images has less restricted forms. For example, [Bavishi et al. \(2023\)](#) and [El-Nouby et al. \(2024\)](#) linearly project image patches as inputs to the transformers, [Liu et al. \(2024b\)](#) pass CLIP embeddings ([Radford et al., 2021](#)) to language models, etc. However, these image encoding formulations are not applicable to image generation. Though not a focus in this work, future work may extend our JPEG-LM and AvC-LM that share the same underlying architectures with regular language models to image and video understanding scenarios.

Compared to raw pixel modeling that would represent a 256x256 image with 65K or 196K tokens (depending on color modes), using canonical codecs like JPEG substantially reduces the sequence length to 5K on average. In terms of sequence length, the VQ transformers are usually more aggressive, representing each image with 1K tokens. It is notable that this an ideal hyperparameter discovered in prior work that helps model global structures—increasing the number of tokens in VQ (thus reducing the downsampling patch size) may lead to degenerated results rather than helping the model learn with more capacity ([Esser et al., 2021](#)). Our work proposes to model canonical codecs as a proof of concept, and future work may compare with more VQ setups or further improve the context efficiency of JPEG-LM.

Theoretically, it is possible for JPEG-LM to generate corrupted binary JPEG data (like generating grammatically incorrect language in NLP). The JPEG decoding algorithm would still be able to decode corrupted data, where the patch containing invalid data is

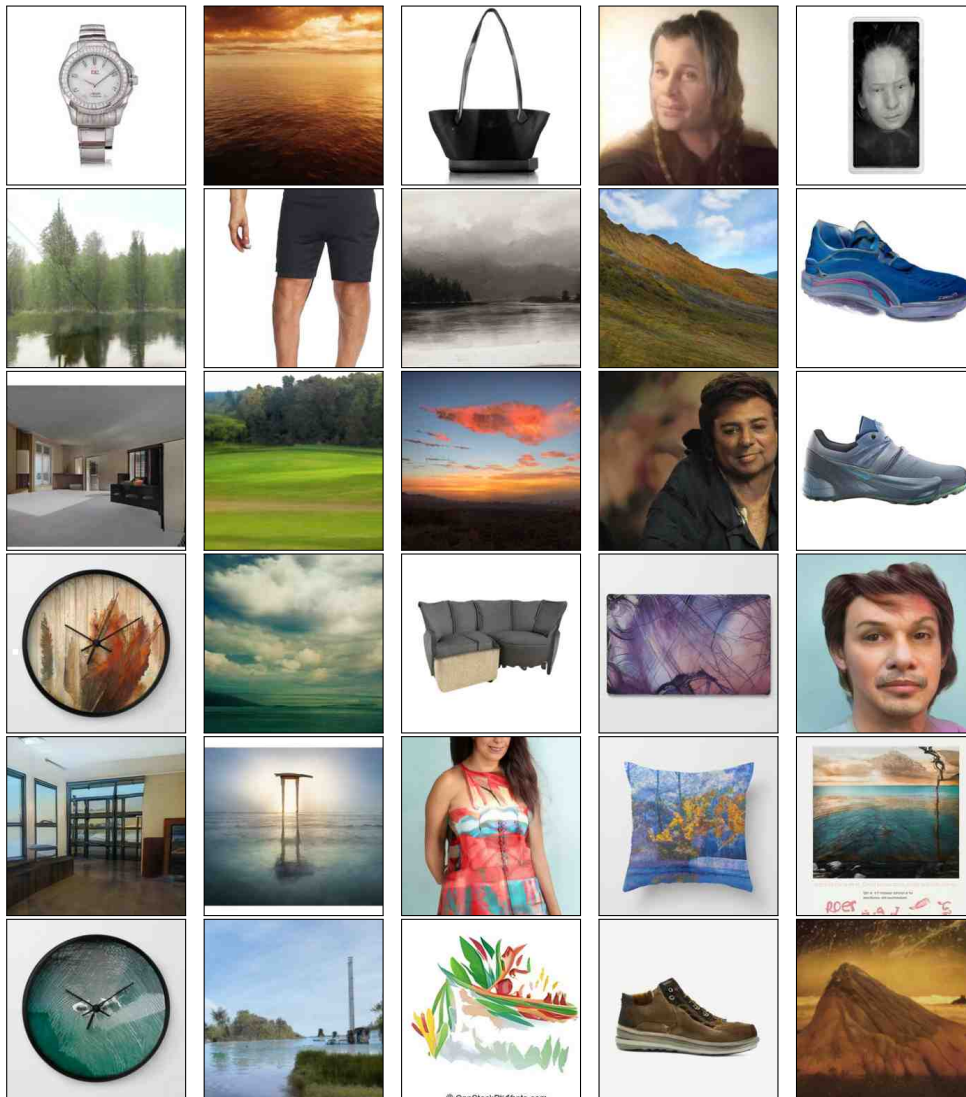


Figure 6.6: Unconditional generation by JPEG-LM.

shown with obviously broken color or a fully gray/chessboard-like pattern. We almost never observe such corruption in the generations from our trained JPEG-LM (at the very early stage of model training, we already stopped seeing corrupted generations, indicating the model learns valid JPEG grammar first and then the underlying semantics). In the rare cases where corrupted binary data is generated, such corrupted data is included and reflected in our overall evaluation results as well.

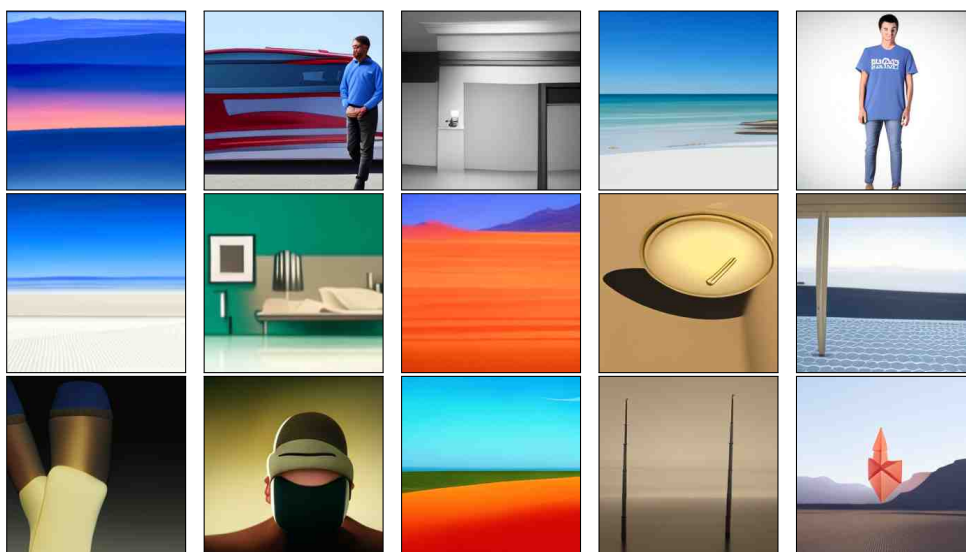


Figure 6.7: Unconditional generation by VQ transformer.

6.6.2 More qualitative examples from JPEG-LM

In Figure 6.8, we show more JPEG-LM completions on partial images from FFHQ (zero-shot). Figure 6.6 and Figure 6.7 show further examples of JPEG-LM and VQ transformer on unconditional generation.

6.6.3 More qualitative examples from AVC-LM

More generations from AvC-LM can be found in Figure 6.9, Figure 6.10, Figure 6.11, and Figure 6.12. Similar to Figure 6.5, we observe realistic object movements such as flags, clouds, clocks, cars on the street, etc.

6.6.4 Detailed configurations for the canonical codecs

Our JPEG encoding uses the pillow package. We specifically encode each image with: `image.save(format='JPEG', quality=25, subsampling="4:2:0", streamtype=2, restart_marker_blocks=1)`. More details about these arguments can be found at <https://pillow.readthedocs.io/en/stable/handbook/image-file-formats.html#jpeg-saving>. Our AVC/H.264 encoding uses the ffmpeg package. Specifically, the configurations/commands we used are: `ffmpeg -vf "fps=3,scale=256:144:force_original_aspect_ratio=decrease, pad=256:144:(ow-ih)/2:(oh-ih)/2" -t 5 -c:v libx264 -pix_fmt yuv420p -profile:v baseline -qp 37 -bf 0 -an -sn -x264opts "slice-max-mbs=1" -trellis 0 -me_method dia -threads 1 -subq 0 -psy 0`

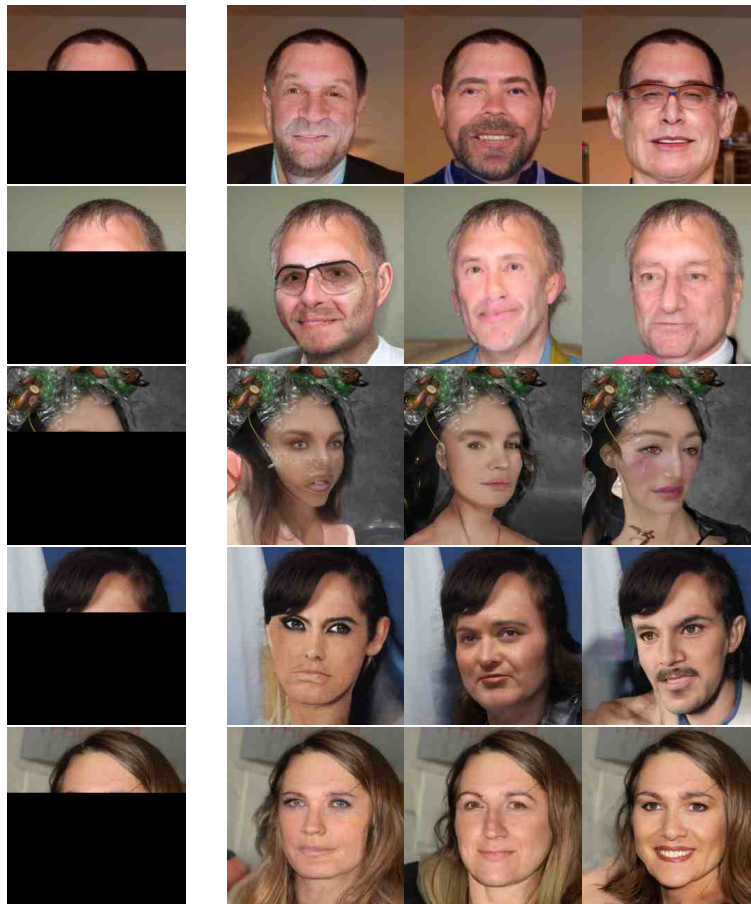
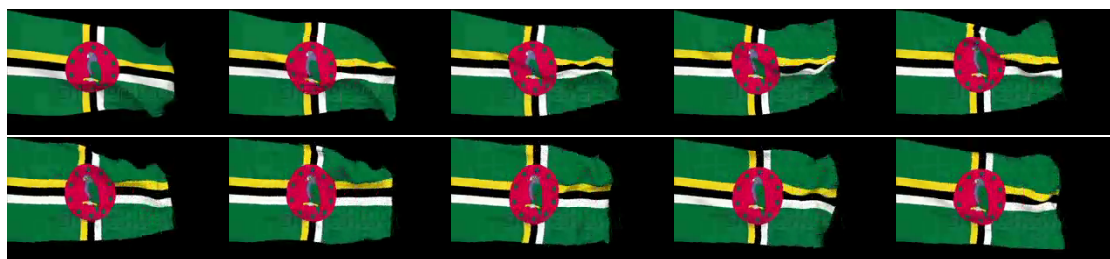


Figure 6.8: Generated images by JPEG-LM with partial FFHQ images as prompts (*zero-shot* generation). Similar to Figure 6.2, the generated facial expressions are modelled as JPEG bytes and mostly look realistic.

-mixed-refs 0 -fast-pskip 0 -partitions none -refs 3 -bsf:v h264_mp4toannexb. More details about these flags can be found at <https://ffmpeg.org/ffmpeg.html>.

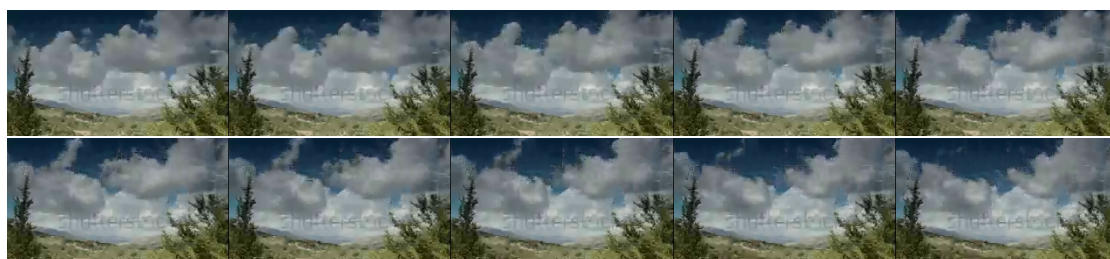


(a) Prompt frames

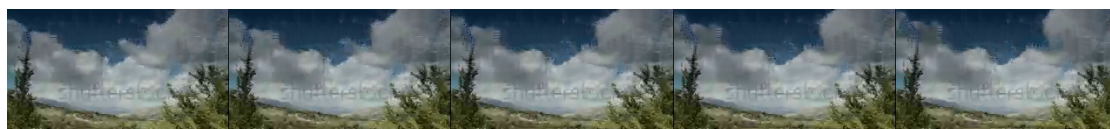


(b) Generation

Figure 6.9: Generated video frames by Avc-LM on held-out test data. The first 10 frames are given to the model as the prompt, and the last 5 frames are generated by the model.

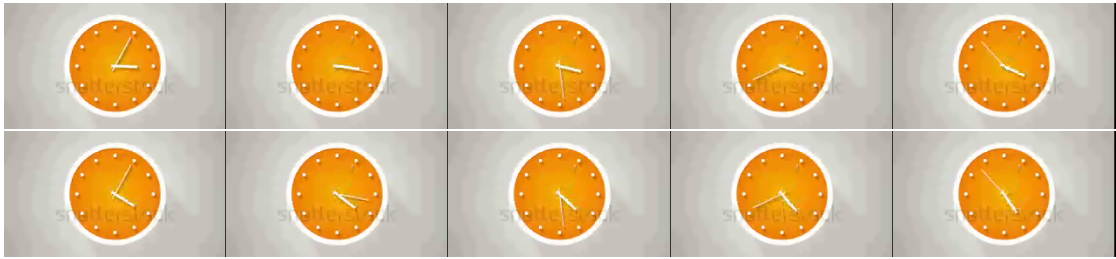


(a) Prompt frames



(b) Generation

Figure 6.10: Generated video frames by Avc-LM on held-out test data. The first 10 frames are given to the model as the prompt, and the last 5 frames are generated by the model.

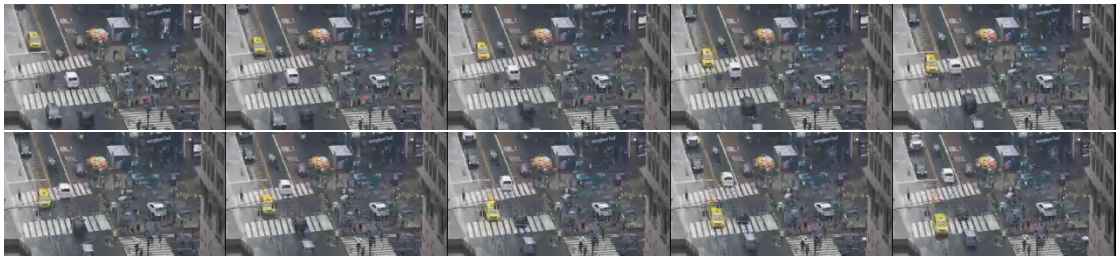


(a) Prompt frames



(b) Generation

Figure 6.11: Generated video frames by Avc-LM on held-out test data. The first 10 frames are given to the model as the prompt, and the last 5 frames are generated by the model.



(a) Prompt frames



(b) Generation

Figure 6.12: Generated video frames by Avc-LM on held-out test data. The first 10 frames are given to the model as the prompt, and the last 5 frames are generated by the model.

Chapter 7

Conclusions

7.1 Summary of Contributions

- I develop SSD-LM (§3) — a diffusion-based language model with two key designs. SSD-LM is *semi-autoregressive*, iteratively generating blocks of text, allowing for flexible output length at decoding time while enabling local bidirectional context updates. SSD-LM is *simplex-based*, performing diffusion on the natural vocabulary space rather than a learned latent space, allowing us to incorporate classifier guidance and modular control using off-the-shelf classifiers without any adaptation. SSD-LM is evaluated on unconstrained text generation as well as controllable text generation benchmarks. I show SSD-LM is a strong language model that can collaborate with discriminative attribute classifiers at inference time without any tuning.
- I develop CAD (§4) — a context-aware decoding algorithm which samples text using two language models in collaboration to mitigate LLM hallucination. Without any modifications to the pretrained models' weights, the two language models are provided with different inputs, with and without the key context the system should focus on. The decoding follows a contrastive output distribution that amplifies the difference between the output probabilities when used with and without context. On summarization and knowledge editing tasks, I show CAD significantly improves LLMs' faithfulness to the given context and reduces model hallucination.

- I develop SSD-2 (§5) — an approach to ensemble at inference time a large general-purpose diffusion LLM with smaller but specialized and contextualized diffusion LLMs. As a part of SSD-2, I pretrained and instruction-tuned a diffusion LLM with 13B parameters. SSD-2 facilitates a weak-strong collaboration between this large diffusion LLM and 100x smaller models that can be customized and deployed by individual users. I show that compared to autoregressive language models, the collaboration between diffusion LLMs is more effective, leading to higher-quality model outputs due to their ability to dynamically incorporate bi-directional contexts.
- I develop JPEG-LM (§6) — an autoregressive language model architecture that generates images and videos as compressed files saved on computers via canonical codecs such as JPEG and AVC. JPEG-LM shows strong performance in image generation evaluations and especially excels in generating long-tail visual elements. This collaboration between a neural LLM architecture and non-neural, canonical compression codecs lowers the barriers between language generation and visual generation, driving progress toward universal multimodal LLMs.

7.2 Future Work

- **Multimodal world models.** LLMs understand and convey worlds through and within the space of language. However, its generative paradigm of predicting the future, synthesizing the unknowns, and aligning with human’s intents and values should not be limited to language. JPEG-LM (§6) lowers the barrier between generating language and generating images and videos by a neural tokenizer-free approach and a generalist LLM framework. The exploration of multimodal models that understand and synthesize the world (Ha and Schmidhuber, 2018; Hafner et al., 2019; Yang et al., 2023; Hu et al., 2023; Bruce et al., 2024; Garrido et al., 2024; Zhou et al., 2024; Shi et al., 2024a) can be essential for machine intelligence to learn knowledge from more comprehensive sources and engage in more constructive interactions with humans.
- **Diffusion-oriented LLMs.** Language is produced autoregressively by default, but research in diffusion LLMs has demonstrated that non-autoregressive language

generation is feasible. SSD-LM and SSD-2 (§3 and §5) show that diffusion-based LLMs can perform not only on-par with autoregressive LLMs but substantially improve the controllability of the LLMs. More recent research in the area explores the training and inference efficiency of diffusion LLMs, making them a strong alternative to standard autoregressive LLMs in production settings (Gulrajani and Hashimoto, 2023; Lou et al., 2023; Inception, 2025).

- **Data selection and generation.** Though this thesis primarily focuses on the modeling perspective of LLMs, training data is another crucial factor for LLMs' effectiveness. Many desired behaviors of LLMs such as in-context learning and alignment to human intents and values emerge because of the training data. As we are approaching the limit of explicit language data we could obtain online, identifying and using a subset of more important data is desired from both the interpretability and generalizability perspectives. Though not presented in this thesis, my other past works also have a focus on training data attribution and selection (Han et al., 2020; Han and Tsvetkov, 2020, 2021, 2022; Han et al., 2023b). Furthermore, generating new synthetic data for LLM alignment and reasoning has been and will continue to be a crucial step in advancing machine intelligence (Wang et al., 2022a; Sun et al., 2023; Yasunaga et al., 2024; OpenAI, 2024; Guo et al., 2025).

References

- David H Ackley, Geoffrey E Hinton, and Terrence J Sejnowski. 1985. A learning algorithm for boltzmann machines. *Cognitive science*, 9(1):147–169.
- J. Atchison and S.M. Shen. 1980. [Logistic-normal distributions:Some properties and uses](#). *Biometrika*, 67(2):261–272.
- Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. 2021. Structured denoising diffusion models in discrete state-spaces. In *Proc. NeurIPS*.
- Max Bain, Arsha Nagrani, Gül Varol, and Andrew Zisserman. 2021. Frozen in time: A joint video and image encoder for end-to-end retrieval. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1728–1738.
- József Bakosi and J. Raymond Ristorcelli. 2013. A stochastic diffusion process for the dirichlet distribution. *arXiv: Mathematical Physics*.
- Nikita Balagansky and Daniil Gavrilov. 2023. Democratized diffusion language model. *ArXiv*, abs/2305.10818.
- Yogesh Balaji, Seungjun Nah, Xun Huang, Arash Vahdat, Jiaming Song, Qinsheng Zhang, Karsten Kreis, Miika Aittala, Timo Aila, Samuli Laine, Bryan Catanzaro, Tero Karras, and Ming-Yu Liu. 2022. ediff-i: Text-to-image diffusion models with an ensemble of expert denoisers. *ArXiv*, abs/2211.01324.
- Francesco Barbieri, Jose Camacho-Collados, Luis Espinosa Anke, and Leonardo Neves. 2020. Tweeteval: Unified benchmark and comparative evaluation for tweet classification. In *Findings of EMNLP*.
- Rohan Bavishi, Erich Elsen, Curtis Hawthorne, Maxwell Nye, Augustus Odena, Arushi Somani, and Sağnak Taşlılar. 2023. [Introducing our multimodal models](#).
- Yoshua Bengio, Réjean Ducharme, and Pascal Vincent. 2000. A neural probabilistic language model. *Advances in neural information processing systems*, 13.
- Stella Biderman, Hailey Schoelkopf, Quentin Anthony, Herbie Bradley, Kyle O’Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar van der Wal. 2023. [Pythia: A suite for analyzing large language models across training and scaling](#).
- Sid Black, Gao Leo, Phil Wang, Connor Leahy, and Stella Biderman. 2021. [GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow](#).
- Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, T. J. Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeff Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *ArXiv*, abs/2005.14165.

- Jake Bruce, Michael D Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, et al. 2024. Genie: Generative interactive environments. In *Forty-first International Conference on Machine Learning*.
- Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. 2023. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*.
- Collin Burns, Pavel Izmailov, Jan Hendrik Kirchner, Bowen Baker, Leo Gao, Leopold Aschenbrenner, Yining Chen, Adrien Ecoffet, Manas Joglekar, Jan Leike, et al. 2023. Weak-to-strong generalization: Eliciting strong capabilities with weak supervision. *arXiv preprint arXiv:2312.09390*.
- Shuyang Cao and Lu Wang. 2021. [CLIFF: Contrastive learning for improving faithfulness and factuality in abstractive summarization](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6633–6649, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Ziqiang Cao, Furu Wei, Wenjie Li, and Sujian Li. 2018. [Faithful to the original: Fact aware neural abstractive summarization](#). In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 4784–4791. AAAI Press.
- Alvin Chan, Yew-Soon Ong, Bill Pung, Aston Zhang, and Jie Fu. 2021. Cocon: A self-supervised approach for controlled text generation. In *Proc. ICLR*.
- Stephanie Chan, Adam Santoro, Andrew Lampinen, Jane Wang, Aaditya Singh, Pierre Richemond, James McClelland, and Felix Hill. 2022. [Data distributional properties drive emergent in-context learning in transformers](#). In *Advances in Neural Information Processing Systems*, volume 35, pages 18878–18891. Curran Associates, Inc.
- Hung-Ting Chen, Michael Zhang, and Eunsol Choi. 2022a. [Rich knowledge sources bring complex knowledge conflicts: Recalibrating models to reflect conflicting evidence](#). In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 2292–2307, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.
- Jiaao Chen, Aston Zhang, Mu Li, Alexander J. Smola, and Diyi Yang. 2023. A cheaper and better diffusion language model with soft-masked noise. *ArXiv*, abs/2304.04746.
- Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. 2020. Generative pretraining from pixels. In *International conference on machine learning*, pages 1691–1703. PMLR.
- Ting Chen, Ruixiang Zhang, and Georj E. Hinton. 2022b. Analog bits: Generating discrete data using diffusion models with self-conditioning. *ArXiv*, abs/2208.04202.

- Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. [Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality](#).
- Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek B Rao, Parker Barnes, Yi Tay, Noam M. Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Benton C. Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier García, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayanan Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Díaz, Orhan Firat, Michele Catasta, Jason Wei, Kathleen S. Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2022. Palm: Scaling language modeling with pathways. *ArXiv*, abs/2204.02311.
- Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022. [Scaling instruction-finetuned language models](#). *ArXiv preprint*, abs/2210.11416.
- Katherine Crowson, Stefan Andreas Baumann, Alex Birch, Tanishq Mathew Abraham, Daniel Z Kaplan, and Enrico Shippole. 2024. Scalable high-resolution pixel-space image synthesis with hourglass diffusion transformers. *arXiv preprint arXiv:2401.11605*.
- Antonio R Damasio and Hanna Damasio. 1992. Brain and language. *Scientific American*, 267(3):88–109.
- Databricks. 2023. Databricks dolly 15k. <https://huggingface.co/datasets/databricks/databricks-dolly-15k>.
- Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. 2020. Plug and play language models: A simple approach to controlled text generation. In *Proc. ICLR*.
- Joe Davison, Joshua Feldman, and Alexander Rush. 2019. [Commonsense knowledge mining from pretrained models](#). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1173–1178, Hong Kong, China. Association for Computational Linguistics.

- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proc. NAACL-HLT*.
- Prafulla Dhariwal and Alex Nichol. 2021. Diffusion models beat gans on image synthesis. *ArXiv*, abs/2105.05233.
- Sander Dieleman, Laurent Sartran, Arman Roshannai, Nikolay Savinov, Yaroslav Ganin, Pierre H. Richemond, A. Doucet, Robin Strudel, Chris Dyer, Conor Durkan, Curtis Hawthorne, Rémi Leblond, Will Grathwohl, and Jonas Adler. 2022. Continuous diffusion for categorical data. *ArXiv*, abs/2211.15089.
- Yue Dong, Shuohang Wang, Zhe Gan, Yu Cheng, Jackie Chi Kit Cheung, and Jingjing Liu. 2020. **Multi-fact correction in abstractive text summarization**. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9320–9331, Online. Association for Computational Linguistics.
- Alaaeldin El-Nouby, Michal Klein, Shuangfei Zhai, Miguel Angel Bautista, Alexander Toshev, Vaishaal Shankar, Joshua M Susskind, and Armand Joulin. 2024. Scalable pre-training of large autoregressive image models. *arXiv preprint arXiv:2401.08541*.
- Patrick Esser, Robin Rombach, and Bjorn Ommer. 2021. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12873–12883.
- Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In *Proc. ACL*.
- Shangbin Feng, Vidhisha Balachandran, Yuyang Bai, and Yulia Tsvetkov. 2023. **FactKB: Generalizable factuality evaluation using language models enhanced with factual knowledge**. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 933–952, Singapore. Association for Computational Linguistics.
- Zhidan Feng, Zhenyu Zhang, Xintong Yu, Yewei Fang, Lanxin Li, Xuyi Chen, Yuxiang Lu, Jiaxiang Liu, Weichong Yin, Shi Feng, Yu Sun, Hao Tian, Hua Wu, and Haifeng Wang. 2022. Ernie-vilg 2.0: Improving text-to-image diffusion model with knowledge-enhanced mixture-of-denoising-experts. *ArXiv*, abs/2210.15257.
- Zhujin Gao, Junliang Guo, Xuejiao Tan, Yongxin Zhu, Fang Zhang, Jiang Bian, and Linli Xu. 2022. Difformer: Empowering diffusion model on embedding space for text generation. *ArXiv*, abs/2212.09412.
- Quentin Garrido, Mahmoud Assran, Nicolas Ballas, Adrien Bardes, Laurent Najman, and Yann LeCun. 2024. Learning and leveraging world models in visual representation learning. *arXiv preprint arXiv:2403.00504*.
- Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. 2019. Mask-predict: Parallel decoding of conditional masked language models. In *Proc. EMNLP*.
- Aaron Gokaslan and Vanya Cohen. 2019. Openwebtext corpus.

- Shansan Gong, Mukai Li, Jiangtao Feng, Zhiyong Wu, and Lingpeng Kong. 2022. Diffuseq: Sequence to sequence text generation with diffusion models. *ArXiv*, abs/2210.08933.
- Tanya Goyal and Greg Durrett. 2021. [Annotating and modeling fine-grained factuality in summarization](#). In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1449–1462, Online. Association for Computational Linguistics.
- Jiatao Gu, James Bradbury, Caiming Xiong, Victor OK Li, and Richard Socher. 2018. Non-autoregressive neural machine translation. In *Proc. ICLR*.
- Jiatao Gu, Changan Wang, and Jake Zhao. 2019. Levenshtein transformer. In *Proc. NeurIPS*.
- Jiatao Gu, Shuangfei Zhai, Yizhe Zhang, Joshua M Susskind, and Navdeep Jaitly. 2023. Matryoshka diffusion models. In *The Twelfth International Conference on Learning Representations*.
- Shuyang Gu, Dong Chen, Jianmin Bao, Fang Wen, Bo Zhang, Dongdong Chen, Lu Yuan, and Baining Guo. 2022. Vector quantized diffusion model for text-to-image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10696–10706.
- Ishaan Gulrajani and Tatsunori B Hashimoto. 2023. Likelihood-based diffusion language models. *Advances in Neural Information Processing Systems*, 36:16693–16715.
- Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.
- Meiqi Guo, Rebecca Hwa, and Adriana Kovashka. 2023. [Decoding symbolism in language models](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3311–3324, Toronto, Canada. Association for Computational Linguistics.
- Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don’t stop pretraining: Adapt language models to domains and tasks. In *Proc. ACL*.
- David Ha and Jürgen Schmidhuber. 2018. World models. *arXiv preprint arXiv:1803.10122*.
- Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. 2019. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*.
- Xiaochuang Han, Marjan Ghazvininejad, Pang Wei Koh, and Yulia Tsvetkov. 2024. Jpeg-lm: Llms as image generators with canonical codec representations. *arXiv preprint arXiv:2408.08459*.
- Xiaochuang Han, Sachin Kumar, and Yulia Tsvetkov. 2022. Ssd-lm: Semi-autoregressive simplex-based diffusion language model for text generation and modular control. *ArXiv*, abs/2210.17432.

- Xiaochuang Han, Sachin Kumar, Yulia Tsvetkov, and Marjan Ghazvininejad. 2023a. David helps goliath: Inference-time collaboration between small specialized and large general diffusion lms. *arXiv preprint arXiv:2305.14771*.
- Xiaochuang Han, Daniel Simig, Todor Mihaylov, Yulia Tsvetkov, Asli Celikyilmaz, and Tianlu Wang. 2023b. Understanding in-context learning via supportive pretraining data. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12660–12673.
- Xiaochuang Han and Yulia Tsvetkov. 2020. Fortifying toxic speech detectors against veiled toxicity. In *Proc. EMNLP*.
- Xiaochuang Han and Yulia Tsvetkov. 2021. Influence tuning: Demoting spurious correlations via instance attribution and instance-driven updates. In *Proc. Findings of EMNLP*.
- Xiaochuang Han and Yulia Tsvetkov. 2022. Orca: Interpreting prompted language models via locating supporting data evidence in the ocean of pretraining data. *arXiv preprint arXiv:2205.12600*.
- Xiaochuang Han, Byron C. Wallace, and Yulia Tsvetkov. 2020. Explaining black box predictions and unveiling data artifacts through influence functions. In *Proc. ACL*.
- Zhengfu He, Tianxiang Sun, Kuan Wang, Xuanjing Huang, and Xipeng Qiu. 2022. Diffusionbert: Improving generative masked language models with diffusion models. *ArXiv*, abs/2211.15029.
- Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. 2017. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30.
- Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851.
- Jonathan Ho and Tim Salimans. 2021. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*.
- Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. 2022. Video diffusion models. *Advances in Neural Information Processing Systems*, 35:8633–8646.
- Cong Duy Vu Hoang, Gholamreza Haffari, and Trevor Cohn. 2017. [Towards decoding as continuous optimisation in neural machine translation](#). In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 146–156, Copenhagen, Denmark. Association for Computational Linguistics.
- Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *International Conference on Learning Representations*.
- Ari Holtzman, Peter West, Vered Shwartz, Yejin Choi, and Luke Zettlemoyer. 2021. [Surface form competition: Why the highest probability answer isn’t always right](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7038–7051, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

- Emiel Hoogeboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. 2021. Argmax flows and multinomial diffusion: Learning categorical distributions. In *Proc. NeurIPS*.
- Maxwell Horton, Sachin Mehta, Ali Farhadi, and Mohammad Rastegari. 2023. Bytes are all you need: Transformers operating directly on file bytes. *arXiv preprint arXiv:2306.00238*.
- Anthony Hu, Lloyd Russell, Hudson Yeo, Zak Murez, George Fedoseev, Alex Kendall, Jamie Shotton, and Gianluca Corrado. 2023. Gaia-1: A generative world model for autonomous driving. *arXiv preprint arXiv:2309.17080*.
- Inception. 2025. [Introducing mercury, the first commercial-scale diffusion large language model](#).
- Zhiying Jiang, Matthew Yang, Mikhail Tsirlin, Raphael Tang, Yiqin Dai, and Jimmy Lin. 2023. “low-resource” text classification: A parameter-free classification method with compressors. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 6810–6828.
- Lukasz Kaiser, Samy Bengio, Aurko Roy, Ashish Vaswani, Niki Parmar, Jakob Uszkoreit, and Noam Shazeer. 2018. Fast decoding in sequence models using discrete latent variables. In *Proc. ICML*, pages 2390–2399. PMLR.
- Eric R Kandel, James H Schwartz, Thomas M Jessell, Steven Siegelbaum, A James Hudspeth, Sarah Mack, et al. 2000. *Principles of neural science*, volume 4. McGraw-hill New York.
- Byeongkeun Kang, Subarna Tripathi, and Truong Q Nguyen. 2019. Toward joint image generation and compression using generative adversarial networks. *arXiv preprint arXiv:1901.07838*.
- Daniel Kang and Tatsunori B. Hashimoto. 2020. [Improved natural language generation via loss truncation](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 718–731, Online. Association for Computational Linguistics.
- Tero Karras, Samuli Laine, and Timo Aila. 2019. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4401–4410.
- Bahjat Kawar, Jiaming Song, Stefano Ermon, and Michael Elad. 2022. Jpeg artifact correction using denoising diffusion restoration models. *arXiv preprint arXiv:2209.11888*.
- Nitish Shirish Keskar, Bryan McCann, Lav R Varshney, Caiming Xiong, and Richard Socher. 2019. Ctrl: A conditional transformer language model for controllable generation. *arXiv preprint arXiv:1909.05858*.
- Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2020. [Generalization through memorization: Nearest neighbor language models](#). In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

- Daniel King, Zejiang Shen, Nishant Subramani, Daniel S. Weld, Iz Beltagy, and Doug Downey. 2022. [Don't say what you don't know: Improving the consistency of abstractive summarization by constraining beam search](#). In *Proceedings of the 2nd Workshop on Natural Language Generation, Evaluation, and Metrics (GEM)*, pages 555–571, Abu Dhabi, United Arab Emirates (Hybrid). Association for Computational Linguistics.
- Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. 2021. Diffwave: A versatile diffusion model for audio synthesis. In *Proc. ICLR*.
- Ben Krause, Akhilesh Deepak Gotmare, Bryan McCann, Nitish Shirish Keskar, Shafiq Joty, Richard Socher, and Nazneen Fatema Rajani. 2021. Gedi: Generative discriminator guided sequence generation. In *Proc. Findings of EMNLP*.
- Wojciech Kryscinski, Bryan McCann, Caiming Xiong, and Richard Socher. 2020. [Evaluating the factual consistency of abstractive text summarization](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9332–9346, Online. Association for Computational Linguistics.
- Sachin Kumar, Eric Malmi, Aliaksei Severyn, and Yulia Tsvetkov. 2021. [Controlled text generation as continuous optimization with multiple constraints](#). In *Proc. NeurIPS*.
- Sachin Kumar, Biswajit Paria, and Yulia Tsvetkov. 2022. Constrained sampling from language models via langevin dynamics in embedding spaces. In *Proc. EMNLP*.
- Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. [Natural questions: A benchmark for question answering research](#). *Transactions of the Association for Computational Linguistics*, 7:452–466.
- Philippe Laban, Tobias Schnabel, Paul N. Bennett, and Marti A. Hearst. 2022. [SummaC: Re-visiting NLI-based models for inconsistency detection in summarization](#). *Transactions of the Association for Computational Linguistics*, 10:163–177.
- Jason Lee, Elman Mansimov, and Kyunghyun Cho. 2018. Deterministic non-autoregressive neural sequence modeling by iterative refinement. In *Proc. EMNLP*.
- Jason Lee, Raphael Shu, and Kyunghyun Cho. 2020. Iterative refinement in the continuous space for non-autoregressive neural machine translation. In *Proc. EMNLP*.
- Brian Lester, Jaehoon Lee, Alex Alemi, Jeffrey Pennington, Adam Roberts, Jascha Sohl-Dickstein, and Noah Constant. 2024. Training llms over neurally compressed text. *arXiv preprint arXiv:2404.03626*.
- Alexander C Li, Mihir Prabhudesai, Shivam Duggal, Ellis Brown, and Deepak Pathak. 2023a. Your diffusion model is secretly a zero-shot classifier. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2206–2217.
- Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. [A diversity-promoting objective function for neural conversation models](#). In *Proceedings of the*

2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 110–119, San Diego, California. Association for Computational Linguistics.

Margaret Li, Suchin Gururangan, Tim Dettmers, Mike Lewis, Tim Althoff, Noah A. Smith, and Luke Zettlemoyer. 2022a. Branch-train-merge: Embarrassingly parallel training of expert language models. *ArXiv*, abs/2208.03306.

Xiang Lisa Li, Ari Holtzman, Daniel Fried, Percy Liang, Jason Eisner, Tatsunori Hashimoto, Luke Zettlemoyer, and Mike Lewis. 2023b. [Contrastive decoding: Open-ended text generation as optimization](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12286–12312, Toronto, Canada. Association for Computational Linguistics.

Xiang Lisa Li, John Thickstun, Ishaan Gulrajani, Percy Liang, and Tatsunori Hashimoto. 2022b. Diffusion-lm improves controllable text generation. *ArXiv*, abs/2205.14217.

Chin-Yew Lin. 2004. [ROUGE: A package for automatic evaluation of summaries](#). In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Zheng-Wen Lin, Yeyun Gong, Yelong Shen, Tong Wu, Zhihao Fan, Chen Lin, Weizhu Chen, and Nan Duan. 2022. Genie: Large scale pre-training for text generation with diffusion model. *ArXiv*, abs/2212.11685.

Alisa Liu and Jiacheng Liu. 2023. The memotrap dataset. <https://github.com/inverse-scaling/prize/blob/main/data-release/README.md>.

Alisa Liu, Maarten Sap, Ximing Lu, Swabha Swayamdipta, Chandra Bhagavatula, Noah A. Smith, and Yejin Choi. 2021. [DEXperts: Decoding-time controlled text generation with experts and anti-experts](#). In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6691–6706, Online. Association for Computational Linguistics.

Hao Liu, Wilson Yan, Matei Zaharia, and Pieter Abbeel. 2024a. World model on million-length video and language with ringattention. *arXiv preprint arXiv:2402.08268*.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2024b. Visual instruction tuning. *Advances in neural information processing systems*, 36.

Yang Liu, Dan Iter, Yichong Xu, Shuo Wang, Ruo Chen Xu, and Chenguang Zhu. 2023. G-eval: Nlg evaluation using gpt-4 with better human alignment. *ArXiv*, abs/2303.16634.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692.

S. Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V. Le, Barret Zoph, Jason Wei, and Adam Roberts. 2023. The flan collection: Designing data and methods for effective instruction tuning. *ArXiv*, abs/2301.13688.

- Shayne Longpre, Kartik Perisetla, Anthony Chen, Nikhil Ramesh, Chris DuBois, and Sameer Singh. 2021. [Entity-based knowledge conflicts in question answering](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7052–7063, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *Proc. ICLR*.
- Aaron Lou, Chenlin Meng, and Stefano Ermon. 2023. Discrete diffusion modeling by estimating the ratios of the data distribution. *arXiv preprint arXiv:2310.16834*.
- Justin Lovelace, Varsha Kishore, Chao gang Wan, Eliot Shekhtman, and Kilian Q. Weinberger. 2022. Latent diffusion for language generation. *ArXiv*, abs/2212.09462.
- Jiasen Lu, Christopher Clark, Sangho Lee, Zichen Zhang, Savya Khosla, Ryan Marten, Derek Hoiem, and Aniruddha Kembhavi. 2023. Unified-io 2: Scaling autoregressive multimodal models with vision, language, audio, and action. *arXiv preprint arXiv:2312.17172*.
- Ximing Lu, Peter West, Rowan Zellers, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. [NeuroLogic decoding: \(un\)supervised neural text generation with predicate logic constraints](#). In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4288–4299, Online. Association for Computational Linguistics.
- Xuezhe Ma, Chunting Zhou, Xian Li, Graham Neubig, and Eduard Hovy. 2019. Flowseq: Non-autoregressive conditional sequence generation with generative flow. In *Proc. EMNLP*.
- Rabeeh Karimi Mahabadi, Jaesung Tae, Hamish Ivison, James Henderson, Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2023. Tess: Text-to-text self-conditioned simplex diffusion. *ArXiv*, abs/2305.08379.
- Nikolay Malkin, Zhen Wang, and Nebojsa Jojic. 2022. [Coherence boosting: When your pretrained language model is not paying enough attention](#). In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8214–8236, Dublin, Ireland. Association for Computational Linguistics.
- Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan McDonald. 2020. [On faithfulness and factuality in abstractive summarization](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1906–1919, Online. Association for Computational Linguistics.
- Clara Meister, Tiago Pimentel, Gian Wiher, and Ryan Cotterell. 2023. Locally typical sampling. *Transactions of the Association for Computational Linguistics*, 11:102–121.
- Chenlin Meng, Ruiqi Gao, Diederik P. Kingma, Stefano Ermon, Jonathan Ho, and Tim Salimans. 2022. On distillation of guided diffusion models. *ArXiv*, abs/2210.03142.
- Fabian Mentzer, David Minnen, Eirikur Agustsson, and Michael Tschannen. 2023. Finite scalar quantization: Vq-vae made simple. *arXiv preprint arXiv:2309.15505*.

- Sewon Min, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen-tau Yih, Pang Wei Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2023a. Factscore: Fine-grained atomic evaluation of factual precision in long form text generation. *arXiv preprint arXiv:2305.14251*.
- Sewon Min, Weijia Shi, Mike Lewis, Xilun Chen, Wen-tau Yih, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2023b. [Nonparametric masked language modeling](#). In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 2097–2118, Toronto, Canada. Association for Computational Linguistics.
- Fatemehsadat Miresghallah, Kartik Goyal, and Taylor Berg-Kirkpatrick. 2022. Mix and match: Learning-free controllable text generation using energy language models. In *Proc. ACL*.
- John Morris, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. 2020. Textattack: A framework for adversarial attacks, data augmentation, and adversarial training in nlp. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 119–126.
- May-Britt Moser, David C Rowland, and Edvard I Moser. 2015. Place cells, grid cells, and memory. *Cold Spring Harbor perspectives in biology*, 7(2):a021808.
- Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. 2016. A corpus and cloze evaluation for deeper understanding of commonsense stories. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 839–849.
- Moin Nadeem, Tianxing He, Kyunghyun Cho, and James Glass. 2020. A systematic characterization of sampling algorithms for open-ended language generation. In *Proc. AACL*.
- Feng Nan, Cicero Nogueira dos Santos, Henghui Zhu, Patrick Ng, Kathleen McKeown, Ramesh Nallapati, Dejiao Zhang, Zhiguo Wang, Andrew O. Arnold, and Bing Xiang. 2021. [Improving factual consistency of abstractive summarization via question answering](#). In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6881–6894, Online. Association for Computational Linguistics.
- Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. [Don’t give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization](#). In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807, Brussels, Belgium. Association for Computational Linguistics.
- Alexander Quinn Nichol and Prafulla Dhariwal. 2021. Improved denoising diffusion probabilistic models. In *Proc. ICML*.
- Jekaterina Novikova, Ondřej Dušek, and Verena Rieser. 2017. The e2e dataset: New challenges for end-to-end generation. *arXiv preprint arXiv:1706.09254*.

- OpenAI. 2023. Gpt-4 technical report. *ArXiv*, abs/2303.08774.
- OpenAI. 2024. [Learning to reason with llms](#).
- Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke E. Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Francis Christiano, Jan Leike, and Ryan J. Lowe. 2022. Training language models to follow instructions with human feedback. *ArXiv*, abs/2203.02155.
- Artidoro Pagnoni, Vidhisha Balachandran, and Yulia Tsvetkov. 2021. [Understanding factuality in abstractive summarization with FRANK: A benchmark for factuality metrics](#). In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4812–4829, Online. Association for Computational Linguistics.
- Jeongsoo Park and Justin Johnson. 2023. Rgb no more: Minimally-decoded jpeg vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22334–22346.
- Damian Pascual, Beni Egressy, Clara Meister, Ryan Cotterell, and Roger Wattenhofer. 2021. [A plug-and-play method for controlled text generation](#). In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 3973–3997, Punta Cana, Dominican Republic. Association for Computational Linguistics.
- William Peebles and Saining Xie. 2023. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4195–4205.
- Xiangyu Peng, Chen Xing, Prafulla Kumar Choubey, Chien-Sheng Wu, and Caiming Xiong. 2022. Model ensemble instead of prompt fusion: a sample-specific knowledge transfer method for few-shot prompt tuning. *ArXiv*, abs/2210.12587.
- Juan C. Perez, A. Pardo, Mattia Soldan, Hani Itani, Juan Leon-Alcazar, and Bernard Ghanem. 2024. Compressed-language models for understanding compressed file formats: a jpeg exploration. *ArXiv*, abs/2405.17146.
- Krishna Pillutla, Swabha Swayamdipta, Rowan Zellers, John Thickstun, Sean Welleck, Yejin Choi, and Zaid Harchaoui. 2021. Mauve: Measuring the gap between neural text and human text using divergence frontiers. *Advances in Neural Information Processing Systems*, 34:4816–4828.
- Weizhen Qi, Yeyun Gong, Jian Jiao, Yu Yan, Weizhu Chen, Dayiheng Liu, Kewen Tang, Houqiang Li, Jiusheng Chen, Ruofei Zhang, et al. 2021. Bang: Bridging autoregressive and non-autoregressive generation with large scale pretraining. In *Proc. ICML*, pages 8630–8639. PMLR.
- Lianhui Qin, Sean Welleck, Daniel Khashabi, and Yejin Choi. 2022. Cold decoding: Energy-based constrained text generation with langevin dynamics. *ArXiv*, abs/2202.11705.

- Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR.
- Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*.
- Colin Raffel, Noam M. Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *JMLR*.
- Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. 2021. Zero-shot text-to-image generation. In *International conference on machine learning*, pages 8821–8831. Pmlr.
- Machel Reid, Vincent J. Hellendoorn, and Graham Neubig. 2022. Diffuser: Discrete diffusion via edit-based reconstruction. *ArXiv*, abs/2210.16886.
- Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695.
- David C Rowland, Yasser Roudi, May-Britt Moser, and Edvard I Moser. 2016. Ten years of grid cells. *Annual review of neuroscience*, 39(1):19–40.
- Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. 2015. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115:211–252.
- Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. 2022. Laion-5b: An open large-scale dataset for training next generation image-text models. *Advances in Neural Information Processing Systems*, 35:25278–25294.
- Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. [Get to the point: Summarization with pointer-generator networks](#). In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.
- Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proc. ACL*.
- Omar Shehata and Matthew Conlen. 2019. [ParametricPress/01-unraveling-the-jpeg: Initial public release](#).
- Weijia Shi, Xiaochuang Han, Mike Lewis, Yulia Tsvetkov, Luke Zettlemoyer, and Scott Wen-tau Yih. 2023. Trusting your evidence: Hallucinate less with context-aware decoding. *arXiv preprint arXiv:2305.14739*.

- Weijia Shi, Xiaochuang Han, Chunting Zhou, Weixin Liang, Xi Victoria Lin, Luke Zettlemoyer, and Lili Yu. 2024a. Llamafusion: Adapting pretrained language models for multimodal generation. *arXiv preprint arXiv:2412.15188*.
- Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Rich James, Mike Lewis, Luke Zettlemoyer, and Wen tau Yih. 2024b. REPLUG: Retrieval-augmented black-box language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.
- Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. 2015. Deep unsupervised learning using nonequilibrium thermodynamics. In *Proc. ICML*.
- Jiaming Song, Chenlin Meng, and Stefano Ermon. 2021. Denoising diffusion implicit models. In *Proc. ICLR*.
- Yang Song and Stefano Ermon. 2019. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32.
- Arvind Krishna Sridhar and Erik Visser. 2022. [Improved beam search for hallucination mitigation in abstractive summarization](#). *ArXiv preprint*, abs/2212.02712.
- Barry E Stein and Terrence R Stanford. 2008. Multisensory integration: current issues from the perspective of the single neuron. *Nature reviews neuroscience*, 9(4):255–266.
- Nicholas A Steinmetz, Peter Zatka-Haas, Matteo Carandini, and Kenneth D Harris. 2019. Distributed coding of choice, action and engagement across the mouse brain. *Nature*, 576(7786):266–273.
- Robin Strudel, Corentin Tallec, Florent Alch'e, Yilun Du, Yaroslav Ganin, Arthur Mensch, Will Grathwohl, Nikolay Savinov, Sander Dieleman, L. Sifre, and Rémi Leblond. 2022. Self-conditioned embedding diffusion for text generation. *ArXiv*, abs/2211.04236.
- Zhiqing Sun, Yikang Shen, Qinhong Zhou, Hongxin Zhang, Zhenfang Chen, David Cox, Yiming Yang, and Chuang Gan. 2023. Principle-driven self-alignment of language models from scratch with minimal human supervision. *Advances in Neural Information Processing Systems*, 36:2511–2565.
- Zecheng Tang, Pinzheng Wang, Keyan Zhou, Juntao Li, Ziqiang Cao, and M. Zhang. 2023. Can diffusion model achieve better performance in text generation? bridging the gap between training and inference! *ArXiv*, abs/2305.04465.
- Zhicong Tang, Shuyang Gu, Jianmin Bao, Dong Chen, and Fang Wen. 2022. Improved vector quantized diffusion models. *arXiv preprint arXiv:2205.16007*.
- Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca.
- Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023a. Llama: Open and efficient foundation language models. *ArXiv*, abs/2302.13971.

- Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023b. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.
- Brian Loeber Trippe, Jason Yim, Doug K Tischer, Tamara Broderick, David Baker, Regina Barzilay, and T. Jaakkola. 2022. Diffusion probabilistic modeling of protein backbones in 3d for the motif-scaffolding problem. *ArXiv*, abs/2206.04119.
- Alan M. Turing. 1950. [Computing machinery and intelligence](#). *Mind*, LIX:433–460.
- Aaron Van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. 2016. Conditional image generation with pixelcnn decoders. *Advances in neural information processing systems*, 29.
- Aäron Van Den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. 2016. Pixel recurrent neural networks. In *International conference on machine learning*, pages 1747–1756. PMLR.
- Aaron Van Den Oord, Oriol Vinyals, et al. 2017. Neural discrete representation learning. *Advances in neural information processing systems*, 30.
- Liam van der Poel, Ryan Cotterell, and Clara Meister. 2022. [Mutual information alleviates hallucinations in abstractive summarization](#). In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5956–5965, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proc. NeurIPS*.
- Gregory K Wallace. 1991. The jpeg still picture compression standard. *Communications of the ACM*, 34(4):30–44.
- Chunqi Wang, Ji Zhang, and Haiqing Chen. 2018. Semi-autoregressive neural machine translation. In *Proc. EMNLP*.
- Yiren Wang, Fei Tian, Di He, Tao Qin, ChengXiang Zhai, and Tie-Yan Liu. 2019. Non-autoregressive machine translation with auxiliary regularization. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 5377–5384.
- Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2022a. Self-instruct: Aligning language models with self-generated instructions. *arXiv preprint arXiv:2212.10560*.
- Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Anjana Arunkumar, Arjun Ashok, Arut Selvan Dhanasekaran, Atharva Naik, David Stap, Eshaan Pathak, Giannis Karamanolakis, Haizhi Gary Lai, Ishan Purohit, Ishani Mondal, Jacob Anderson, Kirby Kuznia, Krima Doshi, Maitreya Patel, Kuntal Kumar Pal, M. Moradshahi, Mihir Parmar, Mirali Purohit, Neeraj Varshney, Phani Rohitha Kaza, Pulkit Verma, Ravsehaj Singh Puri, Rushang Karia, Shailaja Keyur Sampat, Savan Doshi, Siddharth Deepak Mishra, Sujan Reddy,

- Sumanta Patro, Tanay Dixit, Xudong Shen, Chitta Baral, Yejin Choi, Noah A. Smith, Hanna Hajishirzi, and Daniel Khashabi. 2022b. Super-naturalinstructions: Generalization via declarative instructions on 1600+ nlp tasks. In *Conference on Empirical Methods in Natural Language Processing*.
- Sean Welleck, Ilia Kulikov, Stephen Roller, Emily Dinan, Kyunghyun Cho, and Jason Weston. 2020. Neural text generation with unlikelihood training. In *Proc. ICLR*.
- Thomas Wiegand, Gary J Sullivan, Gisle Bjontegaard, and Ajay Luthra. 2003. Overview of the h. 264/avc video coding standard. *IEEE Transactions on circuits and systems for video technology*, 13(7):560–576.
- Kevin E. Wu, Kevin Kaichuang Yang, Rianne van den Berg, James Zou, Alex X. Lu, and Ava P. Amini. 2022. Protein structure generation via folding diffusion. *ArXiv*, abs/2209.15611.
- Shangda Wu, Xu Tan, Zili Wang, Rui Wang, Xiaobing Li, and Maosong Sun. 2024. Beyond language models: Byte models are digital world simulators. *arXiv preprint arXiv:2402.19155*.
- Canwen Xu, Daya Guo, Nan Duan, and Julian McAuley. 2023. Baize: An open-source chat model with parameter-efficient tuning on self-chat data. *ArXiv*, abs/2304.01196.
- Wilson Yan, Yunzhi Zhang, Pieter Abbeel, and Aravind Srinivas. 2021. Videogpt: Video generation using vq-vae and transformers. *arXiv preprint arXiv:2104.10157*.
- Kevin Yang and Dan Klein. 2021. Fudge: Controlled text generation with future discriminators. In *Proc. NAACL*.
- Mengjiao Yang, Yilun Du, Kamyar Ghasemipour, Jonathan Tompson, Dale Schuurmans, and Pieter Abbeel. 2023. Learning interactive real-world simulators. *arXiv preprint arXiv:2310.06114*, 1(2):6.
- Michihiro Yasunaga, Armen Aghajanyan, Weijia Shi, Rich James, Jure Leskovec, Percy Liang, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. 2023. Retrieval-augmented multimodal language modeling. In *International Conference on Machine Learning (ICML)*.
- Michihiro Yasunaga, Leonid Shamis, Chunting Zhou, Andrew Cohen, Jason Weston, Luke Zettlemoyer, and Marjan Ghazvininejad. 2024. Alma: Alignment with minimal annotation. *arXiv preprint arXiv:2412.04305*.
- Jiasheng Ye, Zaixiang Zheng, Yu Bao, Lihua Qian, and Mingxuan Wang. 2023. Di-noiser: Diffused conditional sequence learning by manipulating noises. *ArXiv*, abs/2302.10025.
- Jiahui Yu, Xin Li, Jing Yu Koh, Han Zhang, Ruoming Pang, James Qin, Alexander Ku, Yuanzhong Xu, Jason Baldridge, and Yonghui Wu. 2021. Vector-quantized image modeling with improved vqgan. *arXiv preprint arXiv:2110.04627*.
- Lijun Yu, José Lezama, Nitesh B Gundavarapu, Luca Versari, Kihyuk Sohn, David Minnen, Yong Cheng, Agrim Gupta, Xiuye Gu, Alexander G Hauptmann, et al. 2023. Language model beats diffusion—tokenizer is key to visual generation. *arXiv preprint arXiv:2310.05737*.

- Hongyi Yuan, Zheng Yuan, Chuanqi Tan, Fei Huang, and Songfang Huang. 2022. Seqdiffuseq: Text diffusion with encoder-decoder transformers. *ArXiv*, abs/2212.10325.
- Hugh Zhang, Daniel Duckworth, Daphne Ippolito, and Arvind Neelakantan. 2021. Trading off diversity and quality in natural language generation. In *Proceedings of the Workshop on Human Evaluation of NLP Systems (HumEval)*, pages 25–33.
- Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022. **Opt: Open pre-trained transformer language models**. *ArXiv preprint*, abs/2205.01068.
- Tianyi Zhang, Faisal Ladhak, Esin Durmus, Percy Liang, Kathleen McKeown, and Tatsunori B. Hashimoto. 2024. **Benchmarking Large Language Models for News Summarization**. *Transactions of the Association for Computational Linguistics*, 12:39–57.
- Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Proc. NeurIPS*.
- Lin Zheng, Jianbo Yuan, Lei Yu, and Lingpeng Kong. 2023. A reparameterized discrete diffusion model for text generation. *ArXiv*, abs/2302.05737.
- Chunting Zhou, Pengfei Liu, Puxin Xu, Srini Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, Susan Zhang, Gargi Ghosh, Mike Lewis, Luke Zettlemoyer, and Omer Levy. 2023a. Lima: Less is more for alignment. *ArXiv*, abs/2305.11206.
- Chunting Zhou, Lili Yu, Arun Babu, Kushal Tirumala, Michihiro Yasunaga, Leonid Shamis, Jacob Kahn, Xuezhe Ma, Luke Zettlemoyer, and Omer Levy. 2024. Transfusion: Predict the next token and diffuse images with one multi-modal model. *arXiv preprint arXiv:2408.11039*.
- Kun Zhou, Yifan Li, Wayne Xin Zhao, and Ji rong Wen. 2023b. Diffusion-nat: Self-prompting discrete diffusion for non-autoregressive text generation. *ArXiv*, abs/2305.04044.
- Linqi Zhou, Yilun Du, and Jiajun Wu. 2021. 3d shape generation and completion through point-voxel diffusion. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5806–5815. IEEE.
- Wenxuan Zhou, Sheng Zhang, Hoifung Poon, and Muhao Chen. 2023c. **Context-faithful prompting for large language models**. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 14544–14556, Singapore. Association for Computational Linguistics.
- Chenguang Zhu, William Hinthorn, Ruochen Xu, Qingkai Zeng, Michael Zeng, Xuedong Huang, and Meng Jiang. 2021. **Enhancing factual consistency of abstractive summarization**. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 718–733, Online. Association for Computational Linguistics.