Ph.D. DISSERTATION

Data Optimization for Efficient Deep Learning

효율적인 딥러닝을 위한 데이터 최적화

August 2025

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
COLLEGE OF ENGINEERING
SEOUL NATIONAL UNIVERSITY

Jang-Hyun Kim

Ph.D. DISSERTATION

Data Optimization for Efficient Deep Learning

효율적인 딥러닝을 위한 데이터 최적화

August 2025

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
COLLEGE OF ENGINEERING
SEOUL NATIONAL UNIVERSITY

Jang-Hyun Kim

Data Optimization for Efficient Deep Learning

효율적인 딥러닝을 위한 데이터 최적화

지도교수 송 현 오

이 논문을 공학박사학위논문으로 제출함

2025 년 06 월

서울대학교 대학원

컴퓨터공학부

김 장 현

김 장 현의 공학박사 학위논문을 인준함

2025 년 06 월

위 원 장	김 건 희	(인)
부 위 원 장	송 현 오	(인)
위 원	박 재 식	(인)
위 원	주 한 별	(인)
위 원	윤 상 두	(인)

Abstract

Data serves as the foundation for how deep learning models perceive and predict reality by encapsulating real-world complexities. This dissertation investigates methods for optimizing data to enhance the learning effectiveness and inference efficiency of deep learning systems. Departing from conventional paradigms reliant on fixed-size benchmark datasets, our study introduces novel frameworks capable of generating, cleaning, and processing infinitely large amounts of data. Central to this research is the development of frameworks for co-optimizing data and models, enabling trained models to iteratively and autonomously enhance data processing in a self-reinforcing manner. Empirical evaluations across multiple domains—including visual image recognition, natural language processing, and speech processing—demonstrate the robustness and versatility of the proposed framework. The outcomes highlight significant improvements in learning performance and inference efficiency, underscoring the broad applicability and transformative potential of these techniques across industrial and scientific fields.

Keywords: Deep Learning, Image Recognition, Natural Language Processing, Synthetic Data Generation, Long-Term Memory, Efficient Inference System

Student Number: 2019-26471

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Deep learning has dramatically transformed computational capabilities, enabling breakthroughs across diverse fields, including computer vision, natural language processing, and speech recognition [142, 235]. Landmark models such as ResNet and GPT [60, 16], trained on extensive curated datasets, illustrate these remarkable advancements, consistently achieving unprecedented performance benchmarks and surpassing human-level accuracy in many tasks.

Despite this remarkable progress, recent advancements have encountered significant challenges. Primarily, deep learning models have now consumed virtually available high-quality data from the internet for training purposes [142]. As a result, the gains derived from purely scaling datasets have begun to plateau, underscoring the diminishing returns of current methodologies. Concurrently, new opportunities have emerged, particularly in enabling models to dynamically adapt through interactions and effectively learn within specialized, data-limited regimes, leveraging foundational models [15]. These evolving application scenarios, including real-time interactive AI systems and highly specialized industry domains, introduce unique challenges that traditional deep learning paradigms are poorly equipped to address.

In this dissertation, we systematically address these critical challenges by

explicitly considering the pivotal role of data within deep learning frameworks. Specifically, we tackle two pressing data-centric challenges: effective model learning under scenarios with limited labeled data, and efficient processing and utilization of infinite data streams. To overcome these challenges, we propose innovative data optimization methodologies explicitly designed to enhance both the efficiency and adaptability of deep learning systems.

A central component of our contributions is a synthetic data generation framework explicitly designed to produce informative, high-quality training samples, thereby mitigating the dependency on large-scale labeled datasets [91, 92]. Complementing this, we introduce a comprehensive large-scale data cleaning framework capable of identifying, correcting, and removing label errors and outliers, thus ensuring the integrity and reliability of training data [94]. Furthermore, we develop advanced data compression methods specifically optimized for Transformer-based architectures, significantly improving their efficiency in managing online interactions and memory usage, critical for deployment in interactive, real-world settings [93, 97].

Collectively, these contributions constitute a joint optimization framework, where improvements to the data directly facilitate enhanced model performance, which in turn yields refined data optimization strategies. This iterative, self-reinforcing cycle promotes continuous improvement, enabling deep learning models to effectively leverage both limited labeled datasets and infinitely abundant data streams for robust training and inference.

The methodologies developed and empirically validated throughout our research provide substantial practical utility while simultaneously offering meaningful theoretical insights. By proposing principled extensions to conventional deep learning frameworks, our work lays a robust foundation for future research directions, particularly in adaptive, efficient, and scalable deep learning systems. In the subsequent sections of this dissertation, we detail our key research contributions comprehensively and outline promising trajectories for future exploration inspired by these foundational advancements.

1.1 Thesis Organization

This thesis systematically addresses critical data-centric challenges within deep learning frameworks by presenting a series of structured research contributions. Following this introductory chapter, Chapter 2 outlines essential background and foundational concepts necessary for understanding subsequent chapters.

In Chapters 3 and 4, we first explore synthetic data generation techniques specifically developed for effective training under limited data scenarios. In Chapter 3, we introduce *Puzzle Mix*, an innovative data augmentation technique that strategically exploits image saliency and local statistics to achieve optimal performance in mixup-based learning methods [91]. Building upon this, Chapter 4 presents *Co-Mixup*, a saliency-guided joint mixup approach enhanced by supermodular diversity, further improving generalization by introducing structured diversity into the training data [92].

Effective training not only relies on high-quality synthetic data but also demands robust strategies to ensure data integrity. Chapter 5 introduces the *Neural Relation Graph*, a unified and robust framework that systematically detects and corrects label noise while identifying outlier data points, ensuring reliable and high-quality training datasets [94].

To handle scenarios involving infinite data streams, data compression methods become crucial for efficient training and inference. In Chapters 6 to 8, we discuss novel compression methodologies. Chapter 6 details *Dataset Condensation*, a highly efficient synthetic-data parameterization technique designed to generate concise yet highly informative datasets [93]. Complementing this, Chapter 7 explores *Compressed Context Memory*, an advanced compression method tailored for Transformer-based models, significantly enhancing memory efficiency and enabling real-time interactions [97]. Chapter 8 presents *KVzip*, a training-free compression algorithm for reducing the key-value cache size of Transformer models [98].

Finally, we summarize our contributions, discuss broader implications, and outline promising directions for future research in Chapter 9.

Chapter 2

Preliminary

In this section, we provide a formal description of the data optimization framework proposed in this dissertation.

We start by defining the classical supervised learning scenario. Given a training dataset $\mathcal{D}_{\text{train}}$ and a parameterized neural network f_θ, we optimize the model parameters θ^* by minimizing a loss function ℓ as

$$\theta^* = \underset{\theta}{\operatorname{argmin}}\, \ell(f_\theta, \mathcal{D}_{\text{train}}).$$

Then, the trained model performs inference on new data x_{t} as $f_\theta(x_{\text{t}})$. In contrast to the traditional framework, we propose a novel approach that jointly optimizes data with the model. Specifically, we introduce a function g that generates new data from the existing training set using information derived from the model f_θ. The joint optimization process is defined as

$$\theta^* = \underset{\theta}{\operatorname{argmin}}\, \ell(f_\theta, \mathcal{D}_{\text{syn}}(\theta)) \quad \text{where } \mathcal{D}_{\text{syn}}(\theta) = g(\mathcal{D}_{\text{train}}, f_\theta).$$

We propose data augmentation techniques Puzzle Mix and Co-Mixup, based

on this joint optimization framework [91, 92]. These methods leverage saliency information extracted by the model f_θ to produce enhanced training data through function g. Additionally, we develop a data refinement algorithm g that constructs a relationship graph among training data points using the model f_θ to detect and remove label errors and outliers [94].

We also introduces a dataset condensation framework designed to compress the training dataset, enabling effective model training with significantly reduced storage [93]. This optimization satisfies the following condition:

$$|\mathcal{D}_{\text{syn}}(\theta)| = |g(\mathcal{D}_{\text{train}}, f_\theta)| \ll |\mathcal{D}_{\text{train}}|.$$

Finally, we propose an inference framework tailored for online scenarios, efficiently handling sequential incoming data by compressing information into contextual memory [97]. We introduce a function g that recurrently updates memory states based on the previous memory state, the current input x_t, and the trained model f_{θ^*} as

$$\text{output} = f_{\theta^*}(x_t \mid \text{Mem}_t), \quad \text{where } \text{Mem}_t = g(\text{Mem}_{t-1}, x_t, f_{\theta^*}).$$

Here, the initial memory state Mem_0 is initialized as an empty state. Through this framework, inference on x_t leverages long-term memory states Mem_t, continuously updated by the function g, enabling effective and consistent online inference.

Overall, the proposed joint data-model optimization framework significantly enhances training efficiency by creating novel training data and compressing existing data. Additionally, our approach enables efficient long-term memory mechanisms during inference, supporting consistent and continuous information processing. The following sections detail the background, technical methods, and experimental results of these methodologies.

2.1 Related Work

In this section, we review key literature relevant to our dissertation, grouped into two categories: general data-driven learning and large language models.

2.1.1 Data-Driven Learning

Deep neural networks trained on large-scale datasets have achieved strong performance across a wide range of real-world applications [38, 235]. Landmark architectures like AlexNet [104] and ResNet [60] enabled end-to-end learning of features directly from raw inputs, surpassing traditional handcrafted methodologies [108]. This data-driven paradigm forms the foundation of modern machine learning systems [15]. However, model performance depends critically on the diversity and quality of training data [48].

Although deep networks exhibit high representational capacity, they can easily memorize random labels and noise, raising fundamental questions about generalization [231]. Addressing overfitting remains a central challenge for building reliable systems. Regularization techniques such as dropout [179] and noise injection [71] aim to suppress spurious correlations and improve robustness to label noise. These methods highlight the importance of structural inductive biases and explicit perturbations in achieving generalization. Our work contributes to this objective by introducing regularization through principled data transformations, enhancing both accuracy and robustness [91, 92].

Data augmentation improves generalization by perturbing input samples to create plausible variants [13], using techniques like cropping, flipping [104], and additive noise [12]. Mixup and its extensions interpolate inputs or features to regularize training and encourage smoother decision boundaries [232, 197]. Adaptive and saliency-aware variants address limitations such as unrealistic or semantically destructive augmentations [57]. Building on this line of work, our method introduces informative variation while preserving input locality and semantic coherence [91, 92].

2.1.2 Large Language Models

The advent of deep learning architectures has reshaped natural language processing [39]. Pretraining on large text corpus with scalable architectures achieved impressive generalization capabilities on general natural language processing tasks [199]. Generative pretraining transformers (GPT) advanced this paradigm through autoregressive pretraining, demonstrating that large-scale generative models trained on diverse text corpora can perform a broad range of tasks with minimal supervision [16]. These foundational models underpin today's large language models (LLMs), which exhibit strong few-shot and zero-shot generalization capabilities as they scale in size and data [15].

Transformer architectures process context through a stack of self-attention and feedforward layers, where each token attends to all others in the sequence [196]. The self-attention mechanism computes contextualized representations by projecting input tokens into key, query, and value vectors, then aggregating values weighted by the similarity between queries and keys. This operation captures pairwise dependencies across the sequence, allowing each layer to model token interactions globally. During inference, Transformer decoders maintain a cache of key and value vectors for previously seen tokens, enabling efficient autoregressive generation without recomputing past activations [106]. Extensions such as Transformer-XL [35] and Compressive Transformers [160] introduce mechanisms to preserve and reuse historical context across segments.

As LLMs scale in both parameter count and context length, inference-time efficiency becomes a central challenge. Sparse attention techniques like BigBird [229] and MInference [79] limit attention to a subset of tokens, reducing computational cost. Systems such as Quest [187] and Infinigen [111] introduce KV offloading and retrieval strategies to manage memory usage during decoding. However, these methods typically operate without compressing the KV cache. In contrast, our approach directly compresses the cache, enabling inference-time memory reduction without performance loss [97, 98]. This builds on prior work in sparse Transformers [26, 78, 99] and compressive memory [2, 96, 160], advancing practical techniques for KV compression through principled redundancy elimination.

Chapter 3

Puzzle Mix: Exploiting Saliency and Local Statistics for Optimal Mixup

3.1 Introduction

Deep neural network models are the bedrock of modern AI tasks such as object recognition, speech, natural language processing, and reinforcement learning. However, these models are known to memorize the training data and make overconfident predictions often resulting in degraded generalization performance on test examples [179, 230]. Furthermore, the problem is exacerbated when the models are evaluated on examples under slight distribution shift [10].

To this end, data augmentation approaches aim to alleviate some of these issues by improving the model generalization performance [59, 40]. Recently, a line of research called *mixup* has been proposed. These methods mainly focus on creating previously unseen virtual mixup examples via convex combination or local replacement of data for training [232, 197, 227, 57].

However, the underlying data domains contain rich regional saliency information (*i.e.* foreground objects in vision, prominent syllables in speech, informative textual units in language) [174, 83, 42] and exhibit local regularity structure far from random matrices of numbers [73, 235, 176]. Thus, completely disregarding these aspects of data could lead to creating mixup examples which could misguide the training model and undermine the generalization perfor-

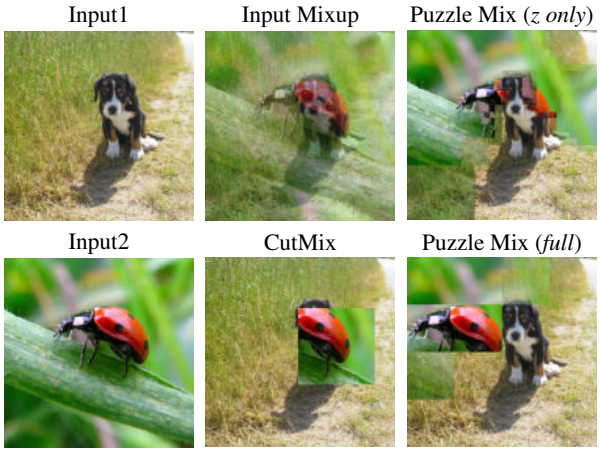


Figure 3.1 A visual comparison of the mixup methods. Puzzle Mix ensures to contain sufficient saliency information while preserving the local statistics of each input.

mance.

Motivated by this intuition, we propose Puzzle Mix, a mixup method for explicitly leveraging the saliency information and the underlying local statistics of natural examples. Our proposed method jointly seek to find (1) the optimal mask for deciding how much of the two inputs to reveal versus conceal in the given region and for (2) the transport for finding the optimal moves in order to maximize the exposed saliency under the mask. The optimization process is reminiscent of the sliding block puzzle and thus the name Puzzle Mix. Additionally, we impose the objective to respect the various underlying local statistics encouraging the optimization to preserve the structural integrity of each data. The proposed method alternates between finding the optimal mask and optimizing the transport plans, and efficiently generates the mixup examples in a mini-batch stochastic gradient descent setting.

Furthermore, our method allows us to incorporate adversarial training without any computation overhead. Adversarial training is a method for training a robust model resistant to adversarial attacks via optimization [127]. We adapt the fast adversarial training method from Wong et al. [213] and stochastically include the adversarially perturbed examples with random restarts for robustness.

Our results on CIFAR-100, Tiny-ImageNet, and ImageNet datasets show significant improvement both in the generalization task and in the adversarial robustness over existing mixup methods by a large margin.

3.2 Related Work

Data augmentation. Methods that implement data augmentation aim to regularize the models from overfitting to the training distribution and improve the generalization performance by generating virtual training examples in the vicinity of the given training dataset [13]. Some of the most commonly used data augmentation techniques are random cropping, horizontal flipping [104], and adding random noise [12]. Recently, a data augmentation method called AugMix is proposed to improve both the generalization performance and the corruption robustness [65]. Our method is complementary to these techniques and could be used in conjunction in order to further increase the generalization and robustness performance.

Mixup. Input mixup creates virtual training examples by linearly interpolating two input data and corresponding one-hot labels [232]. The method induces models to have smoother decision boundaries and reduces overfitting to the training data. Manifold mixup extends this concept from input space to feature space [197]. Also, Guo et al. [57] proposed an adaptive mixup method, which improves Input mixup by preventing the generation of improper mixup data. Yun et al. [227] proposed CutMix which implants a random rectangular region of the input into another. However, these methods can generate improper examples by randomly removing important regions of the data, which may mislead the neural network (see Figure 3.1). Our mixup method aims to prevent these issues by utilizing the saliency signal while preserving the local properties of the input data.

Saliency. Simonyan et al. [174] detects object saliency by computing gradients of a pre-trained deep neural network. Subsequently, other methods were introduced to obtain more precise saliency [240, 201]. However, these methods require modifying the pre-trained network or training new models to compute the saliency. Zhou et al. [243] and Selvaraju et al. [171] proposed methods with the reduced computation cost but at the cost of saliency resolution. We follow the method from Simonyan et al. [174], which does not require any modification to the model, to compute the saliency map. The saliency information has been used in various fields of machine learning [167, 207].

Optimal transport. A transport plan that moves a given distribution to another at the minimal cost is called the optimal transport [198]. Also, the optimal transport with discrete domain can be represented as a linear program or an assignment problem [136, 198]. The optimal transport problem is widely

Method	Mixup function $h(x_0, x_1)$
Input mixup	$(1-\lambda)x_0 + \lambda x_1$
Manifold mixup	$(1-\lambda)f(x_0) + \lambda f(x_1)$
CutMix	$(1-\mathbb{1}_B) \odot x_0 + \mathbb{1}_B \odot x_1$
Puzzle Mix	$(1-z) \odot \Pi_0^\intercal x_0 + z \odot \Pi_1^\intercal x_1$

Table 3.1 Summary of various mixup functions.

applied in various applications areas such as color transfer [158] and domain adaptation [33]. We formulate a binary transport problem for the optimal move, which maximizes the exposed saliency under the mask.

3.3 Preliminaries

Let us define $x \in \mathcal{X}$ to be an input data and $y \in \mathcal{Y}$ be its output label. Let $\mathcal{D}$ be the distribution over $\mathcal{X} \times \mathcal{Y}$. In mixup based data augmentation method, the goal is to optimize the model's loss $\ell : \mathcal{X} \times \mathcal{Y} \times \Theta \to \mathbb{R}$ given the data mixup function $h(\cdot)$ and the mixing distribution q as below.

$$\underset{\theta}{\text{minimize}} \underset{(x_0,y_0),(x_1,y_1)\in\mathcal{D}}{\mathbb{E}} \underset{\lambda\sim q}{\mathbb{E}} \ell(h(x_0,x_1), g(y_0,y_1); \theta), \tag{3.1}$$

where the label mixup function is $g(y_0, y_1) = (1-\lambda)y_0 + \lambda y_1$. Input mixup uses $h(x_0, x_1) = (1-\lambda)x_0 + \lambda x_1$. Manifold mixup employs $h(x_0, x_1) = (1-\lambda)f(x_0) + \lambda f(x_1)$ for some hidden representation f. CutMix defines $h(x_0, x_1) = (1-\mathbb{1}_B) \odot x_0 + \mathbb{1}_B \odot x_1$ for a binary rectangular mask $\mathbb{1}_B$, where $B = [r_x, r_x + r_w] \times [r_y, r_y + r_h]$ with $\lambda = \frac{r_w r_h}{WH}$ and $\odot$ represents the element-wise product. In other words, B is a randomly chosen rectangle covering λ proportion of the input. We propose the following mixup function,

$$h(x_0, x_1) = (1-z) \odot \Pi_0^\intercal x_0 + z \odot \Pi_1^\intercal x_1, \tag{3.2}$$

where z_i represents a mask in $[0, 1]$ with mixing ratio $\lambda = \frac{1}{n}\sum_i z_i$. Π_0 and Π_1 represent $n \times n$ transportation plans of the corresponding data with n dimensions. Π_{ij} encodes how much mass moves from location i to j after the transport. From now on, we omit the dependence of y and θ from the loss function ℓ for clarity. Table 3.1 summarizes various mixup functions described above. We begin Section 3.4 with the formal desiderata for our mixup function and the corresponding optimization objective.

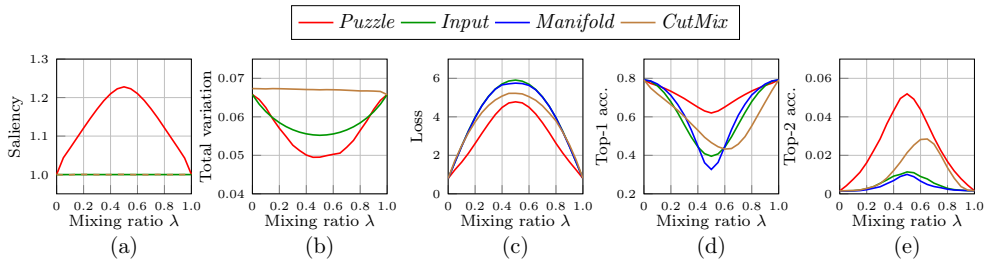


Figure 3.2 (a) Mixed saliency $||h(s(x_0), s(x_1))||_1$. Note the saliency map of each input $s(x_k)$ is normalized to sum up to 1. (b) Total variation of mixed data. (c) Cross entropy loss of mixup data and the corresponding soft-label evaluated by the vanilla classifier (ResNet18). (d) Top-1 prediction accuracy of mixed data. Prediction is counted as correct if the Top-1 prediction belongs to $\{y_0, y_1\}$. (e) Top-2 prediction accuracy of mixed data. Prediction is counted as correct if the Top-2 predictions are equal to $\{y_0, y_1\}$. Manifold mixup is omitted in (a) and (b) as manifold mixup generates mixup examples in the hidden space not in the input space.

3.4 Methods

Our goal is to maximally utilize the saliency information of each input while respecting the underlying local statistics of the data. First, in order to maximally utilize the saliency information, we seek to find the optimal mixing mask z and the optimal transport plans Π under the following criteria.

- Given a pair of transported data and a specific region, the mask z should optimally reveal more salient data of the two while masking the less salient one in the given region.
- Given a data x and the mask z, the transport Π should find the optimal moves that would maximize the saliency of the revealed portion of the data.

The criteria above motivates us to maximize for $(1-z) \odot \Pi_0^\intercal s(x_0) + z \odot \Pi_1^\intercal s(x_1)$. Note, we denote the saliency of the input x as $s(x)$ which is computed by taking ℓ_2 norm of the gradient values across input channels. Figure 3.2 (a) shows the proposed mixup function well preserves the saliency information after mixup. Second, in order to respect the underlying local statistics of the data [73, 235, 176], we consider the following criteria.

- The saliency information can be noisy, which could lead to a suboptimal solution. Therefore, we add spatial regularization terms ψ and $\phi_{i,j}$ to con-

trol the smoothness of the mask and regional smoothness of the resulting mixed example. Figure 3.2 (b) compares the local smoothness measured in total variation.

- We ensure the structural integrity within each data is generally preserved by considering the transport cost C_{ij} (defined as the distance between the locations i and j). Also, to further ensure the local salient structure of the data is preserved without being dispersed across after the transport, we optimize for the binary transport plans as opposed to continuous plans.

Evaluation results on the pretrained vanilla classifier in Figure 3.2 (c), (d), (e) show our mixup examples have the smallest loss and the highest accuracy compared to other methods, verifying our intuitions above. Moreover, we optimize the main objective after down-sampling the saliency information $s(x)$ with average pooling to support multi-scale transport and masking. From now on, we denote n as the down-sampled dimension. In practice, we select the down-sampling resolution randomly per each mini-batch.

To optimize the mask z, we first discretize the range of the mask value. Let $\mathcal{L}$ denote the discretized range $\{\frac{t}{m} \mid t = 0, 1, \ldots, m\}$. In addition, to control the mixing ratio of given inputs, we add a prior term $p(z_i)$, which follows a binomial distribution. We now formalize the complete objective in Equation (3.3).

$$
\begin{aligned}
\underset{\substack{z \in \mathcal{L}^n \\ \Pi_0, \Pi_1 \in \{0,1\}^{n \times n}}}{\text{minimize}} \quad & - \left\| (1 - z) \odot \Pi_0^\intercal s(x_0) \right\|_1 \\
& - \left\| z \odot \Pi_1^\intercal s(x_1) \right\|_1 \\
& + \beta \sum_{(i,j) \in \mathcal{N}} \psi(z_i, z_j) + \gamma \sum_{(i,j) \in \mathcal{N}} \phi_{i,j}(z_i, z_j) \\
& - \eta \sum_i \log p(z_i) + \xi \sum_{k=0,1} \langle \Pi_k, C \rangle \\
\text{subject to} \quad & \Pi_k 1_n = 1_n, \ \Pi_k^\intercal 1_n = 1_n \quad \text{for } k = 0, 1.
\end{aligned}
\tag{3.3}
$$

After solving the optimization problem in Equation (3.3), we obtain the mixed example $h(x_0, x_1) = (1 - z^*) \odot \Pi_0^{*\intercal} x_0 + z^* \odot \Pi_1^{*\intercal} x_1$ which is then used for the model training as in Equation (3.1). Figure 3.3 illustrates the mask z and the transport plan Π optimized with Equation (3.3).

We solve this optimization problem via alternating minimization through iterating first over z and then simultaneously over Π_0 and Π_1. In mixup augmentation, however, one needs to be able to efficiently generate the mixed examples as the generation process takes place per each mini-batch. Therefore, we optimize for one complete cycle of the alternating minimization, as repeated cycles

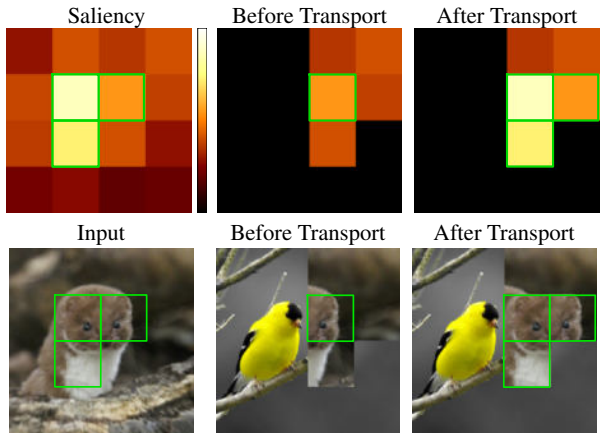


Figure 3.3 Illustration of Puzzle Mix process. After the transport, the salient regions (highlighted in green) replace the other regions, so that the salient information still remains after the mixup. The first row represents the saliency information after down-sampling, *i.e.*, $s(x)$, the masked saliency ($z \odot s(x)$), and the masked saliency after transport ($z \odot \Pi^\intercal s(x)$) respectively. The second row shows the corresponding data.

require additional network evaluations, for efficiency. As for the initialization, we optimize the mask z with Π_k initialized as identity transport, and then optimize each Π_k with the previously solved z. We now formally discuss individual optimization problems in Section 3.4.1 and Section 3.4.2.

3.4.1 Optimizing Mask

Given Π_0 and Π_1, we seek to solve the following discrete optimization problem over z in Equation (3.4). The objective is to decide how to best mix the two transported inputs jointly based on the region saliency measure (unary), the label and data local smoothness (pairwise), and the mixing weight log prior (mix prior) criteria.

$$\begin{aligned} \underset{z \in \mathcal{L}^n}{\text{minimize}} \sum_i u_i(z_i) + \beta \sum_{(i,j) \in \mathcal{N}} \psi(z_i, z_j) \\ + \gamma \sum_{(i,j) \in \mathcal{N}} \phi_{i,j}(z_i, z_j) - \eta \sum_i \log p(z_i), \end{aligned} \tag{3.4}$$

where the unary term $u_i(z_i)$ is defined as $z_i(\Pi_0^\intercal s(x_0))_i + (1 - z_i)(\Pi_1^\intercal s(x_1))_i$. We define the neighborhood $\mathcal{N}$ as a set of adjacent regions, and use the following

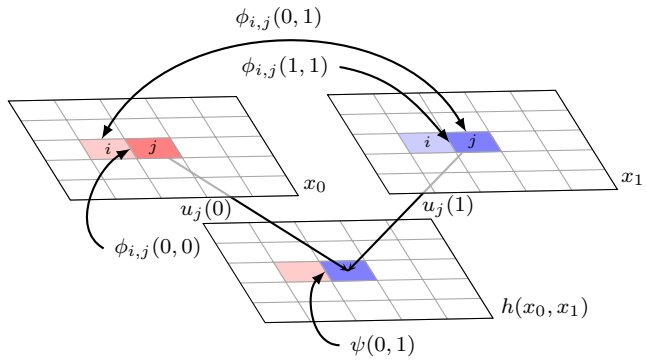


Figure 3.4 Visualization of different components in the mask optimization. Two rectangles in the top show the two inputs x_0 and x_1, and the rectangle in the bottom show the mixed output $h(x_0, x_1)$. Figure reproduced with permission from Julien Mairal.

pairwise terms and the prior term. Figure 3.4 visualizes different components in Equation (3.4).

Definition 1. *(Label smoothness)* $\psi(z_i, z_j) \coloneqq (z_i - z_j)^2$.

For data local smoothness, we measure the distance between input regions. Let d_p denote the distance function. First, we define pairwise terms under the binary case, $\mathcal{L} = \{0, 1\}$, and then extend them to the multi-label case.

Definition 2. *(Data local smoothness for binary labels)*
Let $x_{k,i}$ *represent the* i^{th} *region of data* x_k. *Then,* $\phi^b_{i,j}(z_i, z_j) \coloneqq d_p(x_{z_i,i}, x_{z_j,j})$.

The discrete optimization problem in Equation (3.4) is a type of multi-label energy minimization problem and can be efficiently solved via α-β swap algorithm [14], which is based on the graph-cuts. In the binary label case, finding the minimum s-t cut in the graph returns an equivalent optimal solution if the pairwise term satisfies the submodularity condition [102]. In our problem, the pairwise term is $e_{i,j}(z_i, z_j) = \beta\psi(z_i, z_j) + \gamma\phi_{i,j}(z_i, z_j)$. We now assume that the function values of d_p are bounded in [0,1], which is generally satisfied when data values are bounded in [0,1].

Proposition 1. *Suppose* d_p *function is bounded in [0,1] and* $\phi = \phi^b$. *If* $\gamma \leq \beta$, *then* $e_{i,j}(z_i, z_j)$ *satisfies submodularity for* $z_i, z_j \in \{0, 1\}$.

Proof. $e(0,0)+e(1,1) = \gamma\phi_{i,j}(0,0)+\gamma\phi_{i,j}(1,1) \leq 2\gamma \leq 2\beta = \beta\psi(0,1)+\beta\psi(1,0)$ $\leq e(0,1) + e(1,0)$. □

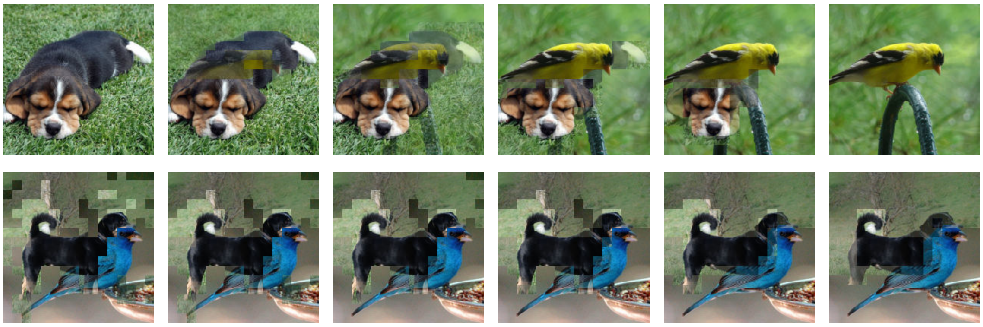


Figure 3.5 (Top row) Puzzle Mix images with increasing mixing weight λ. (Bottom row) Puzzle Mix images with increasing smoothness coefficients, β and γ. Note that the results are obtained without transport.

For multi-label case, the α-β swap algorithm iteratively applies graph-cut as a sub-routine and converges to a local-minimum if the pairwise term satisfies pairwise-submodularity [169]. We can guarantee pairwise-submodularity by slightly modifying $\phi^b_{i,j}$ as

$$\begin{aligned}
\phi^{b'}_{i,j}(0,0) &= \phi^b_{i,j}(0,0) + (\phi^b_{i,j}(0,1) + \phi^b_{i,j}(1,0))/2 \\
\phi^{b'}_{i,j}(0,1) &= \phi^b_{i,j}(0,1) + (\phi^b_{i,j}(0,0) + \phi^b_{i,j}(1,1))/2 \\
\phi^{b'}_{i,j}(1,0) &= \phi^b_{i,j}(1,0) + (\phi^b_{i,j}(0,0) + \phi^b_{i,j}(1,1))/2 \\
\phi^{b'}_{i,j}(1,1) &= \phi^b_{i,j}(1,1) + (\phi^b_{i,j}(0,1) + \phi^b_{i,j}(1,0))/2.
\end{aligned}$$

It is important to note that, $\phi^{b'}_{i,j}(1,0) + \phi^{b'}_{i,j}(0,1) - \phi^{b'}_{i,j}(0,0) - \phi^{b'}_{i,j}(1,1) = 0$.

Definition 3. *(Data local smoothness for the multi labels)*
$\phi_{i,j}(z_i, z_j) \coloneqq z_i z_j \phi^{b'}_{i,j}(1,1) + z_i(1-z_j)\phi^{b'}_{i,j}(1,0) + (1-z_i)z_j\phi^{b'}_{i,j}(0,1) + (1-z_i)(1-z_j)\phi^{b'}_{i,j}(0,0),\ {}^{\forall} z_i, z_j \in \mathcal{L}$.

Proposition 2. *With $\phi_{i,j}$ defined as Definition 3, $e_{i,j}$ satisfies pairwise submodularity.*

Proof. We can represent $\phi_{i,j}$ as follows:

$$\begin{aligned}\phi_{i,j}(z_i, z_j) = f(z_i, z_j)&\frac{\phi^{b'}_{i,j}(1,0) + \phi^{b'}_{i,j}(0,1) - \phi^{b'}_{i,j}(0,0) - \phi^{b'}_{i,j}(1,1)}{2} \\ &+ z_i \frac{\phi^{b'}_{i,j}(1,0) + \phi^{b'}_{i,j}(1,1) - \phi^{b'}_{i,j}(0,0) - \phi^{b'}_{i,j}(0,1)}{2} \\ &+ z_j \frac{\phi^{b'}_{i,j}(0,1) + \phi^{b'}_{i,j}(1,1) - \phi^{b'}_{i,j}(0,0) - \phi^{b'}_{i,j}(1,0)}{2} \\ &+ \phi^{b'}_{i,j}(0,0),\end{aligned}$$

where $f(z_i, z_j) = (1 - z_i)z_j + z_i(1 - z_j)$. By definition $\phi^{b'}_{i,j}(1,0) + \phi^{b'}_{i,j}(0,1) - \phi^{b'}_{i,j}(0,0) - \phi^{b'}_{i,j}(1,1) = 0$, and thus, $\phi_{i,j}(z_i, z_j)$ can be represented as the form of $z_i\phi^{b'_1}_{i,j} + z_j\phi^{b'_2}_{i,j} + c$. Thus, $^\forall x, y \in \mathcal{L},\ \phi_{i,j}(x, y) + \phi_{i,j}(y, x) = x\phi^{b'_1}_{i,j} + y\phi^{b'_2}_{i,j} + c + y\phi^{b'_1}_{i,j} + x\phi^{b'_2}_{i,j} + c = \phi_{i,j}(x, x) + \phi_{i,j}(y, y)$, which means $\phi_{i,j}$ satisfies pairwise submodularity.

By definition ψ satisfies pairwise submodularity, and by Lemma 1, $e_{i,j}$ satisfies pairwise submodularity. □

Lemma 1. *If ψ, ϕ satisfies pairwise submodularity and β, $\gamma \in \mathbb{R}_+$, then $\beta\psi+\gamma\phi$ satisfies pairwise submodularity.*

Finally, we use the prior term to control the ratio of inputs in the mixed output. For the given mixing weight λ, which represents the ratio of x_1 with respect to x_0, we define the prior term p to satisfy $\mathbb{E}_{z_i \sim p}[z_i] = \lambda, \forall i$. Specifically, for the label space $\mathcal{L} = \{\frac{t}{m} |\ t = 0, ..., m\}$, we define the prior term as $p(z_i = \frac{t}{m}) = \binom{m}{t}\lambda^t(1 - \lambda)^{m-t}$ for $t = 0, 1, ..., m$. In other words, $z_i \sim \frac{1}{m}B(m, \lambda)$.

In Figure 3.5, we provide the resulted mixup images using the optimal mask from Equation (3.4). Specifically, we visualize how the Puzzle Mix images change by increasing the mixing weight λ and the coefficients of the smoothness terms, β and γ.

3.4.2 Optimizing Transport

After optimizing the mask z, we optimize the transportation plans for the input data under the optimal mask z^*. Our objective with respect to transportation plans is the following.

$$
\begin{aligned}
\underset{\Pi_0,\Pi_1\in\{0,1\}^{n\times n}}{\text{minimize}} \; & -\left\|(1-z^*)\odot\Pi_0^\intercal s(x_0)\right\|_1 \\
& -\left\|(z^*\odot\Pi_1^\intercal s(x_1)\right\|_1 \\
& +\xi\sum_{k=0,1}\langle\Pi_k, C\rangle \\
\text{subject to} \;\; & \Pi_k 1_n = 1_n,\; \Pi_k^\intercal 1_n = 1_n \quad \text{for } k=0,1.
\end{aligned}
$$

Note the problem is completely separable as two independent optimization problems of each Π_k. Let $s(x_1)_i$ denote the i^{th} entry of the n-dimensional column vector $s(x_1)$. The term $\|z^*\odot\Pi_1^\intercal s(x_1)\|_1$ can be represented as $\sum_{i,j} z_j^* s(x_1)_i \Pi_{1i,j}$ $= \langle\Pi_1, s(x_1)z^{*\intercal}\rangle$. Finally, the transport optimization problem of Π_1 becomes

$$
\begin{aligned}
\underset{\Pi_1\in\{0,1\}^{n\times n}}{\text{minimize}} \; & \langle\Pi_1, C'\rangle \\
\text{subject to} \;\; & \Pi_1 1_n = 1_n,\; \Pi_1^\intercal 1_n = 1_n,
\end{aligned}
\tag{3.5}
$$

where $C' = \xi C - s(x_1)z^{*\intercal}$. C'_{ij} is the cost of moving the i^{th} region to the j^{th} position, which consists of two components. The first component is the distance ξC_{ij}, which is defined as a distance from i to j. The second component is the saliency term, which discounts the transport cost with the saliency value of the i^{th} region if the mask of j^{th} position is non-zero. Briefly speaking, the larger the saliency value, the more the discount in the transport cost.

The optimization problem in Equation (3.5) can be solved exactly by using the Hungarian algorithm and its variants with time complexity of $O(n^3)$ [136, 81]. As we need to efficiently generate mixup examples per each mini-batch, this can be a computational bottleneck as n increases. Thus, we propose an approximate algorithm that can be parallelized on GPUs and efficiently computed in batches. The proposed algorithm can quickly decrease the objective $\langle\Pi, C'\rangle$ and converges to a local-minimum within $n(n-1)/2+1$ steps.

Algorithm 1 progressively alternates between row-wise and column-wise optimizations. The algorithm first minimizes $\langle\Pi, C'\rangle$ only with the $\Pi 1_n = 1_n$ constraint. However, since the optimization is done without the column constraint, there can be multiple 1 values in a column of Π. In the following step, the column with multiple 1 values leaves only one 1 in the row with the smallest cost. We denote the result as Π_{win} in Algorithm 1. The corresponding cost entries for the rows that do not remain in Π_{win} are penalized with a large additive value, and the 1 values are moved to the other columns in the next iteration.

Our algorithm can also take advantage of intermediate Π_{win} as a solution, supported by the following two properties. We suppose that transport cost

Algorithm 1 Masked Transport

Input: mask z^*, cost C', large value v
Initialize $C^{(0)} = C',\ t = 0$
repeat
  $target = \text{argmin}(C^{(t)}, \text{dim} = 1)$
  $\Pi = 0_{n \times n}$
  for $i = 0$ **to** $n - 1$ **do**
    $\Pi[i, target[i]] = 1$
  end for
  $C_{conflict} = C^{(t)} \odot \Pi + v(1 - \Pi)$
  $source = \text{argmin}(C_{conflict}, \text{dim} = 0)$
  $\Pi_{win} = 0_{n \times n}$
  for $j = 0$ **to** $n - 1$ **do**
    $\Pi_{win}[source[j], j] = 1$
  end for
  $\Pi_{win} = \Pi_{win} \odot \Pi$
  $\Pi_{lose} = (1 - \Pi_{win}) \odot \Pi$
  $C^{(t+1)} = C^{(t)} + v\Pi_{lose}$
  $t = t + 1$
until convergence
Return: Π_{win}

matrix C has zeros in diagonal entries and positive values in others. In addition, let $\Pi^{(t)}$ and $\Pi^{(t)}_{win}$ denote Π and Π_{win} at the end of t^{th} step in Algorithm 1.

Proposition 3. *Suppose z^* has values in $\{0, 1\}$. Then for j s.t. $z^*_j = 1$, j^{th} column of $\Pi^{(t)}_{win}$ has exactly one 1.*

Proof. By the definition of $C^{(0)} = \xi C - s(x_0)z^{*\intercal}$, for j s.t. $z^*_j = 1$, j^{th} row of $C^{(0)}$ has a minimum at j^{th} entry. Thus, j^{th} column of $\Pi^{(0)}_{win}$ has exactly one 1 and others are 0. Suppose that, the claim is satisfied for $\Pi^{(t)}_{win}$ and $\Pi^{(t)}_{win}[i(j), j]$ is 1 for j s.t. $z^*_j = 1$. Then, by the definition of $\Pi^{(t)}_{win}$, $C^{(t)}_{win}[i(j), j]$ is the minimum of $i(j)^{th}$ row of $C^{(t)}$ and the row will not be updated in $C^{(t+1)}$. Thus, $i(j)^{th}$ row of $C^{(t+1)}$ has a minimum at j^{th} entry and j^{th} column of $\Pi^{(t+1)}_{win}$ has exactly one 1. By induction, the claim holds. □

Proposition 4. *Under the assumption of Proposition 3, the partial objective $< \Pi^{(t)}_{win}, C'z^* >$ decreases as t increases.*

Proof. By Proposition 3, for j s.t. $z_j^* = 1$, j^{th} column of $\Pi_{win}^{(t)}$ has exactly one 1. Let $i(j;t)$ denote the corresponding row index with the entry 1. Then, it is enough to prove that $C'[i(j;t+1), j] \leq C'[i(j;t), j]$. However, in the last part of the proof of Proposition 3, we showed that $i(j;t)^{th}$ row of $C^{(t+1)}$ has a minimum at j^{th} entry, and thus $\Pi^{(t+1)}[i(j;t), j] = 1$. By Algorithm 1, index $i(j;t+1)$ satisfies $C^{(t+1)}[i(j;t+1), j] \leq C^{(t+1)}[i, j]$, $\forall i$ s.t. $\Pi^{(t+1)}[i, j] = 1$. Thus, $C^{(t+1)}[i(j;t+1), j] \leq C^{(t+1)}[i(j;t), j]$. Finally, $\Pi^{(t+1)}[i, j] = 1$ means that cost from i to j is not updated, *i.e.*, $C^{(t+1)}[i, j] = C'[i, j]$. □

Finally, we introduce the convergence property of Algorithm 1.

Proposition 5. *Algorithm 1 converges to a local-minimum with respect to the update rule at most* $n(n-1)/2 + 1$ *steps.*

3.4.3 Adversarial Training

Since our mix-up strategy utilizes the gradients of the loss function with respect to the given inputs for saliency computation, we can incorporate adversarial training in our mix-up method without *any* additional computation cost.

For adversarial training on mixup data, we adapt the fast adversarial training method of Wong et al. [213], which adds a uniform noise before creating an adversarial perturbation. As shown in Algorithm 2, we add the adversarial perturbation to the proper location of the mixed output, *i.e.*, adding an adversarial signal to the corresponding input and location specified by z. Note that the adversarial perturbation is added to each data probabilistically to prevent possible degradation in the generalization performance.

3.5 Implementation Details

First, to solve the discrete optimization problem with respect to the mask z, we use α-β swap algorithm from the pyGCO python wrapper[1]. Although the minimization is performed example-wise in CPUs, the α-β swap algorithm converges quickly, since we restrict the size of the graph with down-sampling. Note that, in our experiments, the computational bottleneck of the method is in the forward-backward passes of the neural network. In our experiments, we use label space $\mathcal{L} = \{0, \frac{1}{2}, 1\}$. In addition, we randomly sample the size of the graph, *i.e.*, size of mask z, from $\{2 \times 2,\ 4 \times 4,\ 8 \times 8,\ 16 \times 16\}$, and down-sample the given mini-batch for all experiments.

We normalize the down-sampled saliency map, which is used as the unary term, to sum up to 1. This allows us to use consistent hyperparameters *across*

[1] https://github.com/Borda/pyGCO

Algorithm 2 Stochastic Adversarial Puzzle Mix

Input: data x_0, x_1, attack ball ϵ, step τ, probability p
$x_{i,clean} = x_i$ for $i = 0, 1$
Sample $\nu_i \sim B(1, p)$ for $i = 0, 1$
for $i = 1, 2$ **do**
  if $\nu_i == 1$ **then**
    $\kappa_i \sim Uniform(-\epsilon, \epsilon)$
    $x_i \leftarrow x_i + \kappa_i$
  end if
end for
Calculate gradient $\nabla_x l(x_i)$ for $i = 0, 1$
Optimize z^* and Π_i^* in Equation (3.3)
Sample $\delta \sim Uniform(0, 1)$
for $i = 0, 1$ **do**
  if $\nu_i == 1$ **then**
    $\kappa_i \leftarrow \kappa_i + \tau \ sign(\nabla_x l(x_i))$
    $\kappa_i \leftarrow clip(\kappa_i, -\epsilon, \epsilon)$
    $x_i \leftarrow x_{i,clean} + \delta \ \kappa_i$
  end if
end for
Return: $(1 - z^*) \odot \Pi_0^{*\intercal} x_0 + z^* \odot \Pi_1^{*\intercal} x_1$

all the models and datasets. To measure the distance between the two adjacent data regions, we compute the mean of the absolute values of differences on the boundaries. For the mixing ratio λ, we randomly sample λ from $Beta(\alpha, \alpha)$ at each mini-batch. All of the computations in our algorithm except α-β swap are done in mini-batch and can be performed in parallel in GPUs. Note that for-loops in Algorithm 1 can be done in parallel by using the scatter function of PyTorch [147].

Since we calculate the saliency information by back-propagating the gradient of loss function through the model, we can utilize this gradient information without any computational overhead. We regularize the gradient of the model with mixup data as $\nabla_\theta \ell(h(x_0, x_1), g(y_0, y_1); \theta) + \frac{1}{2}\lambda_{clean}(\nabla_\theta \ell(x_0, y_0; \theta) + \nabla_\theta \ell(x_1, y_1; \theta))$. This additional regularization helps us to improve generalization performance on Tiny-ImageNet and ImageNet.

3.6 Experiments

We train and evaluate classifiers on CIFAR-100 [103], Tiny-ImageNet [28], and ImageNet [38] datasets. We first study the generalization performance and adversarial robustness of our method (Section 3.6.1). Next, we show that our method can be used in conjunction with the existing augmentation method (AugMix) to simultaneously improve the corruption robustness and generalization performance (Section 3.6.2). Finally, we perform ablation studies for our method (Section 3.6.3).

3.6.1 Generalization Performance and Adversarial Robustness

CIFAR-100. We train two residual neural networks [59]: WRN28-10 [228] and PreActResNet18 [61]. We follow the training protocol of Verma et al. [197], which trains WRN28-10 for 400 epochs and PreActResNet18 for 1200 epochs. We reproduce the mixup baselines [232, 197, 227, 65] and compare the baselines with our method under the same experimental settings described above. We denote the experiments as *Vanilla, Input, Manifold, CutMix, AugMix, Puzzle Mix* in the experiment tables.

Note however, our mixup method requires an additional forward-backward evaluation of the network per mini-batch to calculate the saliency signal. For some practitioners, a fairer comparison would be to compare the performances at a fixed number of network evaluations (*i.e.,* for power conservation). In order to compare our method in this condition, we also test our method trained for half the epochs and with twice the initial learning rate. We denote this experiment as *Puzzle Mix (half)* in the experiment tables.

In addition, we report experiments with the adversarial training described in Algorithm 2 with $p = 0.1$. We denote this experiment as *Puzzle Mix (adv)* in the tables. We assess the adversarial robustness against FGSM attack of $8/255$ ℓ_∞ epsilon ball following the evaluation protocol of Zhang et al. [232], Verma et al. [197], Yun et al. [227] for fair comparison. The results are summarized in Table 3.2 and Table 3.3.

We observe that Puzzle Mix outperforms other mixup baselines in generalization and adversarial robustness with WRN28-10 (Table 3.2) and PreActResNet18 (Table 3.3). With WRN28-10, *Puzzle Mix* improves Top-1 test error over the best performing baseline by 1.45%, and *Puzzle Mix (half)* outperforms by 1.17%. *Puzzle Mix (adv)* improves FGSM error rate over 8.41% than AugMix while achieving 1.39% lower Top-1 error rate than Manifold mixup, which had the best Top-1 score among baselines. We observe similar results with PreActResNet18. *Puzzle Mix (adv)* reduces the Top-1 error rate by 1.14% and the FGSM error rate by 12.98% over baselines.

Method	Top-1 Error(%)	Top-5 Error(%)	FGSM Error(%)
Vanilla	21.14	6.33	63.92
Input	18.27	4.98	56.60
Manifold	17.40	4.37	60.70
Manifold†	18.04	-	-
CutMix	17.50	4.69	79.34
AugMix	20.44	5.74	55.59
Puzzle Mix	**15.95**	3.92	63.71
Puzzle Mix (half)	16.23	**3.90**	66.74
Puzzle Mix (adv)	16.01	3.91	47.18
Puzzle Mix (half, adv)	16.39	3.94	**46.95**

Table 3.2 Top-1, Top-5, and FGSM error rates on CIFAR-100 dataset of WRN28-10 trained with various mixup methods (400 epochs). † denotes the result reported in the original paper. Top-1 and Top-5 results are median test errors of models in the last 10 epochs.

Tiny-ImageNet. We train PreActResNet18 network on Tiny-ImageNet dataset, which contains 200 classes with 500 training images and 50 test images per class with 64×64 resolution [28].

As in the CIFAR-100 experiment, Puzzle Mix shows significant performance gains both on the generalization performance and the adversarial robustness compared to other mixup baselines (Table 3.4).

Puzzle Mix trained with the same number of epochs achieves 36.52% in Top-1 test error, 5.47% lower than the strongest baseline, and the model trained with same network evaluations (*half*) outperforms the best baseline by 4.35%. Puzzle Mix trained with stochastic adversarial method (*adv*) achieves best Top-1 and FGSM error rate ($\epsilon = 4/255$) compared to other mixup baselines providing 3.44% lower Top-1 error rate and 5.12% lower FGSM error rate.

ImageNet. In ImageNet experiment, we use ResNet-50 to compare the performance. In order to train the model on ImageNet more efficiently, we utilize the cyclic learning rate, and use pre-resized images following the training protocol in Wong et al. [213] which trains models for 100 epochs. Consistent with the previous experiments on CIFAR-100 and Tiny-ImageNet, Puzzle Mix shows the best performance in both Top-1 / Top-5 error rate, achieving 0.43%, 0.24% improvement each, compared to the best baseline (Table 3.5).

We further test Puzzle Mix according to the experimental setting of CutMix [227] which trains models for 300 epochs and measures the **best** test accuracy

Method	Top-1 Error(%)	Top-5 Error(%)	FGSM Error(%)
Vanilla	23.67	8.98	88.89
Input	23.16	7.58	70.09
Manifold	20.98	6.63	73.09
CutMix	23.20	8.09	86.38
AugMix	24.69	8.38	76.99
Puzzle Mix	**19.62**	5.85	79.47
Puzzle Mix (half)	20.09	**5.59**	75.72
Puzzle Mix (adv)	19.84	6.11	**57.11**
Puzzle Mix (half, adv)	19.96	6.20	59.33

Table 3.3 Top-1, Top-5, and FGSM error rates on CIFAR-100 dataset of PreActResNet18 trained with various mixup methods.

among the training. As shown in table 3.6, Puzzle Mix outperforms baselines consistently.

3.6.2 Robustness Against Corruption

Hendrycks et al. [65] proposed AugMix which performs Input mixup between clean and augmented images to improve robustness against corrupted datasets as well as the generalization performance. AugMix uses Jensen-Shannon divergence (JSD) between network outputs of a clean image and two AugMix images as a consistency loss. However, computing the JSD term requires triple the network evaluations compared to other mixup methods to train the network.

We found that simply using our mixup algorithm between two AugMix images, improves both the generalization and corruption robustness over the training strategy with the JSD objective. Note that our method requires only one additional (versus two) network evaluation per each mini-batch. We denote this experiment setting as *Puzzle Mix (aug)*.

We use CIFAR-100-C dataset [63] to evaluate the corruption robustness. The dataset consists of 19 types of corruption, including noise, blur, weather, and digital corruption types. In Table 3.7, we report average test errors on CIFAR-100-C dataset as well as test errors on the clean CIFAR-100 test dataset. Table 3.7 demonstrates that our method using AugMix images improves both the generalization performance and the corruption accuracy by 3.95% and 2.31% each over AugMix baseline.

Method	Top-1 Error(%)	Top-5 Error(%)	FGSM Error(%)
Vanilla	42.77	26.35	91.85
Input	43.41	26.98	88.68
Manifold	41.99	25.88	89.25
Manifold†	41.30	26.41	-
CutMix	43.33	24.48	87.19
AugMix	44.03	25.32	90.00
Puzzle Mix	**36.52**	**18.95**	92.52
Puzzle Mix (half)	37.64	19.37	92.57
Puzzle Mix (adv)	38.55	20.48	**82.07**
Puzzle Mix (half, adv)	38.14	19.70	83.91

Table 3.4 Top-1, Top-5, and FGSM error rates on Tiny-ImageNet dataset for PreActResNet18 trained with various mixup methods.

Model	Top-1 Error(%)	Top-5 Error(%)
Vanilla	24.31	7.34
Input	22.99	6.48
Manifold	23.15	6.50
CutMix	22.92	6.55
AugMix	23.25	6.70
Puzzle Mix	**22.49**	**6.24**

Table 3.5 Top-1 and Top-5 error rates on ImageNet on ResNet-50 following the training protocol in Wong et al. [213] (100 epochs).

3.6.3 Ablation Study

The generalization performance of Puzzle Mix stems from saliency-based multi-label masking and transport. We verified the effectiveness of these two factors in comparative experiments on CIFAR-100 with WRN28-10. Table 3.8 shows that Puzzle Mix with the binary label space (*binary*) has 1.44% higher Top-1 error rate than multi-label case, and Puzzle Mix without transport (*mask only*) has 0.43% higher Top-1 error rate than Puzzle Mix with transport.

We also verify the effects of different factors in stochastic adversarial training. In Algorithm 2, we add an adversarial perturbation to each data based on each Bernoulli sample ν_i and apply linear decay with δ sampled from the uniform distribution. From Table 3.9, we observe that using two independent

Model	Best Top-1 Error(%)	Best Top-5 Error(%)
Vanilla	23.68	7.05
Input	22.58	6.40
Manifold	22.50	6.21
CutMix	21.40	5.92
Puzzle Mix	**21.24**	**5.71**

Table 3.6 Best Top-1 and Top-5 error rates on ImageNet on ResNet-50 following the training protocol in [227] (300 epochs).

Method	Top-1 Error(%)	Corruption Error(%)
Vanilla	21.14	49.08
AugMix	20.45	32.22
Puzzle Mix	**15.95**	42.46
Puzzle Mix (*aug*)	16.50	**29.91**

Table 3.7 Top-1 and Corruption error rates on CIFAR-100 and CIFAR-100-C on WRN28-10.

random variables ν_0 and ν_1 (*adv*) has significant improvement in adversarial robustness over using one variable ($\nu_0 = \nu_1$). In the absence of linear decaying (*fgsm*), there is improvement in the FGSM error rate of 4.02%, but the Top-1 error increases by 0.41%. In all experiments, p is set to 0.1. We use FGSM attack of $8/255$ ℓ_∞ epsilon-ball and 7-step PGD attack with a $2/255$ step size.

3.7 Conclusion

We have presented Puzzle Mix, a mixup augmentation method for optimally leveraging the saliency information while respecting the underlying local statistics of the data. Puzzle Mix efficiently generates the mixup examples in a mini-batch stochastic gradient descent setting and outperforms other mixup baseline methods both in the generalization performance and the robustness against adversarial perturbations and data corruption by a large margin on CIFAR-100, Tiny-ImageNet, and ImageNet datasets.

Method	Top-1 Error(%)	Top-5 Error(%)
Vanilla	21.14	6.33
Puzzle Mix	**15.95**	3.92
Puzzle Mix (*binary*)	17.39	4.34
Puzzle Mix (*mask only*)	16.38	**3.78**

Table 3.8 Top-1 and Top-5 rates on CIFAR-100 dataset of WRN28-10 trained with our mixup methods.

Method	Top-1 Error(%)	FGSM Error(%)	PGD Error(%)
Puzzle Mix (*adv*)	**16.01**	47.18	**90.18**
Puzzle Mix (*fgsm*)	16.42	**43.16**	91.19
Puzzle Mix (ν_0=ν_1)	16.66	65.90	94.05

Table 3.9 Top-1, FGSM, and PGD error rates on CIFAR-100 dataset of WRN28-10 trained with our adversarial mixup methods.

Chapter 4

Co-Mixup: Saliency Guided Joint Mixup with Supermodular Diversity

4.1 Introduction

Deep neural networks have been applied to a wide range of artificial intelligence tasks such as computer vision, natural language processing, and signal processing with remarkable performance [166, 39, 141]. However, it has been shown that neural networks have excessive representation capability and can even fit random data [230]. Due to these characteristics, the neural networks can easily overfit to training data and show a large generalization gap when tested on previously unseen data.

To improve the generalization performance of the neural networks, a body of research has been proposed to develop regularizers based on priors or to augment the training data with task-dependent transforms [13, 34]. Recently, a new task-independent data augmentation technique, called *mixup*, has been proposed [232]. The original mixup, called *Input Mixup*, linearly interpolates a given pair of input data and can be easily applied to various data and tasks, improving the generalization performance and robustness of neural networks. Other mixup methods, such as *manifold mixup* [197] or *CutMix* [227], have also been proposed addressing different ways to mix a given pair of input data. *Puzzle Mix* [91] utilizes saliency information and local statistics to ensure mixup

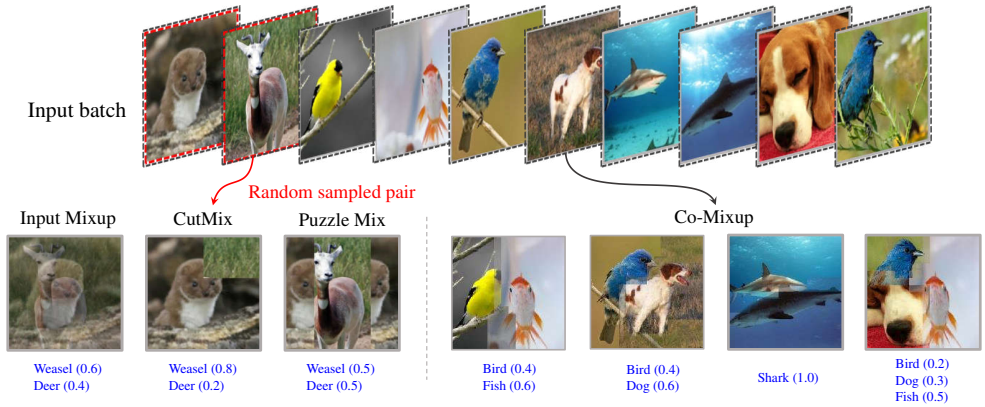


Figure 4.1 Example comparison of existing mixup methods and the proposed Co-Mixup.

data to have rich supervisory signals.

However, these approaches only consider mixing a given random pair of input data and do not fully utilize the rich informative supervisory signal in training data including collection of object saliency, relative arrangement, etc. In this work, we simultaneously consider mix-matching different salient regions among all input data so that each generated mixup example accumulates as many salient regions from multiple input data as possible while ensuring diversity among the generated mixup examples. To this end, we propose a novel optimization problem that maximizes the saliency measure of each individual mixup example while encouraging diversity among them collectively. This formulation results in a novel discrete submodular-supermodular objective. We also propose a practical modular approximation method for the supermodular term and present an efficient iterative submodular minimization algorithm suitable for minibatch-based mixup for neural network training. As illustrated in the Figure 4.1, while the proposed method, *Co-Mixup*, mix-matches the collection of salient regions utilizing inter-arrangements among input data, the existing methods do not consider the saliency information (Input Mixup & CutMix) or disassemble salient parts (Puzzle Mix).

We verify the performance of the proposed method by training classifiers on CIFAR-100, Tiny-ImageNet, ImageNet, and the Google commands dataset [103, 28, 38, 205]. Our experiments show the models trained with Co-Mixup achieve the state of the performance compared to other mixup baselines. In addition to the generalization experiment, we conduct weakly-supervised object localization and robustness tasks and confirm Co-Mixup outperforms other mixup baselines.

4.2 Related work

Mixup. Data augmentation has been widely used to prevent deep neural networks from over-fitting to the training data [12]. The majority of conventional augmentation methods generate new data by applying transformations depending on the data type or the target task [34]. Zhang et al. [232] proposed *mixup*, which can be independently applied to various data types and tasks, and improves generalization and robustness of deep neural networks. *Input mixup* [232] linearly interpolates between two input data and utilizes the mixed data with the corresponding soft label for training. Following this work, *manifold mixup* [197] applies the mixup in the hidden feature space, and *CutMix* [227] suggests a spatial copy and paste based mixup strategy on images. Guo et al. [57] trains an additional neural network to optimize a mixing ratio. *Puzzle Mix* [91] proposes a mixup method based on saliency and local statistics of the given data. In this paper, we propose a discrete optimization-based mixup method simultaneously finding the best combination of collections of salient regions among all input data while encouraging diversity among the generated mixup examples.

Saliency. The seminal work from Simonyan et al. [174] generates a saliency map using a pre-trained neural network classifier without any additional training of the network. Following the work, measuring the saliency of data using neural networks has been studied to obtain a more precise saliency map [240, 201] or to reduce the saliency computation cost [243, 171]. The saliency information is widely applied to the tasks in various domains, such as object segmentation or speech recognition [82, 84].

Submodular-Supermodular optimization. A submodular (supermodular) function is a set function with diminishing (increasing) returns property [137]. It is known that any set function can be expressed as the sum of a submodular and supermodular function [125], called BP function. Various problems in machine learning can be naturally formulated as BP functions [44], but it is known to be NP-hard [125]. Therefore, approximate algorithms based on modular approximations of submodular or supermodular terms have been developed [76]. Our formulation falls into a category of BP function consisting of smoothness function within a mixed output (submodular) and a diversity function among the mixup outputs (supermodular).

4.3 Preliminary

Existing mixup methods return $\{h(x_1, x_{i(1)}), \dots, h(x_m, x_{i(m)})\}$ for given input data $\{x_1, \dots, x_m\}$, where $h : \mathcal{X} \times \mathcal{X} \to \mathcal{X}$ is a mixup function and $(i(1), \dots, i(m))$ is a random permutation of the data indices. In the case of input mixup, $h(x, x')$ is $\lambda x + (1 - \lambda)x'$, where $\lambda \in [0, 1]$ is a random mixing ratio. Manifold mixup applies input mixup in the hidden feature space, and CutMix uses $h(x, x') = \mathbb{1}_B \odot x + (1 - \mathbb{1}_B) \odot x'$, where $\mathbb{1}_B$ is a binary rectangular-shape mask for an image x and $\odot$ represents the element-wise product. Puzzle Mix defines $h(x, x')$ as $z \odot \Pi^\intercal x + (1-z) \odot \Pi'^\intercal x'$, where Π is a transport plan and z is a discrete mask. In detail, for $x \in \mathbb{R}^n$, $\Pi \in \{0, 1\}^n$ and $z \in \mathcal{L}^n$ for $\mathcal{L} = \{\frac{l}{L} \mid l = 0, 1, \dots, L\}$.

In this work, we extend the existing mixup functions as $h : \mathcal{X}^m \to \mathcal{X}^{m'}$ which performs mixup on a collection of input data and returns another collection. Let $x_B \in \mathbb{R}^{m \times n}$ denote the batch of input data in matrix form. Then, our proposed mixup function is

$$h(x_B) = \big(g(z_1 \odot x_B), \dots, g(z_{m'} \odot x_B)\big),$$

where $z_j \in \mathcal{L}^{m \times n}$ for $j = 1, \dots, m'$ with $\mathcal{L} = \{\frac{l}{L} \mid l = 0, 1, \dots, L\}$ and $g : \mathbb{R}^{m \times n} \to \mathbb{R}^n$ returns a column-wise sum of a given matrix. Note that, the k^{th} column of z_j, denoted as $z_{j,k} \in \mathcal{L}^m$, can be interpreted as the mixing ratio among m inputs at the k^{th} location. Also, we enforce $\|z_{j,k}\|_1 = 1$ to maintain the overall statistics of the given input batch. Given the one-hot target labels $y_B \in \{0, 1\}^{m \times C}$ of the input data with C classes, we generate soft target labels for mixup data as $y_B^\intercal \tilde{o}_j$ for $j = 1, \dots, m'$, where $\tilde{o}_j = \frac{1}{n} \sum_{k=1}^{n} z_{j,k} \in [0, 1]^m$ represents the input source ratio of the j^{th} mixup data. We train models to estimate the soft target labels by minimizing the cross-entropy loss.

4.4 Method

4.4.1 Objective

Saliency. Our main objective is to maximize the saliency measure of mixup data while maintaining the local smoothness of data, *i.e.*, spatially nearby patches in a natural image look similar, temporally adjacent signals have similar spectrum in speech, etc. [91]. As we can see from CutMix in Figure 4.1, disregarding saliency can give a misleading supervisory signal by generating mixup data that does not match with the target soft label. While the existing mixup methods only consider the mixup between two inputs, we generalize the number of inputs m to any positive integer. Note, each k^{th} location of outputs has m candidate sources from the inputs. We model the unary labeling cost as

the negative value of the saliency, and denote the cost vector at the k^{th} location as $c_k \in \mathbb{R}^m$. For the saliency measure, we calculate the gradient values of training loss with respect to the input and measure ℓ_2 norm of the gradient values across input channels [174, 91]. Note that this method does not require any additional architecture dependent modules for saliency calculation. In addition to the unary cost, we encourage adjacent locations to have similar labels for the smoothness of each mixup data. In summary, the objective can be formulated as follows:

$$\sum_{j=1}^{m'}\sum_{k=1}^{n} c_k^\intercal z_{j,k} + \beta \sum_{j=1}^{m'} \sum_{(k,k')\in\mathcal{N}} (1 - z_{j,k}^\intercal z_{j,k'}) - \eta \sum_{j=1}^{m'}\sum_{k=1}^{n} \log p(z_{j,k}),$$

where the prior p is given by $z_{j,k} \sim \frac{1}{L} Multi(L, \lambda)$ with $\lambda = (\lambda_1, \ldots, \lambda_m) \sim Dirichlet(\alpha, \ldots, \alpha)$, which is a generalization of the mixing ratio distribution of [232], and $\mathcal{N}$ denotes a set of adjacent locations (*i.e.*, neighboring image patches in vision, subsequent spectrums in speech, etc.).

Diversity. Note that the naive generalization above leads to the identical outputs because the objective is separable and identical for each output. In order to obtain diverse mixup outputs, we model a similarity penalty between outputs. First, we represent the input source information of the j^{th} output by aggregating assigned labels as $\sum_{k=1}^{n} z_{j,k}$. For simplicity, let us denote $\sum_{k=1}^{n} z_{j,k}$ as o_j. Then, we measure the similarity between o_j's by using the inner-product on $\mathbb{R}^m$. In addition to the input source similarity between outputs, we model the compatibility between input sources, represented as a symmetric matrix $A_c \in \mathbb{R}_+^{m\times m}$. Specifically, $A_c[i_1, i_2]$ quantifies the degree to which input i_1 and i_2 are suitable to be mixed together. In summary, we use inner-product on $A = (1-\omega)I + \omega A_c$ for $\omega \in [0, 1]$, resulting in a supermodular penalty term. Note that, by minimizing $\langle o_j, o_{j'} \rangle_A = o_j^\intercal A o_{j'}$, $\forall j \neq j'$, we penalize output mixup examples with similar input sources and encourage each individual mixup examples to have high compatibility within. In this work, we measure the distance between locations of salient objects in each input and use the distance matrix $A_c[i, j] = \|\text{argmax}_k s_i[k] - \text{argmax}_k s_j[k]\|_1$, where s_i is the saliency map of the i^{th} input and k is a location index (*e.g.*, k is a 2-D index for image data). From now on, we denote this inner-product term as the *compatibility* term.

Over-penalization. The conventional mixup methods perform mixup as many as the number of examples in a given mini-batch. In our setting, this is the case when $m = m'$. However, the compatibility penalty between outputs is influenced

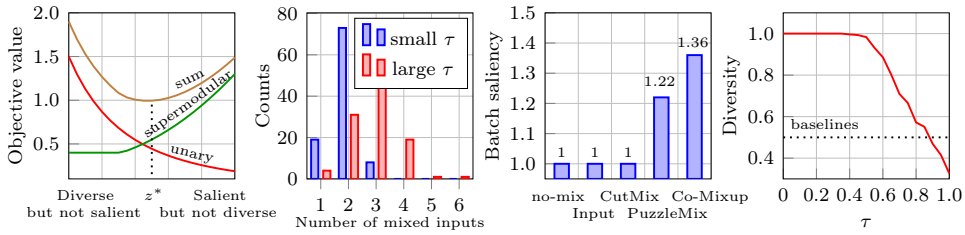


Figure 4.2 (a) Analysis of our BP optimization problem. The x-axis is a one-dimensional arrangement of solutions: The mixed output is more salient but not diverse towards the right and less salient but diverse on the left. The unary term (red) decreases towards the right side of the axis, while the supermodular term (green) increases. By optimizing the sum of the two terms (brown), we obtain the balanced output z^*. (b) A histogram of the number of inputs mixed for each output given a batch of 100 examples from the ImageNet dataset. As τ increases, more inputs are used to create each output on average. (c) Mean batch saliency measurement of a batch of mixup data using the ImageNet dataset. We normalize the saliency measure of each input to sum up to 1. (d) Diversity measurement of a batch of mixup data. We calculate the diversity as $1 - \sum_j \sum_{j' \neq j} \tilde{o}_j^\intercal \tilde{o}_{j'}/m$, where $\tilde{o}_j = o_j / \|o_j\|_1$. We can control the diversity among Co-Mixup data (red) and find the optimum by controlling τ.

by the pigeonhole principle. For example, suppose the first output consists of two inputs. Then, the inputs must be used again for the remaining $m' - 1$ outputs, or only $m - 2$ inputs can be used. In the latter case, the number of available inputs $(m - 2)$ is less than the outputs $(m' - 1)$, and thus, the same input must be used more than twice. Empirically, we found that the remaining compatibility term above over-penalizes the optimization so that a substantial portion of outputs are returned as singletons without any mixup. To mitigate the over-penalization issue, we apply clipping to the compatibility penalty term. Specifically, we model the objective so that no extra penalty would occur when the compatibility among outputs is below a certain level.

Now we present our main objective as following:

$$z^* = \underset{z_{j,k} \in \mathcal{L}^m,\ \|z_{j,k}\|_1 = 1}{\operatorname{argmin}} f(z),$$

where

$$f(z) \coloneqq \sum_{j=1}^{m'} \sum_{k=1}^{n} c_k^{\intercal} z_{j,k} + \beta \sum_{j=1}^{m'} \sum_{(k,k') \in \mathcal{N}} (1 - z_{j,k}^{\intercal} z_{j,k'}) \tag{4.1}$$
$$+ \gamma \underbrace{\max \left\{ \tau, \sum_{j=1}^{m'} \sum_{j' \neq j}^{m'} \left(\sum_{k=1}^{n} z_{j,k} \right)^{\intercal} A \left(\sum_{k=1}^{n} z_{j',k} \right) \right\}}_{= f_c(z)} - \eta \sum_{j=1}^{m'} \sum_{k=1}^{n} \log p(z_{j,k}).$$

In Figure 4.2, we describe the properties of the BP optimization problem of Equation (4.1) and statistics of the resulting mixup data. Next, we verify the supermodularity of the compatibility term. We first extend the definition of the submodularity of a multi-label function as follows [211].

Definition 4. *For a given label set $\mathcal{L}$, a function $s : \mathcal{L}^m \times \mathcal{L}^m \to \mathbb{R}$ is pairwise submodular, if $\forall x, x' \in \mathcal{L}^m$, $s(x, x) + s(x', x') \leq s(x, x') + s(x', x)$. A function s is pairwise supermodular, if $-s$ is pairwise submodular.*

Proposition 6. *The compatibility term f_c in Equation (4.1) is pairwise supermodular for every pair of $(z_{j_1,k}, z_{j_2,k})$ if A is positive semi-definite.*

Proof. $s(x, x) + s(x', x') - s(x, x') - s(x', x) = x^{\intercal}Ax + x^{\intercal}Ax - 2x^{\intercal}Ax' = (x - x')^{\intercal}A(x - x')$, and because A is positive semi-definite, $(x - x')^{\intercal}A(x - x') \geq 0$. □

Finally note that, $A = (1 - \omega)I + \omega A_c$, where A_c is a symmetric matrix. By using spectral decomposition, A_c can be represented as $UDU^{\intercal}$, where D is a diagonal matrix and $U^{\intercal}U = UU^{\intercal} = I$. Then, $A = U((1 - \omega)I + \omega D)U^{\intercal}$, and thus for small $\omega > 0$, we can guarantee A to be positive semi-definite.

4.4.2 Algorithm

Our main objective consists of modular (*unary, prior*), submodular (*smoothness*), and supermodular (*compatibility*) terms. To optimize the main objective, we employ the submodular-supermodular procedure by iteratively approximating the supermodular term as a modular function [137]. Note that z_j represents the labeling of the j^{th} output and o_j represents the aggregated input source information of the j^{th} output, $\sum_{k=1}^{n} z_{j,k}$. Before introducing our algorithm, we first inspect the simpler case without clipping.

Proposition 7. *The compatibility term f_c without clipping is modular with respect to z_j.*

Proof. Note, A is a positive symmetric matrix by the definition. Then, for an index j_0, we can represent f_c without clipping in terms of o_{j_0} as

$$\begin{aligned}
&\sum_{j=1}^{m'} \sum_{j'=1, j' \neq j}^{m'} o_j^\intercal A o_{j'} \\
&= 2 \sum_{j=1, j \neq j_0}^{m'} o_j^\intercal A o_{j_0} + \sum_{j=1, j \neq j_0}^{m'} \sum_{j'=1, j' \notin \{j_0, j\}}^{m'} o_j^\intercal A o_{j'} \\
&= (2 \sum_{j=1, j \neq j_0}^{m'} A o_j)^\intercal o_{j_0} + c \\
&= v_{\text{-}j_0}^\intercal o_{j_0} + c,
\end{aligned}$$

where $v_{\text{-}j_0} \in \mathbb{R}^m$ and $c \in \mathbb{R}$ are values independent with o_{j_0}. Finally, $v_{\text{-}j_0}^\intercal o_{j_0} + c = \sum_{k=1}^{n} v_{\text{-}j_0}^\intercal z_{j_0,k} + c$ is a modular function of z_{j_0}. □

By Proposition 7, we can apply a submodular minimization algorithm to optimize the objective with respect to z_j when there is no clipping. Thus, we can optimize the main objective without clipping in coordinate descent fashion [214]. For the case with clipping, we modularize the supermodular compatibility term under the following criteria:

1. The modularized function value should increase as the compatibility across outputs increases.
2. The modularized function should not apply an extra penalty for the compatibility below a certain level.

Borrowing the notation from the proof in Proposition 7, for an index j, $f_c(z) = \max\{\tau, v_{\text{-}j}^\intercal o_j + c\} = \max\{\tau - c, v_{\text{-}j}^\intercal o_j\} + c$. Note, $o_j = \sum_{k=1}^{n} z_{j,k}$ represents the input source information of the j^{th} output and $v_{\text{-}j} = 2 \sum_{j'=1, j' \neq j}^{m'} A o_{j'}$ encodes the status of the other outputs. Thus, we can interpret the supermodular term as a penalization of each label of o_j in proportion to the corresponding $v_{\text{-}j}$ value (criterion 1), but not for the compatibility below $\tau - c$ (criterion 2). As a modular function which satisfies the criteria above, we use the following function:

$$f_c(z) \approx \max\{\tau', v_{\text{-}j}\}^\intercal o_j \quad \text{for } \exists \tau' \in \mathbb{R}. \tag{4.2}$$

Note that, by satisfying the criteria above, the modular function reflects the diversity and over-penalization desiderata described in Section 4.4.1. We illustrate the proposed mixup procedure with the modularized diversity penalty in Figure 4.3.

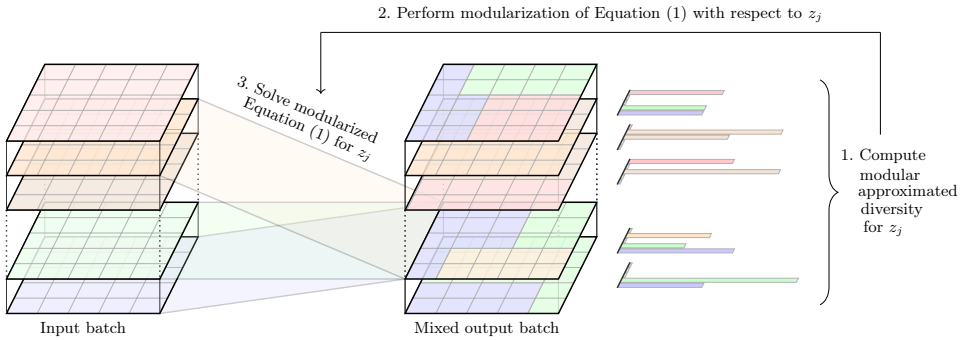


Figure 4.3 Visualization of the proposed mixup procedure. For a given batch of input data (left), a batch of mixup data (right) is generated, which mix-matches different salient regions among the input data while preserving the diversity among the mixup examples. The histograms on the right represent the input source information of each mixup data (o_j).

Proposition 8. *The modularization given by Equation* (4.2) *satisfies the criteria above.*

Proof. For j_1 and j_2, *s.t.*, $j_1 \neq j_2$,

$$\begin{aligned}
&\max\Big\{\tau, \sum_{j=1}^{m'} \sum_{j'=1, j'\neq j}^{m'} (\sum_{k=1}^{n} z_{j,k})^\intercal A (\sum_{k=1}^{n} z_{j',k})\Big\} \\
&= \max\{\tau, c + 2z_{j_1,k}^\intercal A z_{j_2,k}\} \\
&= -\min\{-\tau, -c - 2z_{j_1,k}^\intercal A z_{j_2,k}\},
\end{aligned}$$

for $\exists c \in \mathbb{R}$. By Lemma 1, $-z_{j_1,k}^\intercal A z_{j_2,k}$ is pairwise submodular, and because a budget additive function preserves submodularity [67], $\min\{-\tau, -c - 2z_{j_1,k}^\intercal A z_{j_2,k}\}$ is pairwise submodular with respect to $(z_{j_1,k}, z_{j_2,k})$. □

By applying the modular approximation described in Equation (4.2) to f_c in Equation (4.1), we can iteratively apply a submodular minimization algorithm to obtain the final solution as described in Algorithm 3. In detail, each step can be performed as follows: 1) Conditioning the main objective f on the current values except z_j, denoted as $f_j(z_j) = f(z_j; z_{1:j-1}, z_{j+1:m'})$. 2) Modularization of the compatibility term of f_j as Equation (4.2), resulting in a submodular function $\tilde{f}_j$. We denote the modularization operator as Φ, *i.e.*, $\tilde{f}_j = \Phi(f_j)$. 3) Applying a submodular minimization algorithm to $\tilde{f}_j$.

Algorithm 3 Iterative submodular minimization

Initialize z as $z^{(0)}$.
Let $z^{(t)}$ denote a solution of the t^{th} step.
Φ: modularization operator based on Equation (4.2).
for $t = 1, \ldots, T$ **do**
  for $j = 1, \ldots, m'$ **do**
    $f_j^{(t)}(z_j) \coloneqq f(z_j; z_{1:j-1}^{(t)}, z_{j+1:m'}^{(t-1)})$.
    $\tilde{f}_j^{(t)} = \Phi(f_j^{(t)})$.
    Solve $z_j^{(t)} = \operatorname{argmin} \tilde{f}_j^{(t)}(z_j)$.
  end for
end for
return $z^{(T)}$

Analysis Narasimhan and Bilmes [137] proposed a modularization strategy for general supermodular set functions, and apply a submodular minimization algorithm that can monotonically decrease the original BP objective. However, the proposed Algorithm 3 based on Equation (4.2) is much more suitable for minibatch based mixup for neural network training than the set modularization proposed by Narasimhan and Bilmes [137] in terms of complexity and modularization variance due to randomness. For simplicity, let us assume each $z_{j,k}$ is an m-dimensional one-hot vector. Then, our problem is to optimize $m'n$ one-hot m-dimensional vectors.

To apply the set modularization method, we need to assign each possible value of $z_{j,k}$ as an element of $\{1, 2, \ldots, m\}$. Then the supermodular term in Equation (4.1) can be interpreted as a set function with $m'nm$ elements, and to apply the set modularization, $O(m'nm)$ sequential evaluations of the supermodular term are required. In contrast, Algorithm 3 calculates $v_{\cdot j}$ in Equation (4.2) in only $O(m')$ time per each iteration. In addition, each modularization step of the set modularization method requires a random permutation of the $m'nm$ elements. In this case, the optimization can be strongly affected by the randomness from the permutation step. As a result, the optimal labeling of each $z_{j,k}$ from the compatibility term is strongly influenced by the random ordering undermining the interpretability of the algorithm.

Dataset (Model)	Vanilla	Input	Manifold	CutMix	Puzzle Mix	Co-Mixup
CIFAR-100 (PreActResNet18)	23.59	22.43	21.64	21.29	20.62	**19.87**
CIFAR-100 (WRN16-8)	21.70	20.08	20.55	20.14	19.24	**19.15**
CIFAR-100 (ResNeXt29-4-24)	21.79	21.70	22.28	21.86	21.12	**19.78**
Tiny-ImageNet (PreActResNet18)	43.40	43.48	40.76	43.11	36.52	**35.85**
ImageNet (ResNet-50)	24.03	22.97	23.30	22.92	22.49	**22.39**
Google commands (VGG-11)	4.84	3.91	3.67	3.76	3.70	**3.54**

Table 4.1 Top-1 error rate on various datasets and models. For CIFAR-100, we train each model with three different random seeds and report the mean error.

4.5 Experiments

We evaluate our proposed mixup method on generalization, weakly supervised object localization, calibration, and robustness tasks. First, we compare the generalization performance of the proposed method against baselines by training classifiers on CIFAR-100 [103], Tiny-ImageNet [28], ImageNet [38], and the Google commands speech dataset [205]. Next, we test the localization performance of classifiers following the evaluation protocol of Qin and Kim [157]. We also measure calibration error [56] of classifiers to verify Co-Mixup successfully alleviates the over-confidence issue by Zhang et al. [232]. In Section 4.5.4, we evaluate the robustness of the classifiers on the test dataset with background corruption in response to the recent problem raised by Lee et al. [110] that deep neural network agents often fail to generalize to unseen environments.

4.5.1 Classification

We first train PreActResNet18 [61], WRN16-8 [228], and ResNeXt29-4-24 [220] on CIFAR-100 for 300 epochs. We use stochastic gradient descent with an initial learning rate of 0.2 decayed by factor 0.1 at epochs 100 and 200. We set the momentum as 0.9 and add a weight decay of 0.0001. With this setup, we train a vanilla classifier and reproduce the mixup baselines [232, 197, 227, 91], which we denote as *Vanilla, Input, Manifold, CutMix, Puzzle Mix* in the experiment tables. Note that we use identical hyperparameters regarding Co-Mixup over all of the experiments with different models and datasets.

Table 4.1 shows Co-Mixup significantly outperforms all other baselines in Top-1 error rate. Co-Mixup achieves 19.87% in Top-1 error rate with PreActResNet18, outperforming the best baseline by 0.75%. We further test Co-Mixup on different models (WRN16-8 & ResNeXt29-4-24) and verify Co-Mixup improves Top-1 error rate over the best performing baseline.

We further test Co-Mixup on other datasets; Tiny-ImageNet, ImageNet, and the Google commands dataset (Table 4.1). For Tiny-ImageNet, we train

Task	Vanilla	Input	Manifold	CutMix	Puzzle Mix	Co-Mixup
Localization (Acc. %) (↑)	54.36	55.07	54.86	54.91	55.22	**55.32**
Calibration (ECE %) (↓)	3.9	17.7	13.1	5.6	7.5	**1.9**

Table 4.2 WSOL results on ImageNet and ECE (%) measurements of CIFAR-100 classifiers.

PreActResNet18 for 1200 epochs following the training protocol of Kim et al. [91]. As a result, Co-Mixup consistently improves Top-1 error rate over baselines by 0.67%. In the ImageNet experiment, we follow the experimental protocol provided in Puzzle Mix [91], which trains ResNet-50 [60] for 100 epochs. As a result, Co-Mixup outperforms all of the baselines in Top-1 error rate. We further test Co-Mixup on the speech domain with the Google commands dataset and VGG-11 [173]. From Table 4.1, we confirm that Co-Mixup is the most effective in the speech domain as well.

4.5.2 Localization

We compare weakly supervised object localization (WSOL) performance of classifiers trained on ImageNet (in Table 4.1) to demonstrate that our mixup method better guides a classifier to focus on salient regions. We test the localization performance using CAM [243], a WSOL method using a pre-trained classifier. We evaluate localization performance following the evaluation protocol in Qin and Kim [157], with binarization threshold 0.25 in CAM. Table 4.2 summarizes the WSOL performance of various mixup methods, which shows that our proposed mixup method outperforms other baselines.

4.5.3 Calibration

We evaluate the expected calibration error (ECE) [56] of classifiers trained on CIFAR-100. Note, ECE is calculated by the weighted average of the absolute difference between the confidence and accuracy of a classifier. As shown in Table 4.2, the Co-Mixup classifier has the lowest calibration error among baselines. From Figure 4.4, we find that other mixup baselines tend to have *under-confident* predictions resulting in higher ECE values even than *Vanilla* network (also pointed out by Wen et al. [209]), whereas Co-Mixup has best-calibrated predictions resulting in relatively 48% less ECE value.

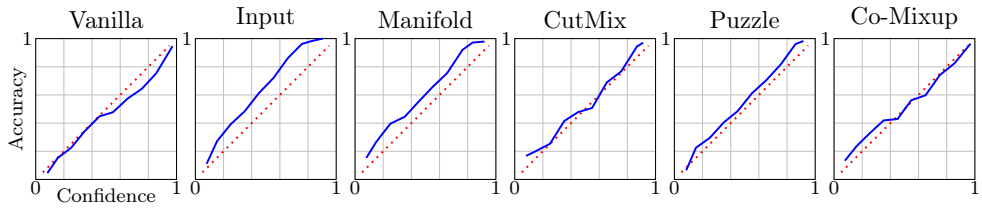


Figure 4.4 Confidence-Accuracy plots for classifiers on CIFAR-100. From the figure, the Vanilla network shows over-confident predictions, whereas other mixup baselines tend to have under-confident predictions. We can find that Co-Mixup has best-calibrated predictions.

Corruption type	Vanilla	Input	Manifold	CutMix	Puzzle Mix	Co-Mixup
Random replacement	41.63 (+17.62)	39.41 (+16.47)	39.72 (+16.47)	46.20 (+23.16)	39.23 (+16.69)	**38.77** (+16.38)
Gaussian noise	29.22 (+5.21)	26.29 (+3.35)	26.79 (+3.54)	27.13 (+4.09)	26.11 (+3.57)	**25.89** (+3.49)

Table 4.3 Top-1 error rates of various mixup methods for background corrupted ImageNet validation set. The values in the parentheses indicate the error rate increment by corrupted inputs compared to clean inputs.

4.5.4 Robustness

In response to the recent problem raised by Lee et al. [110] that deep neural network agents often fail to generalize to unseen environments, we consider the situation where the statistics of the foreground object, such as color or shape, is unchanged, but with the corrupted (or replaced) background. In detail, we consider the following operations: 1) replacement with another image and 2) adding Gaussian noise. We use ground-truth bounding boxes to separate the foreground from the background, and then apply the previous operations independently to obtain test datasets.

With the test datasets described above, we evaluate the robustness of the pre-trained classifiers. As shown in Table 4.3, Co-Mixup shows significant performance gains at various background corruption tests compared to the other mixup baselines. For each corruption case, the classifier trained with Co-Mixup outperforms the others in Top-1 error rate with the performance margins of 2.86% and 3.33% over the Vanilla model.

# inputs for mixup	Input	Manifold	CutMix	Co-Mixup
# inputs = 2	22.43	21.64	21.29	
# inputs = 3	23.03	22.13	22.01	19.87
# inputs = 4	23.12	22.07	22.20	

Table 4.4 Top-1 error rates of mixup baselines with multiple mixing inputs on CIFAR-100 and PreActResNet18. We report the mean values of three different random seeds. Note that Co-Mixup optimally determines the number of inputs for each output by solving the optimization problem.

4.5.5 Baselines with multiple inputs

To further investigate the effect of the number of inputs for the mixup in isolation, we conduct an ablation study on baselines using multiple mixing inputs. For fair comparison, we use Dirichlet$(\alpha, \ldots, \alpha)$ prior for the mixing ratio distribution and select the best performing α in $\{0.2, 1.0, 2.0\}$. Note that we overlay multiple boxes in the case of CutMix. Table 4.4 reports the classification test errors on CIFAR-100 with PreActResNet18. From the table, we find that mixing multiple inputs decreases the performance gains of each mixup baseline. These results demonstrate that mixing multiple inputs could lead to possible degradation of the performance and support the necessity of considering saliency information and diversity as in Co-Mixup.

4.6 Conclusion

We presented Co-Mixup for optimal construction of a batch of mixup examples by finding the best combination of salient regions among a collection of input data while encouraging diversity among the generated mixup examples. This leads to a discrete optimization problem minimizing a novel submodular-supermodular objective. In this respect, we present a practical modular approximation and iterative submodular optimization algorithm suitable for mini-batch based neural network training. Our experiments on generalization, weakly supervised object localization, and robustness against background corruption show Co-Mixup achieves the state of the art performance compared to other mixup baseline methods. The proposed generalized mixup framework tackles the important question of 'what to mix?' while the existing methods only consider 'how to mix?'. We believe this work can be applied to new applications where the existing mixup methods have not been applied, such as multi-label classification, multi-object detection, or source separation.

Chapter 5

Neural Relation Graph: A Unified Framework for Identifying Label Noise and Outlier Data

5.1 Introduction

Identifying problems within datasets is crucial for improving the robustness of machine learning systems and analyzing the model failures [172]. For instance, identifying mislabeled or uninformative data helps construct concise and effective training datasets [139], while identifying whether test data is OOD or corrupted allows for more accurate model evaluation and analysis [195].

In recent years, efforts have been made to identify problematic data by utilizing unary scores on individual data from trained models, such as estimating data influence [100], monitoring prediction variability throughout training [191], and calculating prediction error margins [140]. However, identifying such data can be challenging, particularly when dealing with large-scale datasets from real-world distributions. In real-world settings, datasets may have complex problems, including label errors, under-representation, and outliers, each of which can lead to the model error and prediction sensitivity [101]. For example, Figure 5.1 shows that a neural network exhibits low negative prediction margins and high loss values for both a sample with label error and outlier data. This observation indicates that previous unary scoring methods may have limitations in discerning whether the problem lies with the label or the data itself.

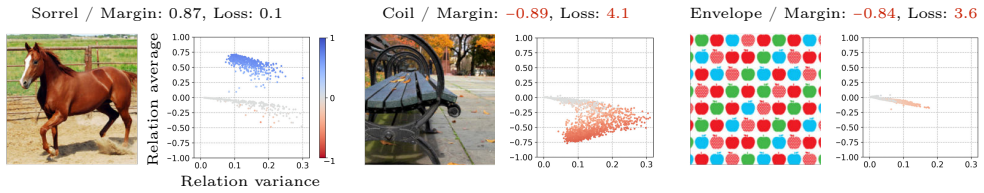


Figure 5.1 ImageNet samples with their labels and the corresponding relation maps by an MAE-Large model [62]. We report the prediction margin score ($\in [-1, 1]$) and the loss value next to the label. The relation map draws a scatter plot of the mean and variance of relation values of a data pair throughout the training process. Here the color represents the relation value at the last converged checkpoint. We present the detailed procedure for generating the relation maps in Section 5.3.6.

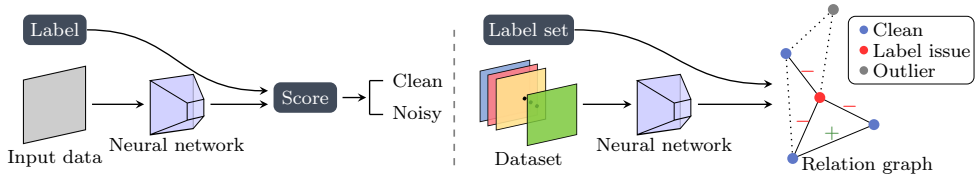


Figure 5.2 The conceptual illustration of the conventional approaches (left) and our proposed approach (right). In the relation graph, positive edges signify complementary relations, negative edges denote conflicting relations, and dashed lines indicate negligible relations between data.

In this work, we propose a unified framework for identifying label errors and outliers by leveraging the feature-embedded structure of a dataset that provides richer information than individual data alone [178, 146]. We measure the relationship among data in the feature embedding space while comparing the assigned labels independently. By comparing input data and labels separately, we are able to isolate the factors contributing to model errors, resulting in improved identification of label errors and outlier data, respectively. Based on the relational information, we construct a novel graph structure on the dataset and identify whether the data itself or the label is problematic (Figure 5.2). To this end, we develop scalable graph algorithms that accurately identify label errors and outlier data points.

In Section 5.3.6, we further introduce a visualization tool named data relation map that captures the relational structure of a data point. Through the map, we can understand the underlying relational structure and interactively

diagnose data. In Figure 5.1, we observe different patterns in the relation maps of the second and third samples, despite their similar margin and loss scores. This highlights that the relational structure provides complementary information not captured by the unary scoring methods.

Our approach only requires the model's feature embedding and prediction score on data, making it more scalable compared to methods that require calculating the network gradient on each data point or retraining models multiple times to estimate data influence [155, 75]. Furthermore, our method is domain- and model-agnostic, and thus is applicable to various tasks. We evaluate our approach on label error and outlier/OOD detection tasks with large-scale image, speech, and language datasets: ImageNet [38], ESC-50 [152], and SST2 [199]. Our experiments show state-of-the-art performance on all tasks, demonstrating its effectiveness for debugging and cleaning datasets over various domains.

5.2 Related work

Label error detection. Label errors in datasets can negatively impact model generalization and destabilize evaluation systems [71, 140]. Prior works address this issue through label error detection using bagging and bootstrapping [175, 164], or employing neural networks [80, 49, 86]. To mitigate overfitting on label errors, Pleiss et al. [153] propose tracking the training process to measure the area under the margin curve. Recent studies demonstrate that simple scoring methods with large pre-trained models, such as prediction margins or loss values, achieve comparable results to previous complex approaches [139, 27]. Meanwhile, Wu et al. [216] propose a unified approach for learning with open-world noisy data. However, the method involves a complicated optimization process during training, which is not suitable for large-scale settings. Another line of approach to identifying label errors involves measuring the influence of a training data point on its own loss [100, 155]. However, these approaches require calculating computationally expensive network gradients on each data point, and their performance is known to be sensitive to outliers and training schemes [8, 9]. In this work, we present a scalable approach that leverages the data relational structure of trained models without additional training procedures, facilitating practical analysis of label issues.

Outlier/OOD detection. Detecting outlier data is crucial for building robust machine learning systems in real-world environments [101]. A recent survey paper defines the problem of finding outliers in training set as *outlier detection* and finding outliers in the inference process as *OOD detection* [223]. The conventional approach for detecting outliers involves measuring k-nearest

distance using efficient sampling methods [182]. More recently, attempts have been made to detect outlier data using scores obtained from trained neural networks, such as Maximum Softmax Probability [64], Energy score [119], and Max Logit score [66]. Other approaches suggest adding perturbations on the inputs or rectifying the activation values to identify the outlier data [116, 183]. Lee et al. [109] propose fitting a Gaussian probabilistic model to estimate the data distribution. Recently, Sun et al. [184] propose a non-parametric approach measuring the k-nearest feature distance. In our work, we explore the use of the relational structure on the feature-embedded space for identifying outlier data. Our approach is applicable to a wide range of domains without requiring additional training while outperforming existing scoring methods on large-scale outlier/OOD detection benchmarks.

5.3 Methods

In this section, we describe our method for identifying label errors and outliers using a model trained on the noisy training dataset. We exploit the feature-embedded structure of the learned neural networks, which are known to effectively capture the underlying semantics of the data [162]. We define data relation to construct a data relation graph on the feature space, and introduce our novel graph algorithms for identifying label errors and outlier data. In Section 5.3.6, we introduce the data relation map as an effective visualization tool for diagnosing and contextualizing data.

5.3.1 Data relation

We describe our approach in the context of a classification task, while also noting that the ideas are generalizable to other types of tasks as well. We assume the presence of a trained neural network on a noisy training dataset with label errors and outliers, $\mathcal{T} = \{(x_i, y_i) \mid i = 1, \ldots, n\}$. By utilizing data features extracted from the network, we measure the semantic similarity between data points with a bounded kernel $k : \mathcal{X} \times \mathcal{X} \to [0, M]$, where a higher kernel value indicates greater similarity between data points. Our framework can accommodate various bounded kernels such as RBF kernel or cosine similarity [226]. We provide detailed information on the kernel function used in our main experiments in Section 5.3.4.

By incorporating the assigned label information with the similarity kernel k, we define the relation function $r : \mathcal{X} \times \mathcal{Y} \times \mathcal{X} \times \mathcal{Y} \to [-M, M]$:

$$r\left((x_i, y_i), (x_j, y_j)\right) = 1(y_i = y_j) \cdot k(x_i, x_j), \tag{5.1}$$

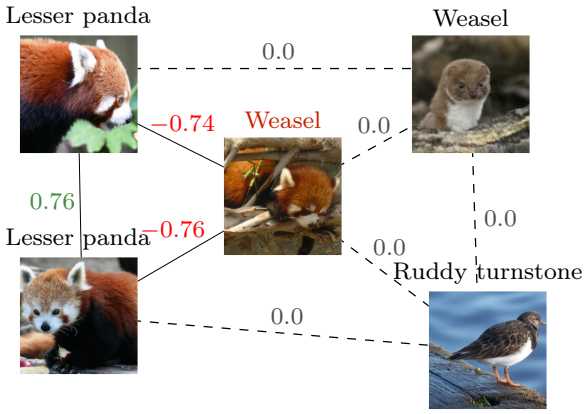


Figure 5.3 Relation values of samples from ImageNet with MAE-Large [62]. We denote the assigned label above each sample. Here, the center image has a label error.

where $1(y_i = y_j) \in \{-1, 1\}$ is a signed indicator value. The relation function reflects the degree to which data samples are complementary or conflicting with each other. In Figure 5.3, the center image with a label error has negative relations to the left samples that belong to the same ground-truth class. In contrast, the two left samples with correct labels have a positive relation. We also note that samples with dissimilar semantics exhibit near-zero relations.

Our relation function r relies solely on the parallelizable forward computation of neural networks, ensuring scalability in large-scale settings.

5.3.2 Label error detection

We consider a fully-connected undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$, where the set of nodes $\mathcal{V}$ corresponds to $\mathcal{T}$ and the weights $\mathcal{W}$ on edges $\mathcal{E}$ are the negative relation values defined in Equation (5.1). For notation clarity, we denote a data point by an index, *i.e.*, $\mathcal{T} = \{1, \ldots, n\}$. Then, for nodes i and j, the edge weight is $w(i, j) = -r(i, j) = -r((x_i, y_i), (x_j, y_j))$. We set $w(i, i)$ to 0, which does not correspond to any edges in the graph. Consistent with previous works [139], we aim to measure the ***label noisiness score*** for each data, where a higher score indicates a higher likelihood of label error. We denote the label noisiness scores for $\mathcal{T}$ as $s \in \mathbb{R}^n$, where $s[i]$ is the score for data i.

As depicted in Figure 5.3, data with label errors exhibit negative relations with other samples, implying that the data have similar features in the embedding space yet have dissimilarly assigned labels. This suggests that the edge

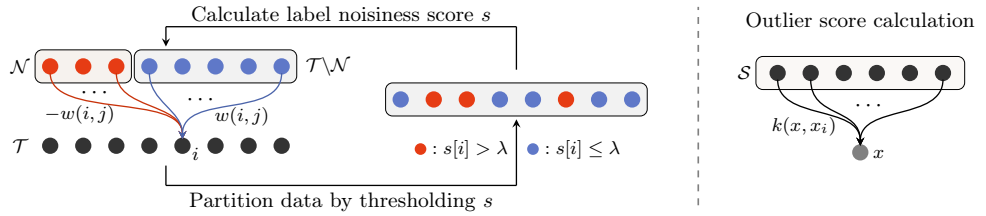


Figure 5.4 Illustration of our scoring algorithms for identifying label noise (left) and outliers (right).

weights $w(i, \cdot)$ quantify the extent to which the label assigned to node i conflicts with the labels of other nodes. However, simply aggregating all edge weights of a node can yield suboptimal results, as negative relations can also contribute to the score for clean data, as shown in Figure 5.3.

To rectify this issue, we develop an algorithm that considers the global structure of the graph instead of simply summing the edge weights of individual nodes. Specifically, we identify subsets of data likely to have correct/incorrect labels and calculate the label noisiness score based on the subsets. We partition the nodes in $\mathcal{T}$ into two groups, where $\mathcal{N} \subset \mathcal{T}$ denotes the ***estimated noisy subset*** and $\mathcal{T}\backslash\mathcal{N}$ denotes the clean subset. To optimize $\mathcal{N}$, we aim to maximize the sum of the edges between the two groups, indicating that the label information of the two groups is the most conflicting. To ensure that $\mathcal{N}$ contains data with incorrect labels, which constitute a relatively small proportion of $\mathcal{T}$, we impose regularization to the cardinality of $\mathcal{N}$ with $\lambda \geq 0$ and formulate the following max-cut problem:

$$\mathcal{N}^* = \underset{\mathcal{N} \subset \mathcal{T}}{\operatorname{argmax}} \ \operatorname{cut}(\mathcal{N}, \mathcal{T}\backslash\mathcal{N}) \ \Big(:= \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{T}\backslash\mathcal{N}} w(i,j) \Big) - \lambda |\mathcal{N}|. \tag{5.2}$$

The max-cut problem is NP-complete [58]. To solve this problem, we adopt the Kerninghan-Lin algorithm, which finds a local optimum by iteratively updating the solution [89]. However, the original algorithm that swaps data one by one at each optimization iteration is not suitable for large-scale settings. To this end, we propose an efficient *set-level* algorithm in Algorithm 4 that alternatively updates the noisy set $\mathcal{N}$ and label noisiness score vector s.

Specifically, given the current estimation of $\mathcal{N}$, the cut value excluding edges of node $i \in \mathcal{T}$ is $\operatorname{cut}(\mathcal{N}\backslash\{i\}, \mathcal{T}\backslash\mathcal{N}\backslash\{i\})$. Algorithm 4 measures the label noisiness score of node i by comparing the objective cut values when including i in $\mathcal{N}$ and when including i in $\mathcal{T}\backslash\mathcal{N}$:

Algorithm 4 Label noise identification

Input: Relation function r $(= -w)$
Notation: The number of data n
for $i = 1$ **to** n **do**
  $\bar{s}[i] = \sum_{j=1}^{n} w(i,j)$ # caching initial score
end for
$s = \bar{s}$
repeat
  $\mathcal{N} = \{i \mid s[i] > \lambda, i \in [1, \ldots, n]\}$
  for $i = 1$ **to** n **do**
    $s[i] \leftarrow \bar{s}[i] - 2\sum_{j \in \mathcal{N}} w(i,j)$
  end for
until convergence
Output: $s, \mathcal{N}$

$$s[i] = \mathrm{cut}(\mathcal{N} \cup \{i\}, \mathcal{T} \backslash \mathcal{N} \backslash \{i\}) - \mathrm{cut}(\mathcal{N} \backslash \{i\}, \mathcal{T} \backslash \mathcal{N} \cup \{i\}) = \sum_{j \in \mathcal{T} \backslash \mathcal{N}} w(i,j) - \sum_{j \in \mathcal{N}} w(i,j).$$

Here we use the assumption $w(i,i) = 0$. In practice, the cardinality of $\mathcal{N}$ is small, so we can efficiently update the score vector s by caching the initial score vector $\bar{s}$ as in Algorithm 4. After calculating the score vector s, we update the noisy set $\mathcal{N}$ by selecting nodes with score values above the value λ. Figure 5.4 illustrates the optimization process. Here larger values of λ result in smaller $\mathcal{N}$ consisting of data samples that are more likely to have label noise.

Algorithm 4 satisfies the convergence property in Proposition 9.

Proposition 9. *Algorithm 4 with a single node update at each iteration converges to local optimum.*

Proof. The change in the objective value of Equation (5.2) by moving data i from $\mathcal{T} \backslash \mathcal{N}$ to $\mathcal{N}$ is

$$\sum_{j \in \mathcal{T} \backslash \mathcal{N}} w(i,j) - \sum_{j \in \mathcal{N}} w(i,j) - \lambda, \tag{5.3}$$

where the change by moving data i from $\mathcal{N}$ to $\mathcal{T}\backslash\mathcal{N}$ is

$$\sum_{j\in\mathcal{N}} w(i,j) - \sum_{j\in\mathcal{T}\backslash\mathcal{N}} w(i,j) + \lambda.$$

Note the score s in Algorithm 4 is

$$s[i] = \sum_{j\in\mathcal{T}} w(i,j) - 2\sum_{j\in\mathcal{N}} w(i,j) = \sum_{j\in\mathcal{T}\backslash\mathcal{N}} w(i,j) - \sum_{j\in\mathcal{N}} w(i,j).$$

Thus the change in the objective value by moving data i to another partition is $s[i] - \lambda$ for $i \in \mathcal{T}\backslash\mathcal{N}$ and $-s[i] + \lambda$ for $i \in \mathcal{N}$, which can be represented as $v[i](s[i] - \lambda)$. Therefore, moving a sample with a positive value of $v[i](s[i] - \lambda)$ to another partition guarantees an increase in the objective function value. Because a cut value in a graph is bounded, the algorithm converges to the local optimum by the monotone convergence theorem. □

Complexity analysis. The time complexity of Algorithm 4 is $O(n^2)$, proportional to the number of edges in a graph. It is noteworthy that our method maintains the best performance when used with graphs consisting of a small number of nodes, as shown in Figure 5.5. This implies that we can partition large datasets and run the algorithm repeatedly for each partition to enhance efficiency while maintaining performance. In this case, the complexity becomes $O(n/k \cdot k^2) = O(nk)$, with k representing the size of each partition and n/k being the number of partitions. Also, computations on these partitions are embarrassingly parallelizable, meaning that the complexity becomes $O(k^2)$ for $k \ll n$ in distributed computing environments.

5.3.3 Outlier/OOD detection

In the previous section, we presented a method for detecting label errors based on data relations with similar feature embeddings but different label information. By employing the identical feature embedding structure, we identify outlier data by measuring the extent to which similar data are absent in the feature embedding space. To quantify the extent of a data point being an outlier, we aggregate the similarity kernel values of a data point in Equation (5.4), thereby processing the entire relational information of the data point. Our approach leverages global information about the data distribution, resulting in a more robust performance across a range of experimental settings compared to existing methods that rely on local information such as k-nearest distance [184].

Specifically, for a subset $\mathcal{S} \subseteq \mathcal{T}$ and data x, we measure the ***outlier score*** as

$$\text{outlier}(x) = \frac{1}{\sum_{i \in \mathcal{S}} k(x, x_i)}.$$

Higher values in the outlier score indicate that the data are more distributionally outliers. We propose to use a *uniform random* sampling for $\mathcal{S}$, adjusting the computational cost and memory requirements for the outlier score calculation to suit the inference environment. In Section 5.4.3, we verify our method maintains the best OOD detection performance even when using only 0.4% of the data in ImageNet.

5.3.4 Proposed similarity kernel

For $x_i \in \mathcal{X}$, we extract the feature representation $\mathbf{f}_i$ and the prediction probability vector $\mathbf{p}_i$ from the trained model. We propose a class of bounded kernel $k : \mathcal{X} \times \mathcal{X} \to [0, M]$ with the following form:

$$k(x_i, x_j) = |s(\mathbf{f}_i, \mathbf{f}_j) \cdot c(\mathbf{p}_i, \mathbf{p}_j)|^t. \tag{5.4}$$

A positive scalar value t controls the sharpness of the kernel value distribution. A larger value of t makes a small kernel value smaller, which is effective in handling small noisy kernel values. A scalar value $s(\mathbf{f}_i, \mathbf{f}_j) \in \mathbb{R}^+$ denotes a similarity measurement between features. In our main experiments, we adopt the truncated cosine-similarity that has been widely used in representation learning [170, 165]. We use the hinge function at zero, resulting in the following positive feature-similarity function:

$$s(\mathbf{f}_i, \mathbf{f}_j) = \max(0, \cos(\mathbf{f}_i, \mathbf{f}_j)).$$

It is worth noting the utility of our framework is not limited to a specific kernel design. In Section 5.4.4, we verify our approach maintains the best performance with $s(\mathbf{f}_i, \mathbf{f}_j)$ defined as the RBF kernel [226].

While the feature similarity captures the meaningful semantic relationship between data points, we observe that considering the prediction scores $\mathbf{p}_i$ can further improve the identification of problematic data. To incorporate prediction scores into our approach, we introduce a scalar term $c(\mathbf{p}_i, \mathbf{p}_j)$ that measures the compatibility between the predictions on data points. Any positive and bounded compatibility function is suitable for the kernel class defined in Equation (5.4). In our main experiments, we use the predicted probability of belonging to the same class as the compatibility term $c(\mathbf{p}_i, \mathbf{p}_j)$. Specifically, given the predicted

label random variables $\hat{y}_i$ and $\hat{y}_j$, the proposed compatibility term is

$$c(\mathbf{p}_i, \mathbf{p}_j) = P(\hat{y}_i = \hat{y}_j) = \mathbf{p}_i^\intercal \mathbf{p}_j. \tag{5.5}$$

From a different perspective, we interpret this term as a measure of confidence for feature similarity. In Section 5.4.4, we verify the effectiveness of the compatibility term through an ablation study.

Interpretation. We establish an understanding of our relation function in Equation (5.1) by drawing a connection to the influence function [155]. For simplicity, we consider the influence function with a single checkpoint, where the influence between data x_i and x_j is given by $\nabla_w \ell(x_i)^\intercal \nabla_w \ell(x_j)$. Here, ℓ denotes the loss function, and w denotes the weight of the checkpoint. We consider the influence function at the feed-forward layer, where $\ell(x_i) = h(\mathbf{f}_i') = h(w^\intercal \mathbf{f}_i)$, following the convention [155]. By the chain rule, we can decompose the weight gradient as $\nabla_w \ell(x_i) = \nabla_{\mathbf{f}'} h(\mathbf{f}_i') \mathbf{f}_i^\intercal$, and represent the influence as $\nabla_{\mathbf{f}'} h(\mathbf{f}_i')^\intercal \nabla_{\mathbf{f}'} h(\mathbf{f}_j') \cdot \mathbf{f}_i^\intercal \mathbf{f}_j$. In contrast, our relation function has a form of $1(y_i = y_j) \cdot |s(\mathbf{f}_i, \mathbf{f}_j) \cdot c(\mathbf{p}_i, \mathbf{p}_j)|^t$.

The main distinction between our relation function and the influence function is the existence of the feature gradient term $\nabla_{\mathbf{f}'} h(\mathbf{f}_i')$. As observed in Barshan et al. [8], outliers have a large feature-gradient norm, leading to difficulties in detecting label errors. Specifically, let us consider the weight w at the classifier layer, where the function h is the softmax cross-entropy loss function. As Pruthi et al. [155], we can express the feature gradient inner-product as

$$\nabla_{\mathbf{f}'} h(\mathbf{f}_i')^\intercal \nabla_{\mathbf{f}'} h(\mathbf{f}_j') = (\mathbf{y}_i - \mathbf{p}_i)^\intercal (\mathbf{y}_j - \mathbf{p}_j),$$

where $\mathbf{y}_i$ denote the one-hot label. The equation above shows that the correctly classified data with $\mathbf{y}_i \approx \mathbf{p}_i$ yields near zero inner-product values, whereas outliers with high entropy predictions exhibit large inner-product values. Consequently, existing influence-based label error detection methods, which detect label errors by identifying data with high influence values, have degraded performance in the presence of outliers [8].

Our relation function differs from influence functions in that it separates label and prediction information using a label comparison term $1(y_i = y_j)$ and a compatibility term $c(\mathbf{p}_i, \mathbf{p}_j)$, respectively. Outlier data typically have a high entropy of model predictions, resulting in lower compatibility values with other data [64]. On the other hand, normal data with label errors exhibit high compatibility values with other normal data. Our detection algorithms exploit these differences and achieve improved detection performance compared to the influence functions.

Model	Feature	Algorithm 4	Gradient
MAE-Base	2300	400	6000
MAE-Large	6900	420	21000

Table 5.1 Time spent (s) for label error detection on ImageNet 1.2M dataset. *Feature* indicates the total computing time for calculating feature embeddings of all data points, and *Gradient* means the total computing time for calculating network gradient on each data point. *Algorithm 1* indicates the time spent by our algorithm, excluding feature calculation.

Model	Unary	Relation
MAE-Large	12.2	12.3
ResNet-50	8.1	8.2

Table 5.2 Time spent per sample (ms) for OOD detection on ImageNet with various models. *Unary* refers to unary scoring methods utilizing logit or probability score for each data point.

5.3.5 Computation time analysis

In this section, we measure the time spent on detection algorithms. We use 1 RTX3090-Ti GPU and conduct experiments on the full ImageNet training set. Table 5.1 compares computation time for Algorithm 4 and feature calculation. Note that all existing methods using neural networks require at least the computation of data features $\mathbf{f}_i$. Table 5.1 shows that Algorithm 4 (excluding feature calculation) requires significantly less computation time than the feature calculation. We also observe that our algorithm efficiently scales up to large neural networks, MAE-Large, which have a larger number of feature embedding dimensions than the base model. It is also worth noting that computing the network gradient takes a much longer time, demonstrating the efficiency of our algorithm in large-scale label error detection compared to the existing methods.

In Table 5.2, we measure the time spent for OOD detection on the full ImageNet training set. The computation of our similarity kernel is embarrassingly parallelizable on GPUs. As shown in the table, the overhead time for computing our outlier scores is negligible compared to the time spent for the neural networks' forward computation on a single data point. We can further reduce the time and memory requirements by measuring the outlier score on a subset of the training set (Figure 5.7).

5.3.6 Data relation map

In this section, we present a visualization method based on our data relation function to contextualize data and comprehend its relational structure. One of the effective approaches for visualizing a dataset is dataset cartography [185], which projects the dataset onto a 2D plot. This approach draws a scatter plot of the mean and standard deviation of the model's prediction probabilities for each data sample during training. Inspired by the dataset cartography, we propose a *data relation map*, which visualizes the relationship between data along the training process. To this end, we uniformly store checkpoints during training. We denote a set of these checkpoints as $\mathcal{K}$, where r_k refers to the relation function for checkpoint $k \in \mathcal{K}$. For each data sample $i \in \mathcal{T}$, we draw a scatter plot of the mean and standard deviation of relation values $\{r_k(i,j) \mid k \in \mathcal{K}\}$ for $j \in \mathcal{T} \backslash \{i\}$.

In Figure 5.1, we provide relation maps of three samples from ImageNet, using 10 checkpoints of MAE-Large [62]. The three samples each represent clean data, data with a label error, and outlier data. From the figure, samples show different relation map patterns. Specifically, the relation map of a clean data sample exhibits a majority of positive relations with relatively small variability. We note that there are gray-colored relations in high variability regions (0.2<std), indicating that the model resolves conflicting relations at convergence. On the other hand, the relation map of the sample with a label error demonstrates a majority of negative relations. Notably, high variance relations result in largely negative relations at convergence, suggesting that conflicts intensify. Lastly, the relation map of the outlier data sample reveals that relations are close to 0 during training. These relation maps can serve as a model-based fingerprint of the data, which our algorithm effectively exploits to identify problematic data.

5.4 Experimental results

In this section, we experimentally verify the effectiveness of our approach in detecting label errors and outliers.

5.4.1 Setting

Datasets. We conduct label error detection experiments on large-scale datasets: ImageNet [38], ESC-50 [152], and SST2 [199]. ImageNet consists of about 1.2M image data from 1,000 classes. ESC-50 consists of 2,000 5-second environmental audio recordings organized into 50 classes. SST2 is a binary text sentiment classification dataset, consisting of 67k movie review sentences.

Label Noise Ratio	0.	0.04	0.08	0.12	0.15
Top-1 Accuracy	85.89	84.96	84.15	82.88	81.50

Table 5.3 Validation top-1 accuracy of MAE-Large trained on ImageNet with noisy labels.

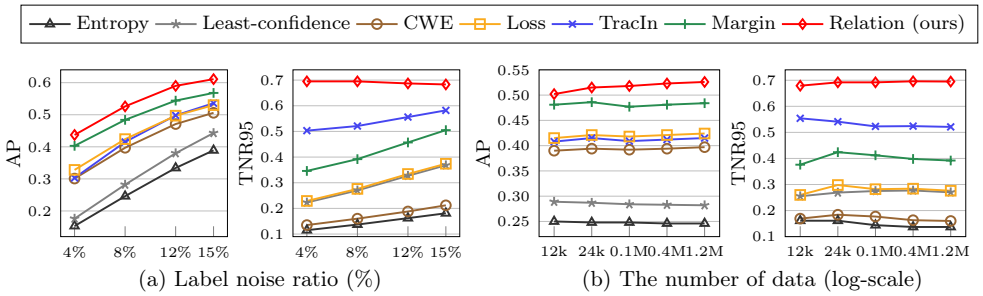
Following Pruthi et al. [155], we construct a noisy training set by flipping labels of certain percentages of correctly classified training data with the top-2 prediction of the trained model. We use different neural network architectures for constructing a noisy training set and detecting label errors to avoid possible correlation. In Table 5.3, we provide the top-1 validation accuracy of MAE-Large trained on training sets with noisy labels, demonstrating the importance of label noise detection and cleaning.

Baselines. We compare our method (*Relation*) to six baselines that are suitable for large-scale datasets. We consider fine-tuned loss from pre-trained models (*Loss*) [27], prediction probability margin score (*Margin*) [139], and the influence-based approach called *TracIn* [155]. We also evaluate model-agnostic scoring methods: *Entropy*, *Least-confidence*, and Confidence-weighted Entropy (*CWE*) [105]. For a fair comparison, we evaluate methods using a single converged neural network in our main experiments, while also providing results with a temporal ensemble suggested by [155] in Table 5.5.

Metric. We evaluate the detection performance based on label reliability scores by each method. We note that detecting label errors is an imbalanced detection problem, which makes the AUROC metric prone to being optimistic and misleading [37]. In this respect, we mainly report the AP (average precision) and TNR95 (TNR at 0.95 TPR).

5.4.2 Label error detection

ImageNet. We measure the label error detection performance on ImageNet with the synthetic label noise by training an MAE-Large model [62]. Note that the model does not have access to information about the changed clean labels during the entire training process. Figure 5.5 (a) shows the detection performance over a wide range of label noise ratios from 4% to 15%. As shown in the figure, our approach achieves the best AP and TNR95 performance compared to the baselines. Especially, our method maintains a high TNR95 over a wide range of noise ratios, indicating that the number of data that need to be re-



(a) Label noise ratio (%) (b) The number of data (log-scale)

Figure 5.5 Label error detection performance on ImageNet with MAE-Large according to (a) label noise ratios and (b) the number of data. We obtain the results in (b) with 8% label noise.

viewed by human annotators is significantly smaller when cleaning the dataset. In Figure 5.10, we present detected label error samples by our algorithm.

It is worth noting that our method relies on the number of data for constructing a relation graph. To measure the sensitivity of our algorithm to the number of data, we evaluate the detection performance using a reduced number of data with uniform random sampling. Figure 5.5 (b) shows the detection performance on 8% label noise with MAE-Large. From the figure, we find that our algorithm maintains the best detection performance even with 1% of the data (12k). This demonstrates that our algorithm is effective even when only a small portion of the training data is available, such as continual learning or federated learning [145, 131]. In Table 5.4 (a), we provide detection performance for different scales of MAE models on 8% label noise. The table shows our approach achieves the best AP with MAE-Base, verifying the robustness of our approach to the network scales. From the table, we note that larger models are more robust to label noise and show better detection performance.

Speech and language domains. We apply our method to speech and language domain datasets: ESC-50 [152] and SST2 [199]. We design the label noise detection settings identical to the previous ImageNet section. Specifically, we train the AST model [50] for ESC-50 and the RoBERTa-Base model [120] for SST2 under the 10% label noise setting. Table 5.4 (b) shows our approach achieves the best AP and TNR95 on the speech and language datasets, demonstrating the generality of our approach across various data types.

Realistic label noise. The ImageNet validation set is known to contain numerous label errors [140]. To tackle this issue, Beyer et al. [11] cleaned the labels

(a) Model architecture scales

Scale	Metric	Baseline	Relation
Base	AP	0.477	**0.514**
	TNR95	0.488	**0.672**
Large	AP	0.484	**0.526**
	TNR95	0.521	**0.695**

(b) Speech/language domains

Dataset	Metric	Baseline	Relation
ESC50	AP	0.739	**0.779**
	TNR95	0.793	**0.847**
SST2	AP	0.861	**0.881**
	TNR95	0.850	**0.870**

(c) Realistic label noise

Model	Baseline	Relation
MAE	0.708	**0.733**
BEIT	0.719	**0.737**
ConvNeXt	0.713	**0.735**
ConvNeXt-22k	0.724	**0.744**

Table 5.4 Label error detection performance on ImageNet with 8% label noise. *Baseline* refers to the *best* performance among the six baselines considered in Figure 5.5. In Table (c), the evaluation metric is AP.

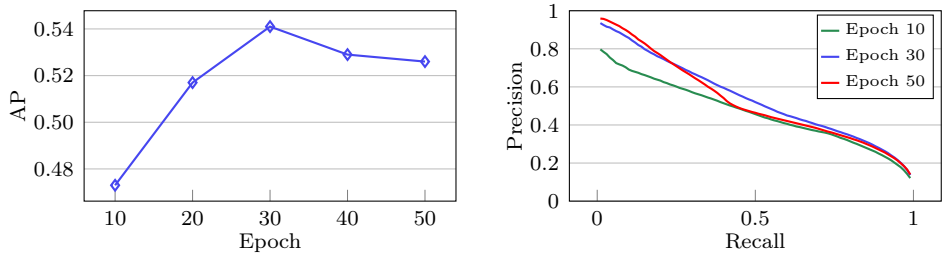


Figure 5.6 Label error detection performance of relation graph throughout the MAE-Large training process on ImageNet with 8% label noise.

with human experts and corrected around 29% of the labels via multi-labeling. With this re-labeled validation set, we conduct experiments under the realistic label noise, with the task of detecting the data samples with changed labels. We measure the detection performance with MAE-Large [62], BEIT-Large [7], and ConvNeXt-Large [121] models. To examine the impact of pre-training on external data, we also include ConvNeXt pre-trained on ImageNet-22k, denoted as ConvNeXt-22k. We construct the relation graph using only the validation set, considering scenarios where the training data are not available. Table 5.4 (c) verifies that our approach outperforms the best baseline across various models. The results on ConvNeXt-22k indicate that pre-training on external data improves the detection performance.

Memorization issue. We investigate the impact of large neural networks' ability to memorize label errors on detection performance [231]. In the left figure of Figure 5.6, we find that the AP score decreases as the training progresses after 30 epochs with MAE-Large which converges at 50 epochs. From the precision-recall curves in Figure 5.6, we observe that precision increases at low recall area but decreases at mid-level recall (~ 0.5) as the training progresses. This suggests that training has both positive and negative effects on detecting label

Entropy	Least-conf.	CWE	Loss	TracIn	Margin	Relation
0.246 (0.007)	0.282 (0.001)	0.397 (0.031)	0.465 (0.041)	0.449 (0.034)	0.544 (0.06)	**0.562** (0.036)

Table 5.5 Label error detection AP with the temporal model ensemble on ImageNet with 8% label noise (MAE-Large). In parenthesis, we denote the performance gain compared to the detection by a single converged model.

noise, and we speculate that memorization is one cause.

Leveraging these observations, we measure the label error detection performance by using the temporal model ensemble [155]. Specifically, we average the label reliability scores from 4 checkpoints that are uniformly sampled throughout training. Table 5.5 shows that this technique improves the performance of all methods, with our approach still exhibiting the best performance. These results confirm the effectiveness of temporal ensembles when more computation and storage are available.

5.4.3 Outlier/OOD detection

Baselines. We consider the following representative outlier scoring approaches: Maximum Softmax Probability (*MSP*) [64], *Max Logit* [66], *Mahalanobis* [109], *Energy* score [119], *ReAct* [183], *KL-Matching* [66], and *KNN* [184]. We tune the KNN method's hyperparameter k based on the paper's guidance as $k = 1000 \times \alpha$, where α represents the ratio of training data used for OOD detection. We also evaluate outlier detection approaches, *Iterative sampling* [182] and *Local outlier factor* [204].

OOD detection. Following Sun et al. [184], we evaluate OOD detection performance on the ImageNet validation set consisting of 50k in-distribution data samples, along with four distinct OOD datasets: *Places* [244], *SUN* [219], *iNaturalist* [194], and *Textures* [29]. Each of these OOD datasets consists of 10k data samples except for Textures which has 5,640 data samples. We also combine these four datasets, denoted as *ALL*, and measure the overall OOD detection performance on this dataset.

Figure 5.7 shows OOD detection performance of MAE-Large on ALL outlier dataset. Our approach and KNN both rely on the number of training data samples ($|\mathcal{S}|$) for outlier score calculation. We examine the effect of training set size by measuring the performance with a reduced number of data using uniform random sampling. Figure 5.7 verifies that our approach outperforms baselines while maintaining performance even with 0.4% of the training dataset (5k). Note that KNN requires hyperparameter tuning according to the training set

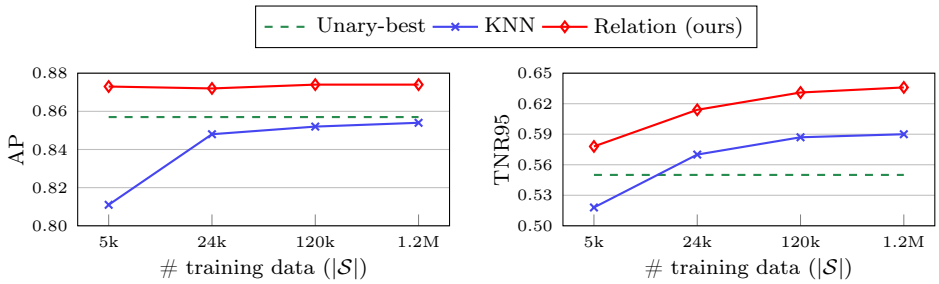


Figure 5.7 OOD detection performance on ImageNet (ALL) with MAE-Large. *Unary-best* means the best performance among the methods that do not rely on the training data for outlier score calculation.

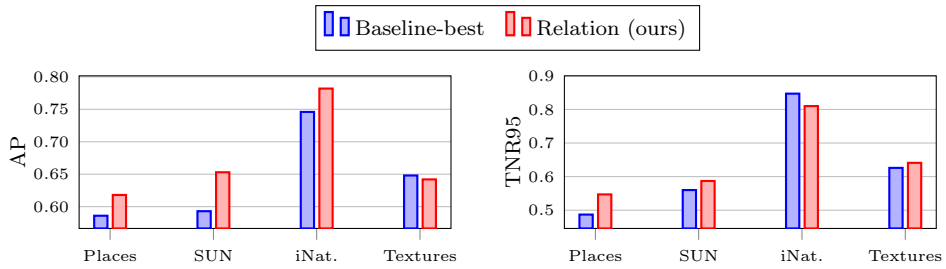


Figure 5.8 OOD detection performance on individual ImageNet OOD datasets with MAE-Large. *Baseline-best* refers to the best performance among the nine baselines.

size, whereas our approach uses the identical hyperparameter ($t = 1$) regardless of the size. Figure 5.8 shows performance on four individual OOD datasets.

Outlier detection. We perform outlier detection experiments following the methodology by Wang et al. [204], where the training set contains outlier data with random labels. We construct the noisy ImageNet-100 training sets by using SUN [219] datasets. We train a ViT-Base model [41] from scratch on these noisy training datasets, and measure outlier detection performance using the trained model.

Table 5.6 shows the outlier detection results on two outlier datasets. As indicated, our method achieves the best performance in the outlier setting, demonstrating its effectiveness in outlier detection. It is worth noting that the considered OOD scoring methods (MSP, Max Logit, Energy) do not achieve good outlier detection performance. We speculate that this is due to the over-

Method	AUROC	AP	TNR95
MSP	0.708	0.335	0.032
Max Logit	0.499	0.216	0.011
Energy	0.417	0.106	0.010
KNN	0.990	0.899	0.960
Iterative sampling	0.973	0.687	0.903
Local outlier factor	0.986	0.850	0.941
Relation (Ours)	**0.993**	**0.906**	**0.971**

Table 5.6 Outlier detection performance with Vit-Base on noisy ImageNet-100 with SUN. Some OOD scoring methods (Mahalanobis, ReAct, KL-Matching) are excluded from the comparison because they require a clean training dataset which is not available in the outlier detection setup.

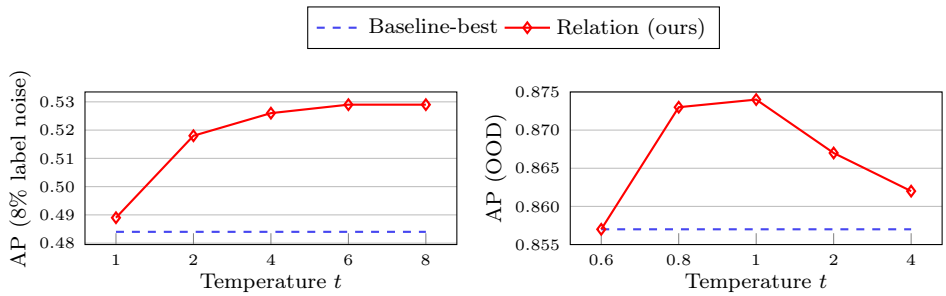


Figure 5.9 The detection AP of MAE-Large across a range of kernel temperatures t. The dashed blue line means the best baseline performance.

fitting of the neural network's predictions on outliers.

Detecting outliers in validation set. We further utilize our method for identifying outliers in the validation set by retrieving data samples with the highest outlier score (Section 5.3.3). In Figure 5.11, we present samples detected by our algorithm from ImageNet and SST2. In the figure, we observe that these samples are not suitable for measuring the predictive performance on labels, which should be excluded from the evaluation dataset.

5.4.4 Ablation study

Temperature t. In Equation (5.4), we introduced a temperature t, where a large value of t increases the influence of large relation values in our algorithm. We conduct sensitivity analysis on t with MAE-Large on ImageNet under 8%

Metric	Baseline	RBF	Cos	RBF $\cdot c$	Cos $\cdot c$
AP	0.484	0.470	0.471	0.525	0.526
TNR95	0.521	0.668	0.671	0.703	0.695

Table 5.7 Comparison of similarity kernel designs. *Baseline* represents the best baseline performance. The term c denotes our compatibility term in Equation (5.5). Note, $Cos \cdot c$ is the kernel function considered in our main experiments, and *RBF / Cos* refers to our method without the compatibility term c.

label noise. Figure 5.9 shows the effect of the temperature value on our detection algorithm's performance. From the figure, we observe that the label error detection performance increases as the t value increases, saturating at $t = 6$. In the case of OOD detection, we achieve the best performance at around $t = 1$. Our algorithm outperforms the best baseline over a wide range of hyperparameters, demonstrating the robustness of our algorithm to the hyperparameter.

Similarity kernel design. We present an empirical analysis of the kernel design choices. Specifically, we replace the cosine similarity term in Equation (5.4) as the RBF kernel and evaluate the detection performance. We further conduct an ablation study on compatibility terms (Equation (5.5)). Table 5.7 summarizes the label error detection performance with different kernel functions on ImageNet with 8% noise ratio. The table shows that our approach largely outperforms the best baseline even with the RBF kernel. Also, we find that our approach without the compatibility term shows comparable AP performance while significantly outperforming baselines in TNR95. These results demonstrate the generality and utility of our relational structure-based framework, which is not limited to a specific kernel design.

5.5 Additional discussions

Why does relation graph work? We discuss the conceptual differences between our relation graph-based approach and previous baselines. Firstly, our method is data-centric, whereas the previous approaches rely on a unary score by models. Our approach identifies problematic data by comparing them to other data, which leads to more reliable identification of problematic data than unary scoring methods that are vulnerable to overfitting [153]. Secondly, our approach aggregates global relational information, whereas previous methods rely on local information such as k-nearest distance [184]. Considering all edge connections, our method obtains more representative information about the

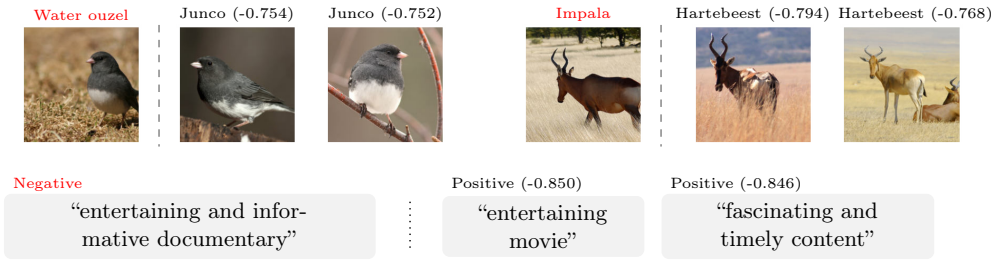


Figure 5.10 Detected data samples with label errors (marked in red) from ImageNet (top) and SST2 (bottom). We present samples with conflicting relations next to the detected samples and denote the corresponding relation value in parenthesis.

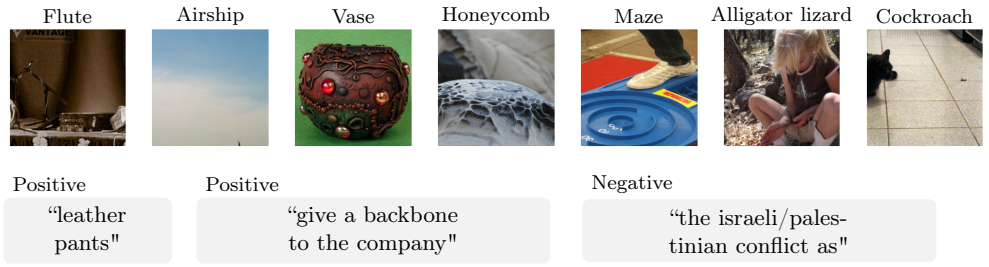


Figure 5.11 Data samples with the highest outlier scores by our method on ImageNet (top) and SST2 (bottom) validation sets. We denote the assigned labels above each data sample.

data distribution. Through temperature parameter t and efficient graph algorithms, we effectively process the entire relations and achieve the improved identification of problematic data.

Limitations and future works. There are several promising future directions for our work. Firstly, the current experiments are limited to the classification task, and it would be valuable to apply our approach to a wider range of tasks, such as segmentation or generative models. These tasks may introduce new and interesting categories of problematic data arising from different label spaces and data structures. Secondly, integrating our method with human annotation and model training processes will also be valuable. This could involve using our approach to identify inconsistencies in label assignments or to conduct a fine-grained evaluation of models.

5.6 Conclusion

In this paper, we propose a novel data relation function and graph algorithms for detecting label errors and outlier data using the relational structure of data in the feature embedding space. Our approach achieves state-of-the-art performance in both label error and outlier/OOD detection tasks, as demonstrated through extensive experiments on large-scale benchmarks. Furthermore, we introduce a data contextualization tool based on our data relation that can aid in data diagnosis. Our algorithms and tools can facilitate the analysis of large-scale datasets, which is crucial for the development of robust machine-learning systems.

Chapter 6

Dataset Condensation via Efficient Synthetic-Data Parameterization

6.1 Introduction

Deep learning has achieved great success in various fields thanks to the recent advances in technology and the availability of massive real-world data [108]. However, this success with massive data comes at a price: huge computational and environmental costs for large-scale neural network training, hyperparameter tuning, and architecture search [148, 16, 34, 245].

An approach to reduce the costs is to construct a compact dataset that contains sufficient information from the original dataset to train models. A classic approach to construct such a dataset is to select the *coreset* [151]. However, selection-based approaches have limitations in that they depend on heuristics and assume the existence of representative samples in the original data [238]. To overcome these limitations, recent studies, called *dataset condensation* or *dataset distillation*, propose to synthesize a compact dataset that has better storage efficiency than the coresets [202]. The synthesized datasets have a variety of applications such as increasing the efficiency of replay exemplars in continual learning and accelerating neural architecture search [239].

The natural data satisfy regularity conditions that form a low-rank data subspace [73], *e.g.*, spatially nearby pixels in a natural image look similar and temporally adjacent signals have similar spectra in speech [235]. However, the existing condensation approaches directly optimize each data element, *e.g.,*

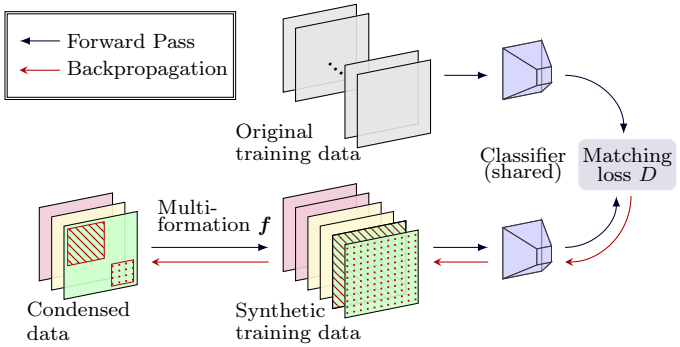


Figure 6.1 Illustration of the proposed dataset condensation framework with multi-formation. Under the fixed-size storage for condensed data, multi-formation synthesizes multiple data used to train models. We optimize the condensed data in an end-to-end fashion by using the differentiable multi-formation functions.

pixel by pixel, without imposing any regularity conditions on the synthetic data [138, 237]. Under the limited storage budget, this inefficient parameterization of synthetic datasets results in the synthesis of a limited number of data, having fundamental limitations on optimization. Furthermore, optimizing the synthetic data that have comparable training performance to the original data is challenging because it requires unrolling the entire training procedure. Recent studies propose surrogate objectives to address the challenge above, however, there are remaining questions on why certain objectives are better proxies for the true objective [237, 238].

In this work, we pay attention to making better use of condensed data elements and propose a novel optimization framework resolving the previous limitations. Specifically, we introduce a *multi-formation* process that creates multiple synthetic data under the same storage constraints as existing approaches (Figure 6.1). Our proposed process naturally imposes regularity on synthetic data while increasing the number of synthetic data, resulting in an enlarged and regularized dataset. In Section 6.3.3, we theoretically analyze the multi-formation framework and examine the conditions where the improvement is guaranteed. We further analyze the optimization challenges in the gradient matching method by Zhao and Bilen [238] in Section 6.4. Their approach induces imbalanced network gradient norms between synthetic and real data, which is problematic during optimization. Based on our analysis and empirical findings, we develop improved optimization techniques utilizing networks trained on the real data with stronger regularization and effectively mitigate the mentioned

problems.

In this regard, we present an end-to-end optimization algorithm that creates information-intensive condensed data significantly outperforming all existing condensation methods. Given fixed storage and computation budgets, neural networks trained on our synthetic data show performance improvements of 10∼20%p compared to state-of-the-art methods in experimental settings with various datasets and domains including ImageNet and Speech Commands [206]. We further verify the utility of our condensed data through experiments on continual learning, demonstrating significant performance improvements compared to existing condensation and coreset methods. We release the source code at `https://github.com/snu-mllab/Efficient-Dataset-Condensation`.

6.2 Preliminary

Given the storage budget, the goal of data condensation is to build a surrogate dataset $\mathcal{S}$ of the original training dataset $\mathcal{T}$ such that an arbitrary model trained on $\mathcal{S}$ is similar to the one trained on $\mathcal{T}$ [202]. Oftentimes, the measure of similarity is in terms of the model performance on the test set because that leads to meaningful applications such as continual learning and neural architecture search [239]. Instead of solving this ultimate objective, previous methods have proposed different surrogates. For example, Wang et al. [202] propose to optimize $\mathcal{S}$ such that a model trained on $\mathcal{S}$ minimizes the loss values over $\mathcal{T}$. However, this approach involves a nested optimization with unrolling multiple training iterations, requiring expensive computation costs.

Rather than direct optimization of model performance, Zhao et al. [239] propose a simpler optimization framework that matches the network gradients on $\mathcal{S}$ to the gradients on $\mathcal{T}$. Let us assume a data point is m-dimensional and $\mathcal{S} \in \mathbb{R}^{n \times m}$, where n is the number of data points in $\mathcal{S}$. Zhao et al. [239] optimize the synthetic data as

$$\begin{aligned} &\underset{\mathcal{S} \in \mathbb{R}^{n \times m}}{\text{maximize}} \sum_{t=0}^{\tau} \text{Cos}\left(\nabla_\theta \ell(\theta_t; \mathcal{S}), \nabla_\theta \ell(\theta_t; \mathcal{T})\right) \\ &\text{subject to } \ \theta_{t+1} = \theta_t - \eta \nabla_\theta \ell(\theta_t; \mathcal{S}) \ \text{ for } t = 0, \ldots, \tau - 1, \end{aligned} \tag{6.1}$$

where θ_t denotes the network weights at t^{th} training step from the randomly initialized weights θ_0 given $\mathcal{S}$, $\ell(\theta; \mathcal{S})$ denotes the training loss for weight θ and the dataset $\mathcal{S}$. $\text{Cos}(\cdot, \cdot)$ denotes the channel-wise cosine similarity. Zhao et al. [239] have reported that the class-wise gradient matching objective is effective for dataset condensation. They propose an alternating optimization algorithm with the following update rules for each class c:

$$S_c \leftarrow S_c + \lambda \nabla_{S_c} \mathrm{Cos}\left(\nabla_\theta \ell(\theta; S_c), \nabla_\theta \ell(\theta; T_c)\right)$$
$$\theta \leftarrow \theta - \eta \nabla_\theta \ell(\theta; \mathcal{S}),$$

where S_c and T_c denote the mini-batches from the datasets $\mathcal{S}$ and $\mathcal{T}$, respectively. Under the formulation, Zhao and Bilen [238] propose to utilize differentiable siamese augmentation (DSA) for a better optimization of the synthetic data. DSA performs gradient matching on augmented data where the objective becomes $\mathbb{E}_{\omega \sim \mathcal{W}}\left[\mathrm{Cos}\left(\nabla_\theta \ell(\theta; a_\omega(\mathcal{S})), \nabla_\theta \ell(\theta; a_\omega(\mathcal{T}))\right)\right]$. Here, a_ω means a parameterized augmentation function and $\mathcal{W}$ denotes an augmentation parameter space. Subsequently, Zhao and Bilen [237] propose to match the hidden features rather than the gradients for fast optimization. However, the feature matching approach has some performance degradation compared to gradient matching [237]. Although this series of works have made great contributions, there are remaining challenges and questions on their surrogate optimization problems. In this work, we try to resolve the challenges by providing a new optimization framework with theoretical analysis and empirical findings.

6.3 Multi-Formation Framework

In this section, we pay attention to the synthetic-data parameterization in optimization and present a novel data formation framework that makes better use of condensed data. We first provide our motivating observations and introduce a multi-formation framework with theoretical analysis.

6.3.1 Observation

We first provide our empirical observations on the effects of the number and resolution of the synthetic data in the matching problem. The existing condensation approaches aim to synthesize a predetermined number of data about 10 to 50 per class [238, 138]. The left subfigure in Figure 6.2 shows the condensation matching loss curves of DSA over various numbers of synthetic data per class. As shown in the figure, more synthetic data lead to a smaller matching loss, indicating the importance of the number of synthetic data in the matching problem. For a comparison under the same data storage budget, we measure the matching loss on the same network after reducing the resolution of the optimized synthetic data and resizing the data to the original size. In the right subfigure in Figure 6.2, we find the resolution produces a moderate change in matching loss as the number of data does, even if we do not take the resolution modification into account during the condensation stage. For example, points

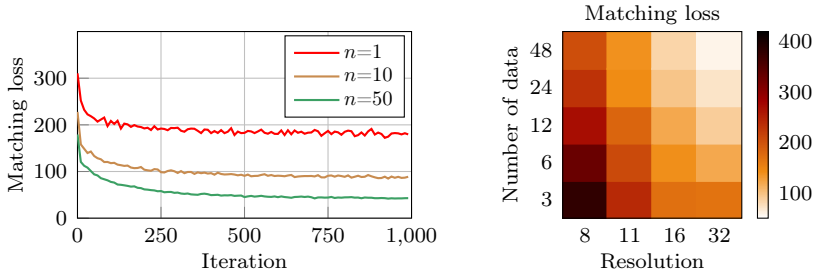


Figure 6.2 (Left) Matching loss curves over an increasing number of synthetic data per class (n). (Right) Matching loss heat map over various resolutions and numbers of data per class. The x-axis refers to the downsampled image resolution. We measure values on the same network after resizing data to the original size (CIFAR-10).

at (16, 48) and (32, 12), which require an equal storage size, have similar loss values. Motivated by these results, we propose a multi-formation framework that makes better use of the condensed data and forms the increased number of synthetic data under the same storage budget.

6.3.2 Multi-Formation

The existing approaches directly match condensed data $\mathcal{S}$ to the original training data $\mathcal{T}$ and use $\mathcal{S}$ as the synthetic training data. Instead, we add an intermediate process that creates an increased number of synthetic data from $\mathcal{S}$ by mapping a data element in $\mathcal{S}$ to multiple data elements in the synthetic data (Figure 6.1). The previous work by Zhao and Bilen [238] reports that the use of random augmentations in matching problems degrades performance due to the misalignment problem. They argue the importance of the deterministic design of the matching problem. In this regard, we propose to use a deterministic process rather than a random process.

Consistent to existing approaches, we optimize and store condensed data $\mathcal{S} \in \mathbb{R}^{n \times m}$. For $n' > n$, we propose a multi-formation function $f : \mathbb{R}^{n \times m} \to \mathbb{R}^{n' \times m}$ that augments the number of condensed data $\mathcal{S}$ and creates multiple synthetic training data $f(\mathcal{S})$ in a deterministic fashion. For any matching objective D (lower the better) and target task objective ℓ, the optimization and evaluation stages of condensed data $\mathcal{S}$ with multi-formation function f are

$$\mathcal{S}^* = \underset{\mathcal{S} \in \mathbb{R}^{n \times m}}{\operatorname{argmin}} D(f(\mathcal{S}), \mathcal{T}) \qquad \text{(Optimization)}$$

$$\theta^* = \underset{\theta}{\operatorname{argmin}}\, \ell(\theta; f(\mathcal{S}^*)). \qquad \text{(Evaluation)}$$

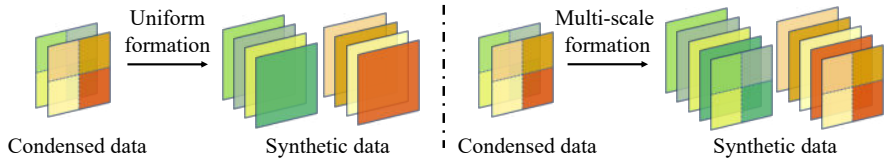


Figure 6.3 Illustration of the proposed multi-formation functions, in the case of multi-formation by a factor of 2.

That is, we perform matching on $\mathcal{T}$ using $f(\mathcal{S})$ and use them for evaluation. This enables us to optimize the synthetic dataset with an increased number of data, using the same storage budget. Figure 6.1 illustrates the optimization process with multi-formation. Note, we can use conventional data augmentations following the multi-formation.

Given a differentiable multi-formation function and matching objective, we optimize $\mathcal{S}$ in an end-to-end fashion by gradient descent. In this work, we design a simple differentiable multi-formation function and evaluate the effectiveness of our approach. The idea is to locally interpolate data elements while preserving the locality of natural data, *i.e.*, spatially nearby pixels in a natural image look similar and temporally adjacent signals have similar spectra in speech [73, 235]. Specifically, we partition each data and resize the partitioned data to the original size by using bilinear upsampling (Figure 6.3). Note, this formation function has negligible computation overhead. Furthermore, the formation function creates locally smooth synthetic data that might naturally regularize the optimization from numerous local minima. We use a fixed uniform partition function in our main experiments in Section 6.5.

6.3.3 Theoretical Analysis

In this section, we aim to theoretically analyze our multi-formation framework. Here, we assume a data point is m-dimensional. The natural data have regularity that makes difference from random noise [73]. We assume that data satisfying this regularity form a subspace $\mathcal{N} \subset \mathbb{R}^m$. That is, the original training dataset $\mathcal{T} = \{t_i\}_{i=1}^{n_t}$ satisfies $t_i \in \mathcal{N}$ for $i = 1, \dots, n_t$. With abuse of notation, we denote the space of datasets with n data points as $\mathbb{R}^{n\times m} = \{\{d_i\}_{i=1}^{n} \mid d_i \in \mathbb{R}^m \text{ for } i = 1, \dots, n\}$. We further define the space of all datasets $\mathcal{D} = \cup_{n\in\mathbb{N}}\mathbb{R}^{n\times m}$ and the synthetic-dataset space of a multi-formation function $f : \mathbb{R}^{n\times m} \to \mathbb{R}^{n'\times m}$, $\mathcal{M}_f = \{f(\mathcal{S}) \mid \mathcal{S} \in \mathbb{R}^{n\times m}\}$. We now introduce our definition of distance measure between datasets. We say data d is closer to dataset $X = \{d_i\}_{i=1}^{k}$ than d', if $\forall i \in [1, \dots, k],\ \|d - d_i\| \leq \|d' - d_i\|$.

Definition 5. *A function $D : \mathcal{D} \times \mathcal{D} \to [0, \infty)$ is a dataset distance measure, if it satisfies the followings: $\forall X, X' \in \mathcal{D}$ where $X = \{d_i\}_{i=1}^k$, $\forall i \in [1, \ldots, k]$,*

1. *$D(X, X) = 0$ and $D(X, X') = D(X', X)$.*
2. *$\forall d \in \mathbb{R}^m$ s.t. d is closer to X' than d_i, $D(X \setminus \{d_i\} \cup \{d\}, X') \leq D(X, X')$.*
3. *$D(X, X' \cup \{d_i\}) \leq D(X, X')$.*

The definition above states reasonable conditions for dataset distance measurement. Specifically, the second condition states that the distance decreases if a data point in a dataset moves closer to the other dataset. The third condition states that the distance decreases if a data point in a dataset is added to the other dataset. Based on the definition, we introduce the following proposition.

Proposition 10. *If $\mathcal{N}^{n'} \subseteq \mathcal{M}_f$, then for any dataset distance measure D,*

$$\min_{\mathcal{S} \in \mathbb{R}^{n \times m}} D(f(\mathcal{S}), \mathcal{T}) \leq \min_{\mathcal{S} \in \mathbb{R}^{n \times m}} D(\mathcal{S}, \mathcal{T}).$$

Proof. For simplicity, we denote $[1, \ldots, n]$ as $[n]$. Let us denote $\mathcal{T} = \{t_i\}_{i=1}^{n_t}$ and $\mathcal{S} = \{s_j\}_{j=1}^{n}$, where $t_i \in \mathcal{N} \subset \mathbb{R}^m$ and $s_j \in \mathbb{R}^m$, $\forall i \in [n_t]$ and $\forall j \in [n]$. Under the assumption that $\mathcal{N}$ is a subspace of $\mathbb{R}^m$, there exists the projection of s_j onto $\mathcal{N}$, $\bar{s}_j \in \mathcal{N}$. Because $t_i \in \mathcal{N}$ for $i = 1, \ldots, n_t$, $\|\bar{s}_j - t_i\| \leq \|s_j - t_i\|$, $\forall j \in [n]$ and $\forall i \in [n_t]$. This means the projection $\bar{s}_j$ is closer to $\mathcal{T}$ than s_j, $\forall j \in [n]$. Let us define a partially projected dataset $\bar{\mathcal{S}}_k = \{\bar{s}_j\}_{j=1}^{k} \cup \{s_j\}_{j=k+1}^{n}$. Then by the second axiom,

$$D(\bar{\mathcal{S}}_n, \mathcal{T}) \leq D(\bar{\mathcal{S}}_{n-1}, \mathcal{T}) \leq \ldots \leq D(\mathcal{S}, \mathcal{T}).$$

This result means that the optimum $\mathcal{S}^* = \operatorname{argmin} D(\mathcal{S}, \mathcal{T})$ satisfies $\mathcal{S}^* \in \mathcal{N}^n$. Note our multi-formation augments the number of data from n to n' where $n < n'$. Let us denote $k' = n' - n$ and $S^*_{add} = \mathcal{S}^* \cup \{t_i\}_{i=1}^{k'}$. By the third axiom,

$$D(S^*_{add}, \mathcal{T}) \leq D(\mathcal{S}^*, \mathcal{T}).$$

The elements of S^*_{add} lie in $\mathcal{N}$ and $S^*_{add} \in \mathcal{N}^{n'}$. From the assumption $\mathcal{N}^{n'} \subseteq \mathcal{M}_f$, $\exists \mathcal{S} \in \mathbb{R}^{n \times m}$ s.t. $f(\mathcal{S}) = S^*_{add}$. Thus,

$$\begin{aligned} \min_{\mathcal{S} \in \mathbb{R}^{n \times m}} D(f(\mathcal{S}), \mathcal{T}) &\leq D(S^*_{add}, \mathcal{T}) \\ &\leq D(\mathcal{S}^*, \mathcal{T}) = \min_{\mathcal{S} \in \mathbb{R}^{n \times m}} D(\mathcal{S}, \mathcal{T}). \end{aligned}$$

□

Proposition 10 states that our multi-formation framework achieves the better optimum, *i.e.*, the synthetic dataset that is closer to the original dataset under any dataset distance measure. Note, the assumption $\mathcal{N}^{n'} \subseteq \mathcal{M}_f$ means that the synthetic-dataset space by f is sufficiently large to contain all data points in $\mathcal{N}$.

6.4 Improved Optimization Techniques

In this section, we develop optimization techniques for dataset condensation. We first analyze gradient matching [238] and seek to provide an interpretation of why gradient matching on condensation works better than feature matching [237]. We then examine some of the shortcomings of existing gradient matching methods and propose improved techniques.

6.4.1 Interpretation

Convolutional or fully-connected layers in neural networks linearly operate on hidden features. From the linearity, it is possible to represent network gradients as features as in Proposition 11. For simplicity, we consider one-dimensional convolution on hidden features and drop channel notations.

Proposition 11. *Let $w_t \in \mathbb{R}^K$ and $h_t \in \mathbb{R}^W$ each denote the convolution weights and hidden features at the t^{th} layer given the input data x. Then, for a loss function ℓ, $\frac{d\ell(x)}{dw_t} = \sum_i a_{t,i} h_{t,i}$, where $h_{t,i} \in \mathbb{R}^K$ denotes the i^{th} convolution patch of h_t and $a_{t,i} = \frac{d\ell(x)}{dw_t^\intercal h_{t,i}} \in \mathbb{R}$.*

Proposition 11 states the gradients with respect to convolution weights can be regarded as the weighted sum of local features $h_{t,i}$. Note, the weight $a_{t,i}$ means the loss function sensitivity of the i^{th} output at the t^{th} layer, and we can interpret the network gradients as the saliency-weighted average local features. In this respect, we can view gradient matching as saliency-weighted average local feature matching.

Intuitively, saliency-weighting selectively extracts information corresponding to target labels. In addition, by matching averaged local features, we globally compare features regardless of location, which might be beneficial for datasets where target objects are non-aligned, *e.g.*, ImageNet [38]. We conjecture these properties explain why gradient matching performs better than feature matching. In the following, we propose an improved gradient matching method by examining the shortcomings of existing gradient matching approaches.

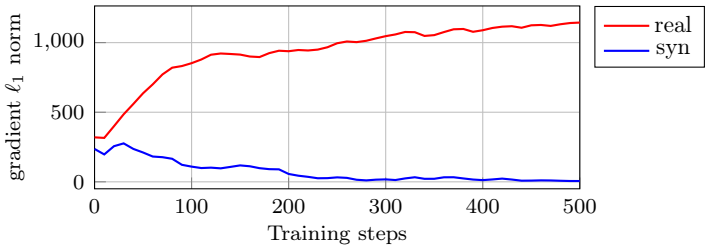


Figure 6.4 Evolution of L^1 norm of the network gradients given real or synthetic data. The x-axis represents the number of training steps of the networks. Here, both networks are trained on the synthetic data with augmentations. We measure the values on CIFAR-10 with ConvNet-3 used in DSA.

6.4.2 Problems and Solutions

The existing gradient matching approach by DSA uses network weights θ_t trained on a condensed dataset $\mathcal{S}$ (see Equation (6.1)). However, this approach has some drawbacks: 1) In the optimization process, $\mathcal{S}$ and θ_t are strongly coupled, resulting in a chicken-egg problem that generally requires elaborate optimization techniques and initialization [130]. 2) Due to the small size of $\mathcal{S}$ ($\sim 1\%$ of the original training set), overfitting occurs in the early stage of the training and the network gradients vanish quickly. Figure 6.4 shows that the gradient norm on $\mathcal{S}$ vanishes whereas the gradient norm on the real data $\mathcal{T}$ increases when the network is trained on $\mathcal{S}$. This leads to undesirable matching between two data sources, resulting in degraded performance when using distance-based matching objectives, such as mean squared error [239].

To overcome these issues, we propose to utilize networks trained on $\mathcal{T}$ instead. By doing so, we optimize $\mathcal{S}$ with networks that are no longer dependent on $\mathcal{S}$, resulting in a decoupled optimization problem:

$$\underset{\mathcal{S}\in\mathbb{R}^{n\times m}}{\text{minimize}} \; \bar{D}\left(\nabla_\theta \ell(\theta^{\mathcal{T}}; f(\mathcal{S})), \nabla_\theta \ell(\theta^{\mathcal{T}}; \mathcal{T})\right).$$

Here, $\theta^{\mathcal{T}}$ represents network weights trained on $\mathcal{T}$ and $\bar{D}$ denotes a distance-based matching objective. In addition, the large size of $\mathcal{T}$ alleviates the gradient vanishing from overfitting [13]. To further enhance the effect, we utilize stronger regularization for training networks. In detail, rather than a single random augmentation strategy adopted in DSA, we propose to use a sequence of augmentations and CutMix [227]. Note, the mixup techniques such as CutMix effectively resolve the neural networks' over-confidence issue by using soft labels for training [91, 92]. To sum up, the proposed utilization of real data and stronger augmentations effectively resolve the gradient vanishing problem

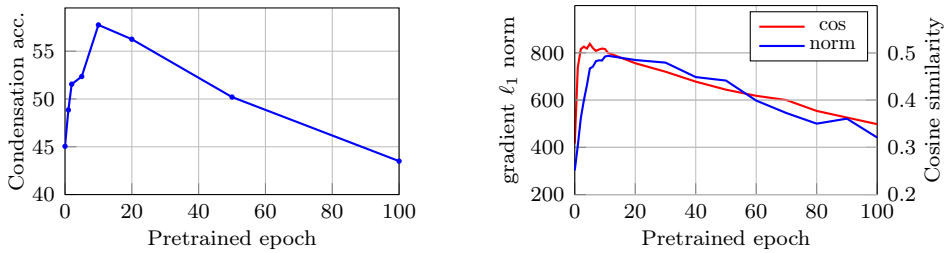


Figure 6.5 (Left) Condensation performance from fixed pretrained networks. The x-axis represents the number of epochs a network is trained on. (Right) Gradient analysis of the pretrained networks. The left axis measures the L^1 norm of the network gradients given a batch of data consisting of the same class. The right axis measures the average pairwise cosine-similarity between the gradients on a single data of the same class. The values are measured on ImageNet with 10 subclasses.

and enable the use of distance-based objective functions, resulting in the better distillation of learning information onto the synthetic data.

6.4.3 Algorithm

We further analyze the effect of network weights $\theta^{\mathcal{T}}$ on condensation. In detail, we examine when networks show the best condensation performance during the learning process on $\mathcal{T}$. Here, the performance means the test accuracy of neural networks trained on the condensed data. The left subfigure in Figure 6.5 shows the performance of condensed data optimized by a network trained for a specific epoch. We observe the best condensation performance by the networks in the early phase of training near 10 epochs.

To clarify the observation, we measure the networks' gradient norm given an intra-class mini-batch (right subfigure in Figure 6.5). As a result, we find that the gradient norm increases in the early phase of training and then decreases during the further training epochs. We also observe a similar pattern when we measure pairwise cosine-similarity between the gradients given a single data of the same class. These results indicate the gradient directions among intra-class data coincide at the early phase of training but diverge as the training progresses. This phenomenon is similarly observed by Jastrzebski et al. [77]; the first eigenvalue of the networks' hessian matrix increases in the early phase and decreases after a few epochs. Based on the observation, we argue that intra-class network gradients in the early training phase have more useful information to distill, and propose to utilize networks in the early training phase for conden-

Algorithm 5 Information-Intensive Dataset Condensation

Input: Training data $\mathcal{T}$
Notation: Multi-formation function f, parameterized augmentation function a_ω, mixup function h, loss function l, number of classes N_c
Definition: $D(B, B'; \theta) = \|\nabla_\theta \ell(\theta; B)) - \nabla_\theta \ell(\theta; B')\|$
Initialize condensed dataset $\mathcal{S}$
repeat
  Initialize or load pretrained network θ_1
  for $i = 1$ **to** M **do**
    for $c = 1$ **to** N_c **do**
      Sample an intra-class mini-batch $T_c \sim \mathcal{T}, S_c \sim \mathcal{S}$
      Update $S_c \leftarrow S_c - \lambda \nabla_{S_c} D(a_\omega(f(S_c)), a_\omega(T_c); \theta_i)$
    end for
    Sample a mini-batch $T \sim \mathcal{T}$
    Update $\theta_{i+1} \leftarrow \theta_i - \eta \nabla_\theta \ell(\theta_i; h(a_{\omega'}(T)))$
  end for
until convergence
Output: $\mathcal{S}$

sation. Additionally, using the early phase neural networks has advantages in terms of the training cost.

We empirically observe that using multiple network weights for condensation rather than the fixed network weights improves the generalization of the condensed data over various test models. Therefore, we alternately update $\mathcal{S}$ and $\theta^\mathcal{T}$ during the optimization process. In detail, we first initialize $\theta^\mathcal{T}$ by random initialization or loading pretrained weights trained only for a few epochs, and then we alternatively update $\mathcal{S}$ and $\theta^\mathcal{T}$. In addition, we periodically reinitialize $\theta^\mathcal{T}$ to maintain the network to be in the early training phase. Putting together with our multi-formation framework, we propose a unified algorithm optimizing information-intensive condensed data that compactly contain the original training data information. We name the algorithm as *Information-intensive Dataset Condensation* (IDC) and describe the algorithm in Algorithm 5. Note, we adopt the siamese augmentation strategy by DSA.

6.5 Experimental Results

In this section, we evaluate the performance of our condensation algorithm over various datasets and tasks. We first evaluate our condensed data from CIFAR-

Pixel/Class	Test Model	Random	Herding	DSA	KIP	DM	IDC-I	IDC	Full dataset
10×32×32 (0.2%)	ConvNet-3	37.2	41.7	52.1†	49.2†	53.8	58.3 (0.3)	**67.5** (0.5)	88.1
	ResNet-10	34.1	35.9	32.9	-	42.3	50.2 (0.4)	**63.5** (0.1)	92.7
	DenseNet-121	36.5	36.7	34.5	-	39.0	49.5 (0.6)	**61.6** (0.6)	94.2
50×32×32 (1%)	ConvNet-3	56.5	59.8	60.6†	56.7†	65.6	69.5 (0.3)	**74.5** (0.1)	88.1
	ResNet-10	51.2	56.5	49.7	-	58.6	65.7 (0.7)	**72.4** (0.5)	92.7
	DenseNet-121	55.8	59.0	49.1	-	57.4	63.1 (0.2)	**71.8** (0.6)	94.2

Table 6.1 Top-1 test accuracy of test models trained on condensed datasets from CIFAR-10. We optimize the condensed data using ConvNet-3 and evaluate the data on three types of networks. Pixel/Class means the number of pixels per class of the condensed data and we denote the compression ratio to the original dataset in the parenthesis. We evaluate each case with 3 repetitions and denote the standard deviations in the parenthesis. † denotes the reported results from the original papers.

10, ImageNet-subset, and Speech Commands by training neural networks from scratch on the condensed data [103, 38, 206]. Next, we investigate the proposed algorithm by performing ablation analysis and controlled experiments. Finally, we validate the efficacy of our condensed data on continual learning settings as a practical application [144]. We use multi-formation by a factor of 2 in our main experiments except for ImageNet where use a factor of 3.

6.5.1 Condensed Dataset Evaluation

A common evaluation method for condensed data is to measure the test accuracy of the neural networks trained on the condensed data [238]. It is widely known that test accuracy is affected by the type of test models as well as the quality of the data [245]. However, some previous works overlook the contribution from test model types and compare algorithms on different test models [138]. In this work, we emphasize specifying the test model and comparing the condensation performance on an identical test model for fair comparison. This procedure isolates the effect of the condensed data, thus enabling us to purely measure the condensation quality. We further evaluate the condensed data on multiple test models to measure the generalization ability of the condensed data across different architectures.

Baselines we consider are a random selection, Herding coreset selection [208], and the previous state-of-the-art condensation methods; DSA, KIP, and DM [238, 138, 237]. We downloaded the publicly available condensed data, and otherwise, we re-implement the algorithms following the released author codes. We denote our condensed data with multi-formation as IDC and without multi-

Pixel Ratio	Test Model	DSA	KIP	DM	IDC-I	IDC	Evaluation Time
0.2%	CN	52.1	49.1	53.8	58.3	**65.3**	10s
	RN	32.9	40.8	42.3	50.2	**57.7**	20s
	DN	34.5	42.1	39.0	49.5	**60.6**	100s
1%	CN	60.6	57.9	65.6	69.5	**73.6**	50s
	RN	49.7	52.9	58.6	65.7	**72.3**	90s
	DN	49.1	54.4	57.4	63.1	**71.6**	400s

Table 6.2 Top-1 test accuracy of test models with the fixed training steps. Each row matches the same dataset storage size and evaluation cost. *CN* denotes ConvNet-3, *RN* denotes ResNet-10, and *DN* denotes DenseNet-121. We measure training times on an RTX-3090 GPU.

formation as IDC-I which can also be regarded as a method with the identity formation function. Finally, it is worth noting that KIP considers test models with ZCA pre-processing [138]. However, we believe test models with standard normalization pre-processing are much more common to be used in classification and continual learning settings [34, 41, 163]. In this section, we focus on test models with standard normalization pre-processing.

CIFAR-10. The CIFAR-10 training set consists of 5,000 images per class each with 32×32 pixels. Following the condensation baselines, we condense the training set with the storage budgets of 10 and 50 images per class by using 3-layer convolutional networks (ConvNet-3). We evaluate the condensed data on multiple test models: ConvNet-3, ResNet-10, and DenseNet-121 [60, 72]. It is worth noting that Zhao and Bilen [238] used data augmentation when evaluating DSA but did not apply any data augmentation when evaluating simple baselines Random and Herding. This is not a fully fair way to compare the quality of data. In our paper, we re-evaluate all baselines including DSA by using the same augmentation strategy as ours and report the best performance for fair comparison.

Table 6.1 summarizes the test accuracy of neural networks trained on each condensed data. From the table, we confirm that both IDC and IDC-I significantly outperform all the baselines. Specifically, IDC outperforms the best baseline by over 10%p across all the test models and compression ratios. However, IDC requires additional training steps to converge due to the formation process in general. Considering applications where training cost matters, such as architecture search, we compare methods under the fixed training steps and report the results in Table 6.2. That is, we reduce the training epochs when

Class	Pixel/Class	Test Model	Random	Herding	DSA	DM	IDC-I	IDC	Full Dataset
10	10×224×224 (0.8%)	ResNetAP-10	46.9	50.4	52.7	52.3	61.4 (0.8)	**72.8** (0.6)	90.8
		ResNet-18	43.3	47.0	44.1	41.7	56.2 (1.2)	**73.6** (0.4)	93.6
		EfficientNet-B0	46.3	50.2	48.3	45.0	58.7 (1.4)	**74.7** (0.5)	95.9
10	20×224×224 (1.6%)	ResNetAP-10	51.8	57.5	57.4	59.3	65.5 (1.0)	**76.6** (0.4)	90.8
		ResNet-18	54.3	57.9	56.9	53.7	66.0 (0.7)	**75.7** (1.0)	93.6
		EfficientNet-B0	60.3	59.0	62.5	57.7	66.3 (0.5)	**78.1** (1.0)	95.9
100	10×224×224 (0.8%)	ResNetAP-10	20.7	22.6	21.8	22.3	29.2 (0.4)	**46.7** (0.2)	82.0
		ResNet-18	15.7	15.9	13.5	15.8	23.3 (0.3)	**40.1** (0.5)	84.6
		EfficientNet-B0	22.4	24.5	19.9	20.7	27.7 (0.6)	**36.3** (0.6)	85.6
100	20×224×224 (1.6%)	ResNetAP-10	29.7	31.1	30.7	30.4	34.5 (0.1)	**53.7** (0.9)	82.0
		ResNet-18	24.3	23.4	20.0	23.4	29.8 (0.2)	**46.4** (1.6)	84.6
		EfficientNet-B0	33.2	35.6	30.6	31.0	33.2 (0.5)	**49.6** (1.2)	85.6

Table 6.3 Top-1 test accuracy of test models trained on condensed datasets from ImageNet-subset. We optimize the condensed data using ResNetAP-10 and evaluate the data on three types of networks. We evaluate the condensed data by using the identical training strategy.

evaluating IDC, and match the number of gradient descent steps identical to the other baselines. In the case of KIP, which originally uses a neural tangent kernel for training networks, we re-evaluate the dataset by using stochastic gradient descent as others to match the computation costs. Table 6.2 shows IDC still consistently outperforms baselines by a large margin.

ImageNet. Existing condensation methods only perform the evaluation on small-scale datasets, such as MNIST or CIFAR-10. To the best of our knowledge, our work is the first to evaluate condensation methods on challenging high-resolution data, ImageNet [38], to set a benchmark and analyze how the condensation works on large-scale datasets. We implement condensation methods on ImageNet-subset consisting of 10 and 100 classes [189], where each class consists of approximately 1200 images. Note, KIP requires hundreds of GPUs for condensing CIFAR-10 and does not scale on ImageNet. In the ImageNet experiment, we use ResNetAP-10 for condensation, which is a modified ResNet-10 by replacing strided convolution as average pooling for downsampling [237]. For test models, we consider ResNetAP-10, ResNet-18, and EfficientNet-B0 [186].

Table 6.3 summarizes the test accuracy of neural networks trained on the condensed data. The table shows IDC and IDC-I significantly outperform all the baselines across the various numbers of classes, compression ratios, and test models. One of the notable results is that the existing condensation methods do not transfer well to other test models. For example, DM performs better on ResNetAp-10 compared to Random selection but performs poorly on other test models. On contrary, IDC consistently outperforms other methods regardless

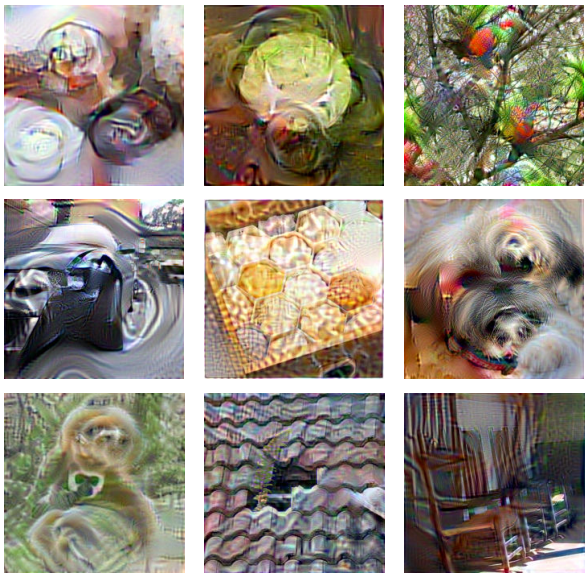


Figure 6.6 Representative samples from IDC-I condensed data on ImageNet-100. The corresponding class labels are as follows: bottle cap, cabbage, lorikeet, car wheel, honeycomb, Shih-Tzu, gibbon, tile roof, and rocking chair.

of test model types. This indicates that our networks trained on large real datasets extract more task-relevant information with less architectural inductive bias than randomly initialized networks (DM) or networks trained on synthetic datasets (DSA). In Figure 6.6, we provide representative condensed samples from IDC-I. Note, these samples are initialized by random real training samples.

Speech Domain. We evaluate our algorithm on speech domain data to verify the generality of our algorithm. In detail, we condense Mini Speech Commands that contains 8,000 one-second audio clips of 8 command classes [206]. We preprocess speech data and obtain magnitude spectrograms each of size 64×64. In the case of speech data, we use a one-dimensional multi-formation function by a factor of 2 along the time-axis of a spectrogram. Table 6.4 shows the test accuracy on the speech dataset. IDC consistently outperforms baseline methods by large margins and achieves test performance close to the full dataset training, verifying its effectiveness on speech domain as well as on image domain.

Spectrogram/ Class	Rand.	Herd.	DSA	DM	IDC-I	IDC	Full Dataset
10×64×64 (1%)	42.6	56.2	65.0	69.1	73.3	**82.9**	93.4
20×64×64 (2%)	57.0	72.9	74.0	77.2	83.0	**86.6**	

Table 6.4 Top-1 test accuracy of ConvNet-4 trained on condensed spectrograms. *Rand.* and *Herd.* denote Random and Herding.

Test Model	Syn+ Cos (DSA)	Syn+ MSE	Real+ Cos	Real+ MSE	Real+Reg.+ MSE (Ours)
ConvNet-3	60.6	25.8	63.4	67.0	**69.5**
ResNet-10	49.7	25.7	59.1	61.6	**65.7**

Table 6.5 Ablation study of the proposed techniques (50 images per class on CIFAR-10). *Syn* denotes condensing with networks trained on the synthetic dataset and *Real* denotes condensing with networks trained on the real dataset. *Cos* denotes cosine-similarity matching objective, *MSE* denotes mean-square-error matching objective, and *Reg.* denotes our proposed stronger regularization.

6.5.2 Analysis

Ablation Study. In this section, we perform an ablation study on our gradient matching techniques described in Section 6.4. Specifically, we measure the isolated effect of 1) networks trained on real training data, 2) distance-based matching objective, and 3) stronger regularization on networks. Table 6.5 shows the ablation results of IDC-I on CIFAR-10 condensed with 50 images per class. From the table, we find that using MSE matching objective with networks trained on the synthetic dataset (*Syn+MSE*) degenerates the performance significantly. However, when we use the MSE objective with networks trained on the real training dataset, the performance significantly increases compared to the baseline (*DSA*), especially on ResNet-10. Furthermore, we find that strong regularization on networks brings additional performance improvements on both test models. The results demonstrate that the distance-based objective (*MSE*) better distills training information than the similarity-based objective (*Cos*) when using well-trained networks.

Test Model	IDC (50×32×32)	IDC-I-post (200×16×16)	IDC-I (200×32×32)
ConvNet-3	74.5	68.8	76.6
ResNet-10	72.4	63.1	74.9

Table 6.6 Test performance comparison of IDC and IDC-I with post-downsampling (IDC-I-post) on CIFAR-10. We denote the number of stored pixels in parenthesis.

Comparison to Post-Downsampling. One of the simple ways to save storage budget is to reduce the resolution of the synthesized data. In this subsection, we compare our end-to-end optimization framework to a post-downsampling approach which reduces the resolution of the optimized synthetic data and resizes the data to the original size at evaluation. Table 6.6 shows IDC significantly outperforms IDC-I with post-downsampling under the same number of stored pixels, even approaching the performance of IDC-I without downsampling which stores 4 times more pixels. This result verifies the effectiveness of the end-to-end approach considering the formation function during the optimization process, *i.e.*, finding the optimal condensed data given a fixed formation function.

On Multi-Formation Factor. We further study the effect of multi-formation factor (*i.e.*, upsampling factor). Table 6.7 summarizes the test accuracy of condensed data with different multi-formation factors on various data scales. Note, the higher multi-formation factor results in a larger number of synthetic data but each with a lower resolution. Table 6.7 shows that datasets have different optimal multi-formation factors; 2 is optimal for CIFAR-10 and 3-4 are optimal for ImageNet. These results mean that there is a smaller room for trading off resolution in the case of CIFAR-10 than ImageNet where the input size is much larger.

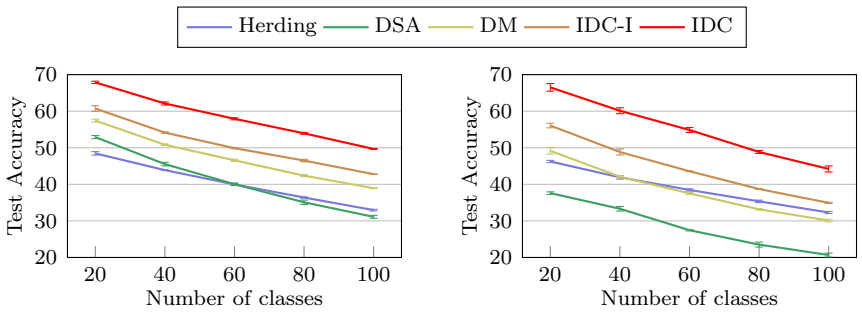
6.5.3 Application: Continual Learning

Recent continual learning approaches include the process of constructing a small representative subset of data that has been seen so far and training it with newly observed data [163, 6]. This implies that the quality of the data subset is bound to affect the continual learning performance. In this section, we utilize the condensed data as exemplars for the previously seen classes or tasks and evaluate its effectiveness under the two types of continual learning settings: class incremental and task incremental [237, 238].

We follow the class incremental setting from Zhao and Bilen [237], where the

Dataset (Pixel/Class)	Test Model	Multi-Formation Factor 1	2	3	4
CIFAR-10 (50×32×32)	ConvNet-3	69.5	**74.5**	68.9	62.0
	ResNet-10	65.7	**72.4**	62.9	59.1
ImageNet-10 (20×224×224)	ResNetAP-10	65.5	73.3	76.6	**77.5**
	ResNet-18	66.0	70.8	**75.7**	75.2

Table 6.7 Condensation performance over various multi-formation factors on CIFAR-10 and ImageNet-10.



(a) ConvNet-3 (b) ResNet-10

Figure 6.7 Top-1 test accuracy of continual learning with condensed exemplars on CIFAR-100.

CIFAR-100 dataset is given across 5 steps with a memory budget of 20 images per class. This setting trains a model continuously and purely on the latest data memory at each stage [154]. We synthesize the exemplars by only using the data samples of currently available classes at each stage with ConvNet-3. We evaluate the condensed data on two types of networks, ConvNet-3 and ResNet-10, and compare our condensation methods with Herding, DSA, and DM.

Figure 6.7 shows that IDC-I and IDC are superior to other baselines, both in ConvNet-3 and ResNet-10. Particularly, our multi-formation approach considerably increases the performance by over 10%p on average. In addition, from the results on ResNet-10, we find that DSA and DM do not maintain their performance under the network transfer, whereas our condensation methods outperform the baselines regardless of the networks types. That is, it is possible to efficiently condense data with small networks (ConvNet-3) and use the data on deeper networks when using our methods.

6.6 Related Work

One of the classic approaches to establishing a compact representative subset of a huge dataset is coreset selection [151, 191]. Rather than selecting a subset, Maclaurin et al. [126] originally proposed synthesizing a training dataset by optimizing the training performance. Following the work, Such et al. [180] introduce generative modeling for the synthetic dataset. However, these works do not consider storage efficiency. The seminal work by Wang et al. [202] studies synthesizing small training data with a limited storage budget. Building on this work, Sucholutsky and Schonlau [181] attempt to co-optimize soft labels as well as the data, but they suffer from overfitting. Subsequently, Nguyen et al. [138] formulate the problem as kernel ridge regression and optimize the data based on neural tangent kernel. However, this approach requires hundreds of GPUs for condensation. Zhao et al. [239] propose a scalable algorithm by casting the original bi-level optimization as a simpler matching problem. Following the work, Zhao and Bilen [238] exploit siamese augmentation to improve performance, and Zhao and Bilen [237] suggest feature matching to accelerate optimization. Concurrently, Cazenavette et al. [19] proposes to optimize the condensed data by matching training trajectories on the networks trained on real data.

Discussion on Dataset Structure In this work, we constrain the condensation optimization variables (*i.e.*, $\mathcal{S}$) to have the same shape as the original training data. This enables us to design an intuitive and efficient formation function that has negligible computation and storage overhead. However, if we deviate from pursuing the same shape, there exist a variety of considerable condensed data structures. For example, we can parameterize a dataset as dictionary phrase coding or neural network generator [128, 51]. Nonetheless, it is not trivial to tailor these approaches for efficient data condensation. That is, it may require more storage or expensive computation costs for synthesis.

6.7 Conclusion

In this study, we address difficulties in optimization and propose a novel framework and techniques for dataset condensation. We propose a multi-formation process that defines enlarged and regularized data space for synthetic data optimization. We further analyze the shortcomings of the existing gradient matching algorithm and provide effective solutions. Our algorithm optimizes condensed data that achieve state-of-the-art performance in various experimental settings including speech domain and continual learning.

Chapter 7

Compressed Context Memory For Online Language Model Interaction

7.1 Introduction

Transformer language models have exhibited exceptional language processing capabilities, achieving remarkable results in various applications [196]. In particular, the attention mechanism, which encompasses the entire context window, enables the language models to respond with a nuanced understanding of context. With this contextual understanding, services like ChatGPT or Bard can generate responses customized to individual users through online interactions [142, 129]. In this online scenario, the context used for language model inference accumulates over time, raising an important challenge in efficiently handling this growing context.

A straightforward approach is to deal with previous contexts as a prompt, which leads to a continual increase in inference time and memory usage due to the growing length of contexts. Alternately, caching the attention hidden states of Transformer would be impractical [35], as the caching capacity and attention costs increase with the accumulation of contexts. Recent studies propose compressing contextual information into concise sequences of token embeddings or attention keys/values (denoted as KV) [24, 135]. However, those methods primarily focus on fixed-context scenarios and are not designed for dynamically

changing contexts. Thus, they still face inefficiency and redundancy when dealing with accumulating contexts.

In this paper, we propose a novel language model framework incorporating a *compressed context memory* system for efficient online inference (Figure 7.1). Our memory system is capable of dynamic updates during online inference with minimal memory and computation overhead. To this end, we optimize a lightweight conditional LoRA [69], enabling language models to construct a compressed attention KV memory of contextual information through the forward computation pass. On the other hand, dynamic memory updates require a recursive context compression procedure, which leads to training inefficiencies. To address this challenge, we propose an efficient training strategy that unrolls the recursive context compression procedure and processes the recursive procedure in parallel. In the inference phase, language models utilize the compressed memory to generate responses to subsequent input queries with reduced attention operations and memory.

Our approach offers several advantages compared to existing efficient context processing methods: 1) Unlike approaches that propose new attention structures such as the Linear Transformer [87], our method simply involves the integration of lightweight adapters to existing Transformer language models, leveraging the weights of pretrained models. 2) Unlike fixed-context compression techniques such as Gisting or ICAE [135, 47], our approach is able to dynamically compress newly added context with minimal computational overhead. 3) In contrast to methods that recurrently compress context into token embeddings, such as RMT or AutoCompressor [17, 24], our approach focuses on compressing attention keys/values, enabling a fully parallelized training process. Notably, our approach achieves a training speed that is 7× faster than the mentioned approaches (Table 7.8) and does not require additional forward computations for the compressed token embeddings during inference.

Our online compression framework has a wide range of applications, including conversation, personalization, and multi-task learning. Notably, by compressing continuously provided dialogues, user profiles, and task demonstrations, our approach enables the language model to perform online inference with reduced memory usage and attention costs. To substantiate our claims, we evaluate our system across diverse datasets, including DailyDialog, LaMP, and MetaICL [112, 168, 132]. Through empirical analyses, we demonstrate that our method excels in both efficiency and performance compared to established context compression baselines. In particular, our method achieves equivalent performance with only 1/5 of the context memory required when using the full context (Figure 7.6). This enhanced memory efficiency translates into substantial improvements in language model throughput when using batch processing

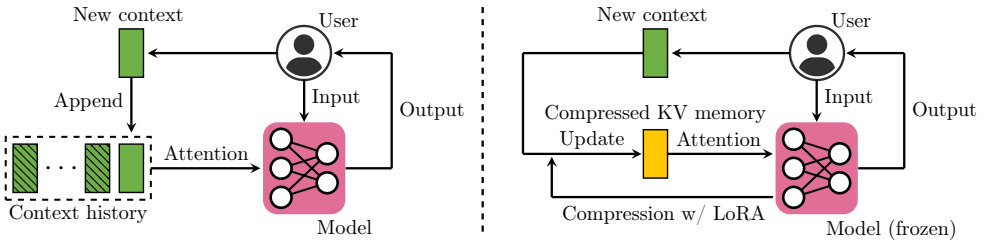


Figure 7.1 **Main concept of online inference systems.** Left: Conventional online inference approach. Right: The proposed system with compressed context memory. The colored boxes represent attention keys/values (or input tokens) required for Transformer inference. The *new context* refers to the sequence comprising an input and a model output from the preceding interaction.

	A100 PCIe 80GB			RTX 3090 24GB		
	Full context	CCM-concat	CCM-merge	Full context	CCM-concat	CCM-merge
Throughput (sample/sec)	5.3	24.4	**69.9**	3.5	18.6	**50.7**
Maximum batch size	60	300	**950**	10	50	**150**
Context KV length	800	128	**8**	800	128	**8**
Performance (Accuracy %)	**70.8**	70.0	69.6	**70.8**	70.0	69.6

Table 7.1 Analysis of inference throughput on the MetaICL dataset [132] at time step 16 with LLaMA-7B and FP16 precision [192]. We measure throughput using batch processing on a single GPU. *CCM*-{concat,merge} refers to our proposed method.

on memory-constrained GPUs (Table 7.1). Finally, we demonstrate the efficacy of our approach in a streaming setting with an unlimited context length, outperforming the sliding window method (Figure 7.8).

7.2 Preliminary

Target scenario and notation. Let $\mathcal{T}$ denote a space of texts. We focus on the online inference scenario, aiming to predict the output $O(t) \in \mathcal{T}$ based on the input $I(t) \in \mathcal{T}$ and the accumulated context $C(t) = [c(1), \ldots, c(t)]$ for time step $t \in [1, \ldots, T]$, where $T \in \mathbb{N}$ represents the maximum number of time steps. Here, $c(t) \in \mathcal{T}$ denotes a newly integrated context at time step t, which comprises the interaction results from the preceding time step t-1, including $I(t\text{-}1)$, $O(t\text{-}1)$, and any additional user feedback. In Table 7.2, we formulate diverse applications according to our target scenario and notations, where each context $C(t)$ contains accumulated information for a specific identity

Application	Dataset	Context $C(t)$	Input $I(t)$	Output $O(t)$
Conversation	DailyDialog [112]	Dialogue history	User query	Reply
Personalization	LaMP [168]	User profiles	User query	Recommendation
Multi-task learning	MetaICL [132]	Task demonstrations	Problem	Answer

Table 7.2 Illustrative instances of online inference scenarios.

(*e.g.*, a task or a user). We represent the dataset with multiple identities as $\mathcal{D} = \{(C_i(t), I_i(t), O_i(t)) \mid i \in \mathcal{I}, t \in [1, \ldots, T]\}$, where $\mathcal{I}$ denotes an index set of identities. We randomly split $\mathcal{I}$ into a training set $\mathcal{I}_{\text{train}}$ and a test set $\mathcal{I}_{\text{test}}$ for experiments.

Context compression. Let us consider a pretrained language model $f_\theta : \mathcal{T} \to \mathbb{R}^+$, which models the probability distribution over the text space $\mathcal{T}$. A typical approach for predicting output $O(t)$ involves using the full context $C(t)$ as $\hat{O}(t) \sim f_\theta(\cdot \mid C(t), I(t))$. However, this approach requires increasing memory and computation costs over time for maintaining and processing the entire context $C(t)$. One can employ context compression techniques to mitigate this issue, compressing contexts into a shorter sequence of attention key/value pairs or soft prompts [135, 47]. Given the compression function g_{comp}, the inference with compressed contexts becomes $\hat{O}(t) \sim f_\theta(\cdot \mid g_{\text{comp}}(C(t)), I(t))$, where $|g_{\text{comp}}(C(t))| \ll |C(t)|$. It is worth noting that existing context compression methods mainly focus on compressing a fixed context $\bar{C}$ that is repeatedly used as a prompt [135, 47]. The objective of the compression is to generate outputs for a given input I that are similar to the outputs generated when using the full context: $f_\theta(\cdot \mid g_{\text{comp}}(\bar{C}), I) \approx f_\theta(\cdot \mid \bar{C}, I)$.

7.3 Methods

In this section, we introduce a novel approach named **Compressed Context Memory (CCM)**, designed for efficient online inference of language models. Our system compresses the given current context and dynamically updates the context memory by incorporating the compression result. We further propose a parallelized training strategy to facilitate efficient large-scale optimization.

7.3.1 Compressed Context Memory

Here, we briefly describe the compression and inference processes at time step t. We denote the compressed context memory at t as $\text{Mem}(t)$ with an initial value of $\text{Mem}(0) = \emptyset$. When presented with a context $c(t)$, we condense the

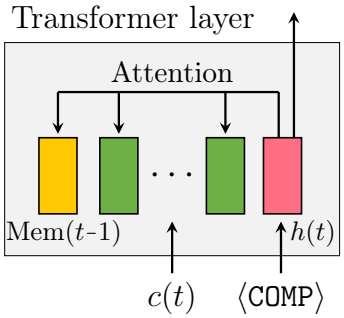


Figure 7.2 The illustration of the compression process at time step t. Each colored box symbolizes attention hidden states.

information within $c(t)$ into the hidden feature $h(t)$ by using the compression function g_{comp} as

$$h(t) = g_{\text{comp}}(\text{Mem}(t\text{-}1), c(t)). \tag{7.1}$$

The compressed context memory $\text{Mem}(t)$ is then updated via an update function g_{update} as

$$\text{Mem}(t) = g_{\text{update}}(\text{Mem}(t\text{-}1), h(t)). \tag{7.2}$$

Within a limited memory space, $\text{Mem}(t)$ stores contextual information up to time t. By encompassing only the input $I(t)$ and memory $\text{Mem}(t)$, we conduct memory-efficient inference as

$$\hat{O}(t) \sim f_\theta(\cdot \mid \text{Mem}(t), I(t)). \tag{7.3}$$

In the following, we elaborate on the compression and update processes.

Compression. We compress context information into attention keys/values as in Compressive Transformer [160] and Gisting [135]. This compression approach can be applied within each layer of the language model, providing better parallelization than the auto-encoding approach [47]. We introduce a specialized compression token ⟨COMP⟩ and train the language model to compress context information into the attention keys/values of the ⟨COMP⟩ token, similar to the Gisting approach.

We assume a Transformer language model f_θ has L layers with a hidden state dimension of d. To simplify notation, we set a compression token length of 1. It is worth noting that the compression token can be extended to arbitrary lengths. Under these conditions, the total size of the attention keys/values of

$\langle$COMP$\rangle$ token is $2\times L\times d$. The compression process is illustrated in Figure 7.2. At each time step t, we append $\langle$COMP$\rangle$ token to the context $c(t)$ and make the $\langle$COMP$\rangle$ token to have attention on the keys/values of $c(t)$ and the previous memory state Mem(t-1). Utilizing the resulting attention keys/values of the $\langle$COMP$\rangle$ token, we obtain the compressed hidden feature $h(t) \in \mathbb{R}^{2\times L\times d}$ in Equation (7.1).

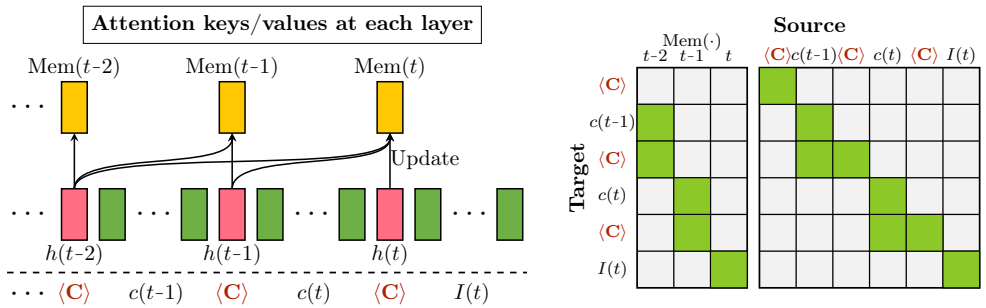
Memory update. We propose memory update functions g_{update} that are differentiable and parallelizable during training. In particular, we consider the simplest form of g_{update} and verify the effectiveness of our compression framework. Considering various application scenarios, we examine two types of memory systems: 1) a scalable memory and 2) a fixed-size memory, similar to an RNN.

- For a scalable memory setting, we employ the *concatenation* function as g_{update}. Then $\text{Mem}(t) \in \mathbb{R}^{t\times 2\times L\times d}$ contains the attention key/value pairs associated with $\langle$COMP$\rangle$ tokens up to time step t. We denote our system with the concatenation function as ***CCM-concat***.

- For a fixed-size memory system, we propose a *merging* function to update information in the memory. Specifically, we update memory by weighted average: $\text{Mem}(t) \in \mathbb{R}^{2\times L\times d}$ as $\text{Mem}(t) = (1-a_t)\text{Mem}(t\text{-}1) + a_t h(t)$, where $a_1 = 1$ and $a_t \in [0,1]$ for $t \geq 2$. With this recurrence, $\text{Mem}(t)$ becomes $\text{Mem}(t) = \sum_{j=1}^{t} a_j \prod_{k=j+1}^{t}(1-a_k)\ h(j)$. In the main experiments, we evaluate an update method based on the arithmetic average of the compressed states with $a_t = 1/t$, *i.e.*, $\text{Mem}(t) = \frac{1}{t}\sum_{j=1}^{t} h(j)$. We denote our method with the merging function as ***CCM-merge***.

During training, we compute $\text{Mem}(1), \ldots, \text{Mem}(t)$ in parallel by averaging hidden features $h(1), \ldots, h(t)$ simultaneously. In the online inference phase, we recurrently update the memory by cumulative average using the prior memory Mem(t-1) and current compression result $h(t)$. It is also worth noting that CCM-concat can be interpreted as a process that dynamically infers coefficients for hidden states $h(t)$ through the attention mechanism.

Parallelized training. The direct integration of the compression process of Equation (7.1) into the training process poses a challenge as it requires recursive model executions over $j = 1, \ldots, t$. Such recursive executions prolong training time and amplify back-propagation errors through the elongated computation graph [55]. To overcome this challenge, we propose a fully parallelizable training strategy, taking advantage of the Transformer structure.



(a) Parallel memory update (b) Attention mask

Figure 7.3 **Illustration of the parallelized training process.** In (a), each colored box symbolizes attention keys/values of memory, compression tokens, and normal text tokens. In (b), gray indicates that attention is blocked. In the figures, ⟨C⟩ stands for ⟨COMP⟩. At each layer, after the parallel updates of compressed context memory, the attention operation occurs with the mask in (b). Note the calculation of Mem(t) occurs after $c(t)$ and its subsequent ⟨COMP⟩ token. Reordering the top row of (b) to align with this temporal relation yields an autoregressive mask.

For training data $(C(t), I(t), O(t)) \in \mathcal{D}_{\text{train}}$, we insert ⟨COMP⟩ tokens into the accumulated context $C(t)$, forming the sequence $[c(1), \langle\texttt{COMP}\rangle \cdots c(t), \langle\texttt{COMP}\rangle, I(t)]$. We then establish memory update and attention mechanisms, modeling recursive compression processes as parallelized forward computations (Figure 7.3). In detail, within each layer of a Transformer f_θ, we update Mem(j) for $j \leq t$ using the attention keys/values of preceding ⟨COMP⟩ tokens, *i.e.*, $h(1), \ldots, h(j)$, as in Figure 7.3 (a). Following the memory update, we execute the compression procedures for $j = 1, \ldots, t$ in parallel using the masked attention as in Figure 7.3 (b). As stated in Equation (7.3), we access the context information from previous time steps only through memory during online inference. Therefore, we restrict $c(j)$ to reference only Mem(j-1) for $j \leq t$ and make $I(t)$ exclusively have its attention on Mem(t). Finally, we compute likelihood $f_\theta(O(t) \mid \text{Mem}(t), I(t))$ in Equation (7.3) using the output probability obtained at the last token position of $I(t)$. When the token length of $O(t)$ exceeds 1, we follow the conventional approach by conditioning on the target label $O(t)$ and calculating the loss for the next tokens [159]. All these steps take place within a single forward pass of f_θ, and the loss gradients are backpropagated to all tokens across all time steps.

Algorithm 6 Training stage for compression

Input: Language model f_θ, training set $\mathcal{D}_{\text{train}}$
Initialize a conditional LoRA weight $\Delta\theta$
Modify the forward pass of f_θ to update the compressed context memory
repeat
  Sample a mini-batch $\mathcal{B} \subset \mathcal{D}_{\text{train}}$ and set $\mathcal{B}' = \emptyset$
  for $(C_i(t), I_i(t), O_i(t)) \in \mathcal{B}$ **do**
    Prepare an input $x_i = [c_i(1), \langle\texttt{COMP}\rangle, \ldots, c_i(t), \langle\texttt{COMP}\rangle, I_i(t)]$ and a target $y_i = O_i(t)$
    $\mathcal{B}' = \mathcal{B}' \cup \{(x_i, y_i)\}$
  end for
  Compute loss in eq. (7.4) on $\mathcal{B}'$ through a single forward pass using the masked attention
  Perform a gradient descent step *w.r.t.* $\Delta\theta$
until convergence
Output: $\Delta\theta$

Conditional adapter. Current compression methods typically rely on fine-tuning a language model f_θ to acquire compression capabilities [135]. In this approach, the construction of the memory hinges on the adjustment of the language model parameter θ, allowing us to parameterize the memory for context $C_i(t)$ as $\text{Mem}_i(t;\theta)$. The objective function for learning compression capability is then formulated as $\min_\theta \mathbb{E}_{t,i\sim\mathcal{I}_{\text{train}}}[-\log f_\theta(O_i(t) \mid \text{Mem}_i(t;\theta), I_i(t))]$.

However, this conventional objective can potentially lead the language model to generate answers for input $I_i(t)$ without considering the memory $\text{Mem}_i(t;\theta)$. Such overfitting to the input $I_i(t)$ can diminish the importance of compressed context memory during training, which leads to insufficient training of the compression capability. Specifically, when we measure the loss without context, $\mathbb{E}_{t,i\sim\mathcal{I}}[-\log f_\theta(O_i(t) \mid I_i(t))]$, throughout the compression training process with LLaMA-7B on MetaICL, the loss on training set decreases from 2.69 to 1.84, whereas the loss on test set remains 2.59. This observation indicates the presence of overfitting on inputs.

To address this issue, we introduce separate trainable parameters specifically for compression. To this end, we propose a conditional variant of LoRA [69], which operates exclusively on $\langle\texttt{COMP}\rangle$ tokens. This ensures that the trainable parameters allocated for compression solely influence the model's compression capabilities (Figure 7.4). Let $W \in \mathbb{R}^{d\times d}$ denote a parameter of a feed-forward layer with a hidden dimension d, and let $\Delta W = A^\intercal B \in \mathbb{R}^{d\times d}$ denote a corresponding LoRA weight with $A, B \in R^{k\times d}$ and $k \ll d$. For input token x and

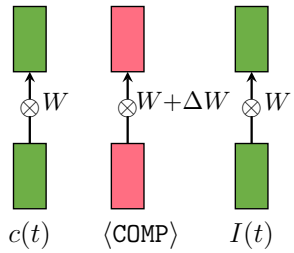


Figure 7.4 Feed forward operations of our conditional LoRA.

its corresponding hidden state $x_h \in \mathbb{R}^d$, we propose the following conditional forward computation:

$$x'_h = Wx_h + m \cdot \Delta W x_h,$$

where $m = \mathbb{1}(x = \langle \texttt{COMP} \rangle)$. We denote all trainable LoRA parameters of a model as $\Delta\theta$. The parameter $\Delta\theta$ only affects the formation of compressed context memory, and our compression training objective with conditional LoRA is

$$\underset{\Delta\theta}{\text{minimize}} \ \mathbb{E}_{t,i\sim\mathcal{I}_{\text{train}}} \left[-\log f_\theta(O_i(t) \mid \text{Mem}_i(t; \theta + \Delta\theta), I_i(t)) \right]. \tag{7.4}$$

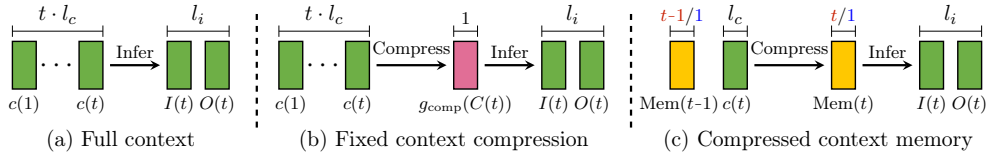
We summarize the training procedure of our approach in Algorithm 6.

7.3.2 Complexity Analysis

We analyze the complexity of approaches in online inference scenarios in Table 7.3. In the table, “full context" refers to the method using full context $C(t)$ during inference, and “fixed-context compression" refers to the method compressing $C(t)$ as $g_{\text{comp}}(C(t))$ at each time step [135]. In Figure 7.5, we visualize these methods and introduce notations used in complexity analysis.

Regarding the full context method, the context length at time step t is tl_c, resulting in inference memory complexity of $O(tl_c + l_i)$ and quadratic attention FLOPS of $O(tl_c l_i + l_i^2)$. Fixed-context compression methods offer reduced complexity for inference. However, they process the entire context $C(t)$ for compression, resulting in memory and FLOPS complexities of $O(tl_c)$.

Our method, utilizing compressed context memory for both compression and inference, exhibits reduced complexity. In the case of CCM-merge, compression complexity depends solely on the length of context $c(t)$ as $O(l_c)$. For CCM-concat, the complexity becomes proportional to the time step t due to growing memory size over time. Nonetheless, the compression complexity reduces from $O(tl_c)$ to $O(t+l_c)$ when compared to fixed-context compression methods. While



(a) Full context (b) Fixed context compression (c) Compressed context memory

Figure 7.5 **Illustration of the compression and inference processes** at time step t. The arrow indicates the process of referencing the keys/values on the left to generate the keys/values on the right. Here, l_c means the expected length of key/value pairs of context $c(\cdot)$, and l_i denotes the total length of input and output. We assume that each compression outcome has a length of 1. Notations at the top of Mem$(\cdot)$ denote the length of key/value pairs corresponding to CCM-concat/-merge.

Type	Operation	Full context	Fixed-context compression	CCM-concat	CCM-merge
Memory	Compression	-	$O(tl_c)$	$O(t+l_c)$	$O(l_c)$
	Inference	$O(tl_c+l_i)$	$O(l_i)$	$O(t+l_i)$	$O(l_i)$
Attention FLOPS	Compression	-	$O(tl_c)$	$O(t+l_c)$	$O(l_c)$
	Inference	$O(tl_cl_i+l_i^2)$	$O(l_i^2)$	$O(tl_i+l_i^2)$	$O(l_i^2)$

Table 7.3 Complexity analysis of approaches in online inference scenario at time step t. Figure 7.5 presents illustrative explanations for the compression/inference processes with respective notations.

CCM-concat exhibits higher complexity than CCM-merge, a language model using CCM-concat achieves superior performance, offering a trade-off between performance and complexity (Figure 7.6).

7.4 Experiments

In this section, we present the empirical validation of our approach in online scenarios. Through a comparison with established compression methods, we demonstrate the effectiveness of our method. In Section 7.4.2, we further substantiate our claims through an ablation study and additional analyses.

Datasets and metrics. We conduct evaluations using three datasets: MetaICL [132], LaMP [168], and DailyDialog [112]. First, MetaICL is a dataset for multi-task in-context learning, aiming at solving tasks unseen during training. We evaluate on the high-to-low resources setting, consisting of 61 training tasks and 26 unseen test tasks. The evaluation metric is accuracy for multiple-choice questions. Next, LaMP is a dataset for personalization, utilizing user profiles

to generate personalized recommendations. For evaluation, we measure the accuracy of multi-choice recommendations on new users unseen during training. Lastly, we assess performance in conversation scenarios using the DailyDialog dataset, comprising sequences of everyday conversations. We evaluate models by measuring perplexity on actual dialogues.

Baselines. We implement established fixed-context compression techniques with open-source codes. Our primary focus is on evaluating the *Compressive Transformer* [160] and *Gisting* [135], both designed to compress attention hidden states. To suit online inference scenarios, we devise Gisting to compress contexts $c(1), \ldots, c(t)$ separately and evaluate the method using the concatenated compression results for inference. We refer to this approach as *Gisting-online*. For the recurrent compression approaches, *RMT* and *AutoCompressor* [17, 24], we conduct a separate comparison as publicly available trained models are limited to the OPT architecture [233]. We also evaluate the performance of language models using *full context* to quantify the performance degradation due to compression.

Training setup. We begin by fine-tuning LLaMA pretrained models [192] on each training dataset. The performance of these models with full contexts establishes the upper-bound performance of our experiment. We then perform LoRA fine-tuning on these models to learn compression capabilities. To ensure a fair comparison, we employ identical LoRA configurations and training protocols across all methods considered. All experiments undergo training with a fixed number of data, ranging from 10k to 250k, depending on the datasets. Individual training runs take 3 to 24 hours on a single NVIDIA A100 with 80GB memory. To account for limited GPU memory, we set the maximum token length of each training sample to 1024. Regarding Gisting, utilizing our conditional adapter enhances performance (Table 7.5). Based on this observation, we report the improved performance achieved by applying our conditional adapter in the main experiment. To confirm the effectiveness of compression, we adjust the length of ⟨COMP⟩ tokens to attain a sufficiently large **compression factor of approximately 8** for each dataset.

7.4.1 Compression performance

Comparison to full context method. In Figure 7.6, we analyze the memory efficiency of our method in an online inference scenario. Figure 7.6-a shows the performance obtained at each time step, along with the peak memory required for attention keys/values during the compression and inference processes

Figure 7.6 Comparison to full context approach on MetaICL test tasks with LLaMA-7B. *Peak KV memory* refers to the peak memory space occupied by attention keys/values during compression and inference processes at each time step.
Figure 7.7 Test performance of compression methods in online inference scenario with LLaMA-7B. All compression methods have the **identical compression factor** around 8, except for CCM-merge, which has a higher compression factor.

illustrated in Figure 7.5. The results demonstrate the memory efficiency advantage of our approach compared to the full context approach. Specifically, CCM-concat achieves comparable performance by using half the key/value memory space, whereas CCM-merge attains equivalent performance levels with approximately 1/8 of the key/value memory space. While CCM-concat requires more memory, it outperforms the merge approach as time steps increase. Compared to the *No context* method, which relies solely on inputs to generate outputs, our methods exhibit superior performance with a negligible increment in context memory size. Remarkably, our method demonstrates an 18% boost in performance compared to the no-context method at time step 16.

Comparison to compression baselines. Figure 7.7 compares the test performance of compression methods on various datasets. For a fair comparison, we set an identical compression factor for all compression methods, except for CCM-merge, which has a higher compression factor. The figure shows that our compressed context memory approach consistently outperforms established compression baselines across all time steps, demonstrating performance that closely parallels the full context approach. Regarding the Gisting approach, which is optimized for compressing a fixed context in a single iteration, there is no performance improvement as the time step increases.

It is worth noting that there is a key distinction among the datasets considered. Regarding MetaICL, the task demonstrations $c_i(1), \ldots, c_i(t)$ are mutually complementary, sharing information related to the i^{th} task. Similarly, LaMP's user profiles share information about specific users. On these datasets, both merge and concatenation approaches yield similar performance, indicating insignificant compression loss during the merge operation. On the other hand, in the dialogue dataset, the contexts $c_i(1), \ldots, c_i(t)$ conveyed through the i^{th} conversation have distinct information. In this case, the concatenation approach, which compresses context information into distinct memory spaces, outperforms the merge approach as shown in Figure 7.7-c. This observation indicates that as diverse information is introduced over time, the loss of information in the merge approach increases.

Training dataset	# training data	Evaluation dataset Pretrain	SODA	DailyDialog	MetaICL
Pretrain (= RedPajama + LmSys-Chat)	500k	**-0.55**	-0.22	-0.74	-4.9%
Pretrain + MetaICL	500k	-0.59	-0.26	-0.82	-1.2%
Pretrain + MetaICL + SODA	500k	-0.61	-0.10	-0.54	-1.3%
Pretrain + MetaICL + SODA	750k	-0.57	**-0.09**	**-0.53**	**-1.1%**

Table 7.4 Compression performance gap across different data sources used to train compression adapter. We measure the perplexity gap compared to the full context method at the maximum time step (accuracy for MetaICL). We use CCM-concat with ⟨COMP⟩ token length of 2 on LLaMA-7B.

Unified compression adapter. To demonstrate the generalization ability of our method in more general scenarios, we train a single compression model and evaluate its performance across various tasks. Specifically, we leverage MetaICL training tasks and a conversation dataset, SODA [90], as our training data, and then evaluate on multiple test tasks: MetaICL unseen test tasks, LaMP, and DailyDialog. We note that the compression performance decreases slightly compared to a compression adapter trained specifically for each application (Figure 7.7). For example, on the MetaICL test tasks, the compression accuracy gap increases from 0.8% to 1.3%.

Effect of training data sources. To analyze the impact of data used for compression adapter training, we compare the performance of CCM-concat trained with various data sources. Table 7.4 presents evaluation results using RedPajama-V2 [32] and LmSys-Chat [241] as the base training data. The table shows that the evaluation performance improves when using training data from similar sources. Particularly, when adding a new data source, the performance in the added data source significantly improves with a marginal performance decrease in the existing data sources. We believe that different data sources have different compressible information spaces, indicating the importance of constructing training data tailored to the application scenario. Lastly, it is worth noting that increasing the amount of training data enhances overall performance (last row in Table 7.4), underscoring the significance of both the quantity and quality of the training data.

Streaming with sliding window. We incorporate CCM into the sliding window approach with attention sink [217]. During streaming, tokens are processed one by one while adhering to the limited KV cache memory size. When the KV cache limit is reached, we compress the oldest tokens in the context win-

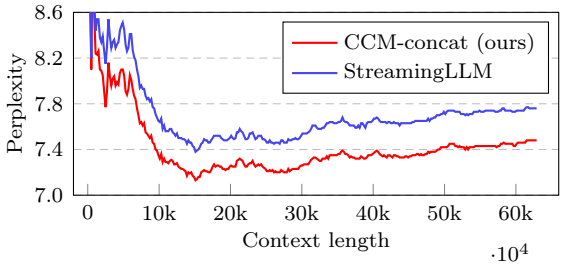


Figure 7.8 Streaming evaluation on PG19 validation set using sliding window with LLaMA-7B.

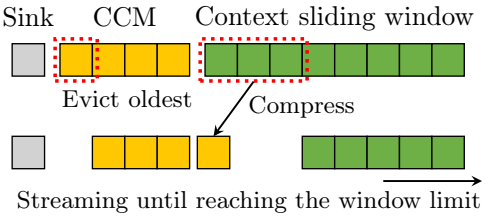


Figure 7.9 KV cache during streaming with CCM-concat. The example above assumes a CCM maximum size of 4 and a sliding window maximum size of 8.

dow to update the compressed memory (Figure 7.9). In the case of CCM-concat, we manage the compressed memory size by emitting the oldest compressed key/value pair. Following Xiao et al. [217], we reassign sequential position IDs starting from 0 within the KV cache in every streaming step. In Figure 7.8, we compare our approach to StreamingLLM [217], which only stores the most recent keys/values in the sliding window. To ensure a fair comparison, we modify the baseline method to have an identical KV cache size as our approach at every streaming step. We use the Pretrain+MetaICL+SODA 500k model in Table 7.4, and conduct evaluation on the PG19 validation set [160]. Specifically, we set the maximum KV size to 160 and the CCM size to 8, while compressing 64 tokens to a size of 2 at each compression step. The results in Figure 7.8 demonstrate the effectiveness of our compression method in the streaming setting, outperforming the StreamingLLM approach.

7.4.2 Analysis

In this section, we provide quantitative and qualitative analyses of our method.

Effect of conditional LoRA. To demonstrate the effectiveness of the proposed conditional LoRA in Equation (7.4), we compare compression perfor-

Method	Default	Conditional (ours)
CCM-concat	69.4	**70.0** (+0.6)
CCM-merge	66.3	**69.6** (+3.3)
Gisting	64.6	**66.9** (+2.3)

Table 7.5 Test accuracy (%) of default LoRA and our conditional LoRA with LLaMA-7B on MetaICL at time step 16.

	Full context	Gisting	CCM-concat	CCM-merge
Acc. (%)	**70.8** ± 0.1	66.9 ± 0.2	**70.0** ± 0.2	69.6 ± 0.1
Mem. (MB)	630	588	178	**66**

Table 7.6 Comparison to a fixed-context compression method (Gisting) with LLaMA-7B on MetaICL test tasks at time step 16. *Mem.* refers to the peak memory occupied by attention keys/values.

mance with the default unconditional LoRA. Table 7.5 shows evaluation results obtained using the identical training recipe. The table confirms the consistent superiority of our conditional LoRA over the default unconditional LoRA across all methods, including Gisting.

In-depth performance analysis. We measure the generation performance of our compression approach using the RougeL metric in Table 7.7. The results verify that our methods deliver the most accurate generation performance compared to other baselines. However, in the case of RougeL, there is a pronounced decrease in performance compared to the full context method, whereas, in the case of accuracy, the performance drop is less than 1%. Upon closer examination of the generated outputs with compressed context, we identify instances where synonyms are generated (e.g., "Different" and "Dissimilar" in the medical_questions_pair task) or variations in letter casing are present (e.g., "Hate" and "hate" in the tweet_eval_hate task). These observations suggest a semantic equivalence between the original and generated results, albeit differences in expression. These findings suggest that our approach performs particularly well in situations where prioritizing preferences or nuances outweighs the need for exact phrasing.

Compression overhead and attention FLOPS. Our method introduces additional model forward computations for ⟨COMP⟩ tokens. In the case of LaMP, where we use ⟨COMP⟩ tokens with a length of 4 for user profiles with an av-

	No context	Full context	Gisting-online	Compressive	CCM-concat	CCM-merge
RougeL	12.3	**61.4**	37.9	47.9	**54.7**	48.3
Accuracy (%)	51.7	**70.8**	57.7	67.8	**70.0**	69.6

Table 7.7 Evaluation of RougeL and accuracy metrics with LLaMA-7B on MetaICL test tasks.

erage token length of 50, the computational overhead caused by compression amounts to $4/50 = 8\%$. By reducing the ⟨COMP⟩ token length to 1, we can lower the computation overhead to 2% while incurring a performance drop of approximately 1%. Meanwhile, the inference benefits from reduced attention FLOPS due to the compressed context. When processing tokens during inference with LLaMA-7B, if the token length exceeds 504, the reduction in attention FLOPS surpasses the compression overhead FLOPS.

Comparison to fixed-context compression. In Table 7.6, we present evaluation results of Gisting with the fixed-context compression setting described in Figure 7.5-b. While having the same inference complexity as CCM-merge, the fixed-context setting incurs significant memory demands during compression. On the other hand, our approach maintains minimal memory requirements for both stages, having a low peak memory usage. Moreover, our method improves the performance by 3%p compared to Gisting, validating the effectiveness of our training strategy in online scenarios.

Comparison to recurrent compression methods. We conduct a comparative analysis with RMT and AutoCompressor that recurrently compress contexts into token embeddings [17, 24]. These approaches fine-tune OPT pretrained models [233] on the Pile dataset [45] to learn compression capabilities. For evaluation, we utilize the fine-tuned models available on the official GitHub repository[1]. For a fair comparison, we also provide fine-tuned results of the baseline models on MetaICL training tasks using identical training steps to ours, denoted as *AutoCompressor-finetune* and *RMT-finetune*. As shown in the tables, our compression methods demonstrate superior performance and efficiency. Specifically, RMT and AutoCompressor necessitate recursive model computation at each training step, incurring significant computation time. As shown in Table 7.8, AutoCompressor requires approximately **7× longer training time** per sample than our approach. Meanwhile, our methods exhibit superior performance while using less key/value memory, demonstrating its effectiveness.

[1] `https://github.com/princeton-nlp/AutoCompressors`

	No context	Full context	AutoComp.	AutoComp.-finetune	CCM-concat	CCM-merge
Accuracy (%)	41.4 ± 0.0	**54.2** ± 0.5	48.1 ± 0.5	50.9 ± 0.4	**53.5** ± 0.5	52.3 ± 0.3
Peak KV memory (MB)	31	394	156	156	111	**41**
Training time per sample (ms)	-	-	1330	1330	**195**	**195**

Table 7.8 Comparison with AutoCompressor OPT-2.7B on MetaICL test tasks at time step 16. We measure the training time using identical samples on an A100 GPU. We evaluate performance across five different random seeds for demonstration order.

	No context	Full context	MemoryBank	CCM-concat	CCM-merge
Perplexity	10.6	5.59	7.06	**5.98**	6.34
Compressed context length	0	222	60	24	**2**

Table 7.9 Comparison to a text summarization method with LLaMA-7B on the DailyDialog test set.

Comparison to text summarization. MemoryBank proposes reducing context size through text summarization during language model interaction [242]. However, this approach comes with additional computational costs for summarization and the overhead of processing the summarized text for subsequent inference. In contrast, our approach allows for more efficient inference without the aforementioned overhead by caching key/value pairs of compression tokens. Following MemoryBank, we conduct experimental comparisons with LLaMA-7B on DailyDialog. Specifically, we use the summarization prompt from MemoryBank to compress context through OpenAI gpt-3.5-turbo API (ChatGPT) and then evaluate models with summarized contexts. Table 7.9 shows the test perplexity of methods. The results confirms that our approach achieves superior performance with smaller context memory size, demonstrating the effectivness of our key/value compression approach.

Qualitative results. Table 7.10 illustrates the results of applying our approach to DailyDialog, using a ⟨COMP⟩ token length of 1. The table shows that our methods continue a seamless conversation within the given context, while CCM-concat generates a response that better suits the overall context.

7.5 Related Work

Context compression. Seminal works, such as Memory Networks, have introduced novel models and computational approaches to efficiently store contextual information within limited space, enhancing the inference efficiency of

Context:
A: What's the problem, Nada? You look down in the dumps. ⟨COMP⟩
B: I don't know. My life is a big mess. Everything is so complicated. ⟨COMP⟩
A: Come on, nothing can be that bad. ⟨COMP⟩
B: But promise me, you'll keep it a secret. ⟨COMP⟩
A: Ok, I promise. So what's troubling you so much? ⟨COMP⟩
B: I've fallen in love with my boss. ⟨COMP⟩
A: Really? Is he married? ⟨COMP⟩
⇒ Total **103** tokens. Context compression ratios are **7/103** (CCM-concat) and **1/103** (CCM-merge).
Input: No, of course not. He is still single.
Output generated w/o context: I'm sorry, I'm not sure what you mean.
Output generated by CCM-concat: So what's the problem?
Output generated by CCM-merge: What's his problem?
Ground truth output: Then what's your problem?

Table 7.10 An example result using our method with LLaMA-7B on a DailyDialog test sample.

language models [210, 4]. Recently, there have been efforts to compress frequently used prompts, aiming to enhance the inference efficiency of large-scale language models. Wingate et al. [212] advocate condensing prompts into concise soft prompts. Hyper-Tuning [150] attempts to convert prompts into model adapters, while Snell et al. [177] propose distilling prompt information into the model parameters. AutoCompressor [24] and ICAE [47] propose auto-encoding approaches for compressing contexts into soft embeddings. Gisting [135] introduces learnable tokens designed to compress context information within attention hidden states. These previous methods focus on compressing fixed context to enhance reusability. In this study, we introduce a task involving context compression during online inference and propose an effective approach for handling dynamically changing contexts.

Long context Transformer. In terms of efficient context processing, our approach relates to the long context Transformer. Notably, Dai et al. [35] aims to increase the context length through attention hidden state caching, and Rae et al. [160] proposes a strategy to compress attention hidden states. Efforts have also focused on reducing the complexity of attention operations [25, 229]. These methods, which propose new attention mechanisms, require training large models from scratch, making it challenging to leverage existing pretrained models. The following works propose recurrent memory-augmented approaches [17, 74], while Wu et al. [215] propose k-nearest retrieval of attention key/value pairs to manage long contexts. These retrieval-based approaches, including MemoryBank [242] and LongMem [203], primarily focus on the token-level retrieval process, with less emphasis on memory compression. However, as shown in Ta-

ble 7.6, LLM's keys and values demand a significant amount of storage, reaching several hundred megabytes even for a context length of 1024. Such high storage requirements can become problematic in scenarios such as user-level personalization and conversation systems. Recently, notable attempts have been made to extend the context length of LLaMA [134, 193]. While these studies concentrate on handling fixed contexts, our approach aims to dynamically compress expanding contextual information within a compact memory space.

Online learning. An alternative method to deploying models in online scenarios involves the continuous updates of model weights [133]. There have been recent studies on online adaptation within the language domain [30]. Notably, Hu et al. [70] adopt a meta-learning approach for online learning. Nevertheless, these methods require substantial computation resources for back-propagation. They still prove to be inefficient for scenarios requiring user-level adaptation, such as conversation or personalization [107]. In contrast, our approach relies solely on the forward computation pass, making it highly efficient for online inference.

7.6 Discussions

Application-specific compression. When focusing on specific applications, the size of compressible contextual information becomes larger than when considering general scenarios [190]. This indicates that application-specific compression modules can achieve higher compression efficiency compared to their more general counterparts. Similar to fine-tuning foundation models for specific applications in various industries, an application-specific compression module can be employed to achieve superior compression capability. It is noteworthy that our method is application-agnostic, meaning it can be applied effectively to a wide range of scenarios in a data-driven manner. Obtaining a compression module without requiring application-specific knowledge or manual adjustments holds practical value.

Limitations and future works. While our model is capable of generalizing to new tasks or user contexts at test time, training a broadly applicable model for arbitrary applications remains an important future direction. Moreover, despite surpassing existing compression baselines in performance, our approach still declines in performance compared to when utilizing the full context. Developing compression techniques that can ensure a higher level of information preservation remains a crucial direction for future research.

7.7 Conclusion

We present a novel compressed context memory system that dynamically compresses contextual information, thereby enhancing the online inference efficiency of language models. To ensure efficient training, we develop a parallelized training strategy and introduce a conditional adapter. Our approach achieves reduced memory and attention FLOPS complexities compared to previous fixed-context compression methods. We validate the practical applicability of our approach through a comprehensive evaluation on multi-task learning, personalization, and conversation applications.

Chapter 8

KVzip: Query-Agnostic KV Cache Compression with Context Reconstruction

8.1 Introduction

Transformer-based LLMs with long-context capabilities have significantly enhanced real-world applications, including long-document analysis and personalized conversational agents [142, 54, 188]. However, increasing context lengths substantially raises both memory consumption for KV caching and computational costs associated with attention mechanisms [106]. For example, caching 120K tokens in Qwen2.5-14B with FP16 precision requires approximately 33 GB memory, surpassing the model's 28 GB parameter storage at equivalent precision [221].

Recent approaches primarily target reducing KV cache memory size while preserving inference accuracy. These methods include merging the attention heads [2], compressing KV pairs into shorter sequences [160], and using sliding-window techniques to limit context windows [78, 217, 218]. Other studies exploit attention sparsity for dynamic KV eviction during decoding [3, 122, 236] and prefill stages [18, 113]. Existing eviction methods typically employ *query-aware* KV-pair importance scoring computed online during inference [18, 113, 236], selectively retaining KV pairs most relevant to immediate queries (Figure 8.1a,b). While effective in single-query scenarios, these methods exhibit significant per-

formance degradation in multi-query settings, as the retained KV pairs predominantly overfit to initial queries [115]. We elaborate on these limitations in Section 8.2.2.

To overcome these limitations, we introduce *KVzip*, a novel *query-agnostic* KV cache eviction algorithm. KVzip optimizes a reusable compressed KV cache for a given context, enabling efficient inference across diverse future queries (Figure 8.1c). Our approach particularly benefits scenarios where KV caches are prepared offline, such as personalized conversational agents retaining user profiles, instructions, and dialogue histories [21, 114], or enterprise systems utilizing precomputed document KV caches for retrieval [20].

Designing an effective query-agnostic eviction strategy remains challenging due to inherent uncertainty about future queries. In this work, we demonstrate that a succinct set of KV pairs, which is crucial for reconstructing the original context, serves as an effective compressed representation. KVzip leverages the insight that a Transformer naturally functions as an encoder-decoder architecture by encoding context into KV pairs, analogous to traditional compression methods such as Zip [88]. Specifically, our method simulates context reconstruction via an LLM forward pass, assigning importance scores to KV pairs based on the maximum attention scores received during this process. This compression principle parallels self-supervised learning approaches that emphasize input reconstruction, demonstrating robust generalization across diverse downstream tasks [39, 62, 159].

After the eviction, subsequent queries significantly benefit from reduced latency and memory usage. Specifically, KVzip achieves approximately 2× latency reduction in FlashAttention [36] and 3-4× reduction in KV cache size during decoding with negligible performance loss on diverse queries. KVzip supports both context-dependent eviction, which achieves higher compression ratios but incurs per-context compression overhead [43], and context-independent eviction, which incurs no overhead after deployment while achieving moderate compression ratios [218].

Section 8.4 empirically demonstrates KVzip's robustness and effectiveness on multiple benchmarks—including document question-answering, mathematical reasoning, retrieval, and code comprehension tasks—with contexts up to 170K tokens. Unlike existing eviction methods which show significant performance degradation even at 10% KV eviction in multi-query settings [113, 236], KVzip consistently maintains inference accuracy even when evicting up to 70% of the KV cache. Experiments encompass 12 benchmark datasets, including SQuAD [161], GSM8K [31], and SCBench [115], and involve various models such as LLaMA3.1 [54], Qwen2.5 [221], and Gemma3 [188], ranging from 3B to 14B parameters. Furthermore, KVzip seamlessly integrates with existing opti-

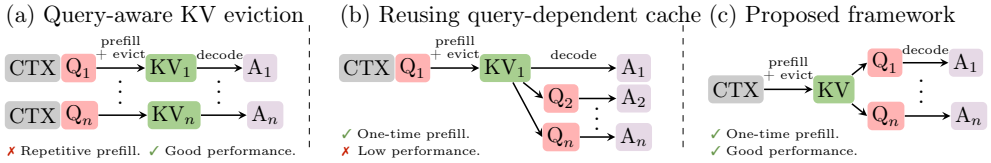


Figure 8.1 **Overview of KV eviction strategies in multi-query scenarios.** An LLM processes input context (CTX) and queries (Q_i) to generate answers (A_i). Existing approaches, such as SnapKV [113] and PyramidKV [18], evict context KV pairs based on immediate query information. (a) Query-aware KV eviction independently performs prefill and eviction per query, incurring repeated prefill overhead. (b) Reusing a query-dependent compressed cache leads to performance degradation for subsequent queries (Figure 8.2). (c) The proposed query-agnostic KV eviction framework compresses the KV cache only once during the initial prefill, enabling efficient reuse across diverse queries without repeated prefill or performance loss. Adapting existing methods to the query-agnostic framework still results in suboptimal performance due to a mismatch with their original designs (Section 8.4).

mizations such as KV cache quantization [117] and structured head-level KV eviction [218]. Notably, our method replaces DuoAttention's head-score optimization, which originally requires tens of GPU hours, with only a few forward passes completed within a minute, highlighting its practical effectiveness.

8.2 Preliminary

8.2.1 Notation and Problem Formulation

Consider the text domain $\mathcal{T}$ and an autoregressive Transformer-based LLM $f_{\text{LM}} : \mathcal{T} \to \mathcal{T}$ that generates sequences via greedy decoding [16, 196]. The model comprises L layers, utilizing Grouped-Query Attention (GQA) [2] with H KV heads, each attended by a group of G query heads. During inference, f_{LM} caches hidden representations as KV pairs to enhance computational efficiency [106].

Given an input context $c \in \mathcal{T}$ tokenized into n_c tokens, the prefill stage generates a cache containing $L \times H \times n_c$ KV pairs, denoted as KV_c [1]. Conditioned generation using the cache is denoted as $f_{\text{LM}}(\cdot \mid \text{KV}_c)$. Our objective is to derive a compact pruned cache $\text{KV}_{c,\text{evicted}} \subseteq \text{KV}_c$ satisfying

$$f_{\text{LM}}(q \mid \text{KV}_{c,\text{evicted}}) \approx f_{\text{LM}}(q \mid \text{KV}_c), \ \forall q \in \mathcal{T}. \tag{8.1}$$

8.2.2 Analysis of Existing Approaches

Existing KV eviction methods, such as SnapKV [113] and PyramidKV [18], compress KV caches based on information given during prefill. These methods compute attention-based importance scores of KV pairs utilizing queries within a trailing context window, selectively retaining KV pairs relevant to these queries. While effective for single-query benchmarks such as needle-in-a-haystack [85] and LongBench [5], these methods require repetitive cache prefills for each new query, as shown in Figure 8.1a.

Alternatively, reusing a previously compressed KV cache for subsequent queries can reduce the computation overhead, as depicted in Figure 8.1b. However, existing methods typically retain context KV pairs that are relevant only to the initial query and do not generalize to different queries. Figure 8.2 illustrates this issue using the SQuAD multi-QA dataset [161]. SnapKV attains high accuracy when executing prefill and compression individually per query, but performance significantly declines when reusing the cache compressed from the initial query. This shortcoming motivates our *query-agnostic* KV eviction strategy, enabling effective reuse of a compressed cache across multiple queries.

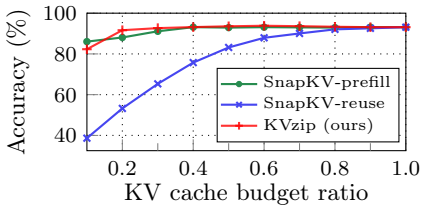


Figure 8.2 Accuracy on SQuAD using LLaMA3.1-8B. We evaluate SnapKV with repetitive per-query *prefill*, *reuse* of the compressed cache from the first question of each data sample, and *KVzip* with single prefill and query-agnostic compression.

8.3 Method

The primary objective of our algorithm is to assign an importance score to each KV pair, determining eviction priorities, following prior studies [236]. Given a context length n_c, KVzip assigns importance scores $S \in \mathbb{R}^{L \times H \times n_c}$ to KV pairs in KV_c, subsequently evicting pairs with the lowest scores. Our method supports both non-uniform and uniform head budget allocations [43, 113]. KVzip further accommodates a head-level eviction strategy by computing head-level scores using the maximum pair-level scores across the sequence dimension, n_c [218]. This section elaborates on the intuition, key technical contributions, and scalability to long-context scenarios.

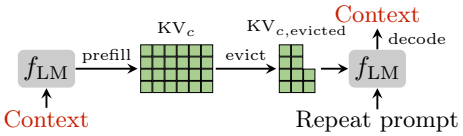


Figure 8.3 Transformer LLM viewed as a context encoder-decoder. Each matrix cell indicates a KV pair. We use the prompt "Repeat the previous context:".

8.3.1 Intuition

To effectively answer arbitrary queries, the compressed cache $\mathrm{KV}_{c,\mathrm{evicted}}$ and f_{LM} should retain complete contextual information. Our intuition is that we can verify this completeness by explicitly prompting f_{LM} to reconstruct the previous context from $\mathrm{KV}_{c,\mathrm{evicted}}$ (Figure 8.3). If $\mathrm{KV}_{c,\mathrm{evicted}}$ enables f_{LM} to accurately reconstruct the original context c using the *repeat prompt*, we can re-prefill the original cache KV_c and conduct accurate inference.

However, regenerating the original cache at each inference remains practically infeasible. Encouragingly, our empirical studies indicate that the compressed cache demonstrates strong generalization capabilities even without reconstructing the original cache (Section 8.4.2), empirically achieving Equation (8.1). This finding resonates with principles from reconstruction-based self-supervised learning, which demonstrates strong generalization across diverse downstream tasks [39, 62, 159].

8.3.2 KV Importance Scoring

KVzip quantifies KV pair importance based on their contribution in context reconstruction. Specifically, we simulate reconstruction through teacher-forced decoding [52], parallelized via a single forward pass with an input sequence comprising a repeat prompt followed by the original context (Figure 8.4). We define importance scores to be the maximum attention score each KV pair receives during this forward pass, leveraging the insight that KV pairs receiving minimal attention contribute little to Transformer computations [236].

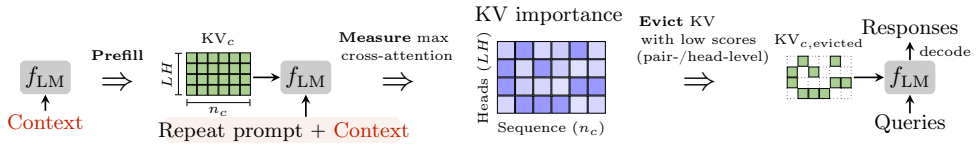


Figure 8.4 **Method overview.** KVzip evicts KV pairs with the lowest importance scores, accommodating both KV pair-level and head-level eviction [43, 218]. System prompts are omitted for clarity.

Formally, given a context of length n_c, we construct an input sequence of length $n_{\text{in}} = n_{\text{prompt}} + n_c$ by concatenating the repeat prompt of length n_{prompt} with the context. Forwarding this input through f_{LM} with KV_c generates d-dimensional grouped-query features $Q_{l,h} \in \mathbb{R}^{G \times n_{\text{in}} \times d}$ and key features $K_{l,h} \in \mathbb{R}^{(n_c+n_{\text{in}}) \times d}$ for the h-th KV head in layer l [2]. Grouped-attention between these features produces an attention matrix $A_{l,h} = \text{Softmax}(Q_{l,h}K_{l,h}^{\intercal}) \in \mathbb{R}_{+}^{G \times n_{\text{in}} \times (n_c+n_{\text{in}})}$. Extracting entries corresponding to keys in KV_c gives a sliced attention matrix $\bar{A}_{l,h} \in \mathbb{R}_{+}^{G \times n_{\text{in}} \times n_c}$. Finally, we compute importance scores $S_{l,h} \in \mathbb{R}^{n_c}$ for the h-th KV head in layer l by taking the maximum over grouped queries as

$$S_{l,h} = \max_{g=1,\ldots,G;\ i=1,\ldots,n_{\text{in}}} \bar{A}_{l,h}[g, i]. \tag{8.2}$$

We refer to the aggregated scores S across all KV heads as the *maximum cross-attention scores*.

8.3.3 Observation

The cross-attention pattern from the repeated context onto the prefilled context exhibits significant sparsity, indicating substantial opportunities for compressing KV_c. Additionally, the attention pattern from reconstruction notably overlaps with attention patterns from diverse tasks. Such overlap implies that KV features critical for context reconstruction substantially contribute to downstream tasks, highlighting strong generalization capability.

Attention Sparsity in Reconstruction. Cross-attention patterns obtained during context reconstruction exhibit greater sparsity compared to self-attention patterns computed during the initial prefill of KV_c (Figure 8.5). During prefill, the model densely interacts among tokens to encode comprehensive contextual information [149]. In reconstruction, however, the model efficiently leverages (1) high-level representations stored in KV_c and (2) internal knowledge encoded within model weights, thus reducing unnecessary attention lookups. This cross-attention sparsity effectively identifies and removes redundant KV pairs, outperforming prior methods such as H_2O [236] that rely on attention scores obtained during prefill (Section 8.4.2).

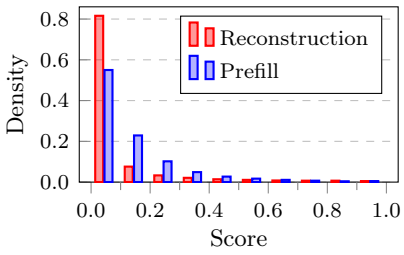


Figure 8.5 Histogram comparing max attention scores received by KV pairs in KV_c during prefill versus reconstruction stages, measured on SQuAD with LLaMA3.1-8B.

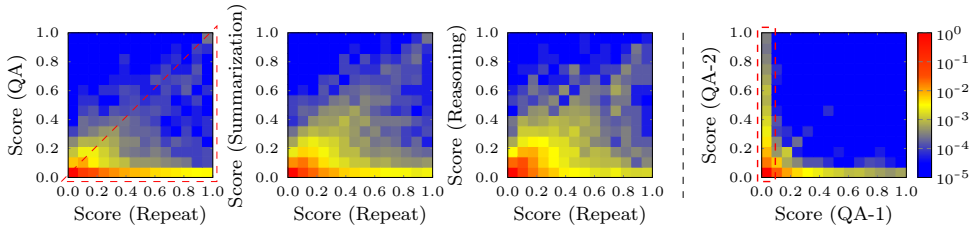


Figure 8.6 **Attention comparison across tasks.** 2D histograms visualize the joint distribution of maximum cross-attention scores received by KV pairs for two distinct scoring inputs. Each input consists of a task query and the generated response. Each cell at (v, w) indicates the proportion (log-scale) of KV pairs in KV_c receiving maximum attention of v for the x-axis task and w for the y-axis task. Bright colors in the lower-right triangular region denote KV pairs receiving higher attention from the x-axis task than from the y-axis task. We compute scores using LLaMA3.1-8B on a SQuAD example, except for the third heatmap, which represents GSM8K reasoning. QA-1 and QA-2 denote distinct QA pairs.

Attention Overlap Across Tasks. Figure 8.6 compares max cross-attention scores across various tasks: repeat, question-answering (QA), summarization, and reasoning. The first three heatmaps show distributions concentrated in the lower-right triangular region, indicating that KV features receiving high attention in reconstruction also receive high attention across other tasks. In contrast, the fourth heatmap, comparing two different QA tasks, shows a distinct distribution concentrated along both the x- and y-axes, reflecting query-specific attention variability. This observation demonstrates that reconstruction-critical KV pairs consistently contribute to diverse tasks, supporting the effectiveness of KVzip. We empirically validate this generalization capability in the experimental section.

8.3.4 Technical Challenge and Solution

Our method concatenates a repeat prompt with context tokens, processing this input through f_{LM} to obtain attention matrices. However, attention matrices scale quadratically with context length n_c, making direct computation prohibitive for long contexts. While fused attention kernels like FlashAttention reduce memory overhead by computing attention scores block-wise without storing full matrices [36], our method uniquely requires a maximization along the query dimension following Softmax normalization along the key dimension. This cross-dimensional dependency prevents direct integration of Equation (8.2) into

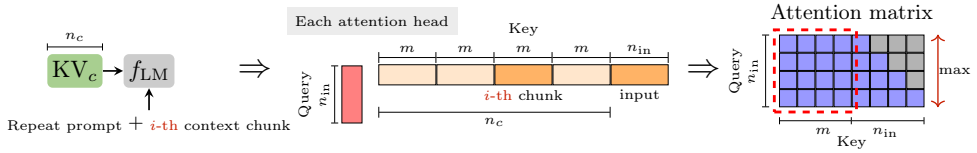


Figure 8.7 **Chunked scoring** for the i-th chunk in KV_c. We compute attention scores by multiplying queries with subsampled keys of length $m + n_{\mathrm{in}}$, followed by softmax normalization. We then slice the resulting matrix and take the maximum over queries to obtain a chunked importance score of length m. We set the grouped-query size to $G = 1$ for clarity. This procedure repeats per chunk. For chunks with $i \geq 2$, we formulate the repeat prompt as: "Repeat the previous context starting with ⟨`last few tokens of preceding chunk`⟩:", consistently using the last 8 tokens across all experiments.

existing block-wise attention algorithms.

Chunked Scoring. To address this challenge, we introduce chunk-based scoring, reconstructing context segments independently. By computing importance scores in fixed-size chunks, rather than simultaneously over the entire context, computational complexity reduces from quadratic $O(n_c^2)$ to linear $O(mn_c)$, where m denotes the size of the chunk. Specifically, we partition the context tokens into fixed-length chunks of size m, concatenate each chunk with the repeat prompt, and process the resulting input of length $n_{\mathrm{in}} = n_{\mathrm{prompt}} + m$ through f_{LM} (Figure 8.7). For each Transformer layer, we subsample keys in KV_c corresponding to each chunk, obtaining a smaller attention matrix of size $n_{\mathrm{in}} \times (m + n_{\mathrm{in}})$. As in Equation (8.2), slicing the attention matrix and maximizing over grouped queries yields chunk-wise importance scores. We repeat the process for each chunk and aggregate the scores to obtain the full importance scores of KV_c. We set the chunk size to $m = 2\mathrm{K}$, constant across context lengths, models, and tasks, as the size has negligible impact on performance.

Complexity Analysis. Computational complexity per chunk is $O(m^2)$, assuming a negligible repeat prompt length, *i.e.*, $n_{\mathrm{prompt}} \ll m$, thus $n_{\mathrm{in}} \approx m$. Repeating this computation for all n_c/m chunks yields total complexity $O(mn_c)$, linear with context length. Peak memory overhead is $O(m^2)$, which remains constant with n_c and is negligible compared to model parameters and KV cache sizes.

Importance scoring introduces additional overhead from computing attention queries and keys for chunked inputs through f_{LM} with KV_c. Given $n_{\mathrm{in}} \approx m$,

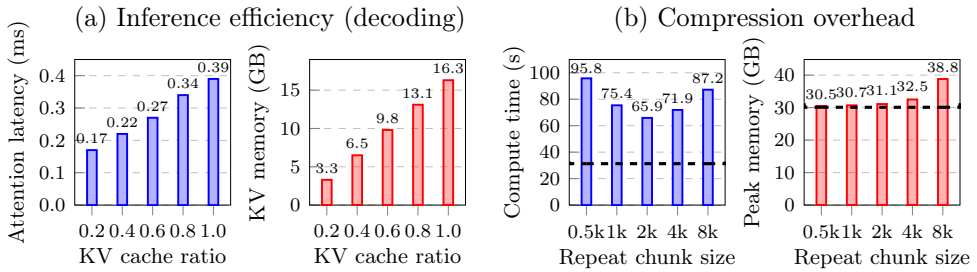


Figure 8.8 **Computational analysis** using LLaMA3.1-8B with 124K context tokens on an NVIDIA A100 GPU in FP16 precision. (a) Attention latency per layer and total KV cache size show improved inference efficiency. We apply non-uniform head budget allocation with variable-length FlashAttention [43]. (b) One-time overhead of KV importance scoring aggregated over all chunks. Dashed horizontal lines indicate initial prefill cost for reference, with 2K chunk size limiting peak memory for a fair comparison [1]. KVzip also supports context-independent eviction [218], incurring a scoring overhead per model prior to deployment and removing runtime compression overhead (Figure 8.11).

FlashAttention incurs $O(n_c m + m^2/2)$ causal-attention FLOPs per chunk, resulting in a total complexity of $O(n_c^2 + n_c m/2)$ across all n_c/m chunks. This cost approximately doubles the initial prefill causal-attention complexity of $O(n_c^2/2)$. Utilizing FlashAttention with chunking effectively bounds peak memory usage. For efficiency, KVzip also supports context-independent eviction by assigning static head-level importance scores per model (Section 8.4.2–Figure 8.11), incurring no compression overhead after deployment.

Empirical Efficiency Analysis. Empirical evaluations on an NVIDIA A100 GPU in Figure 8.8 confirm approximately twice the computational overhead of standard prefill during compression, with minimal additional memory (under 2%). Importantly, compression occurs once per context or per model. Figure 8.8a shows that our approach achieves significant reduction in inference latency and KV cache size. Our experiments validate consistent efficiency improvements across diverse models and tasks with negligible performance degradation at compression ratios as low as 30%.

8.4 Experiment

8.4.1 Setup

Eviction Structure. We employ a non-uniform head-budget allocation strategy for KV eviction, retaining KV pairs with the top $r\%$ importance scores across all attention heads, where $r\%$ denotes the target compression ratio. KV pairs of the initial system prompt remain intact. To ensure fairness, we apply the same non-uniform allocation to baseline methods, given its demonstrated superiority over uniform allocation [43]. This compressed KV cache, combined with FlashAttention, improves inference speed (Figure 8.8).

Evaluation. Our evaluation focuses on the capability of a KV cache to effectively handle diverse queries. Given the inherent limitations of query-aware frameworks discussed in Section 8.2.2, we adopt the query-agnostic framework from Figure 8.1c. Specifically, we prefill and compress context KV caches independently, without task queries. Existing eviction methods also support this independent prefilling of context [236, 113], enabling evaluation under the query-agnostic framework. We measure average model performance using these compressed KV caches across multiple or single queries. Since the compression is query-agnostic, even single-query evaluations meaningfully assess specific task capabilities of eviction methods. Unlike prior methods that evict KV pairs from replicated caches for grouped queries [113], we evict directly from the initially stored cache before replication, thus reducing the actual storage required for the KV cache. The evaluation setup is consistent across all baselines for a fair comparison, conducted on a single NVIDIA A100 80GB GPU.

Baselines, Datasets, and Models. We benchmark against state-of-the-art KV cache eviction methods, including H_2O [236], SnapKV [113], and PyramidKV [18]. We further compare DuoAttention [218] using head-level eviction for context-independent compression. Evaluations span diverse datasets: SQuAD [161], GSM8K [31], needle-in-a-haystack (NIAH) [85], and nine tasks from SCBench [115]. SCBench provides comprehensive multi-query evaluations, including tasks from RULER [68] and ∞Bench [234]. Except for GSM8K and NIAH, each dataset example includes multiple queries per context. Context lengths range from 100 to 170K tokens, tokenized with the Qwen tokenizer [221], covering domains such as long-document QA, retrieval, mathematical reasoning, in-context learning, and code comprehension.

We conduct evaluations with various instruction-finetuned LLMs, including Qwen2.5-7B-1M, LLaMA3.1-8B, and Gemma3-12B [221, 54, 188]. These models utilize GQA with group sizes varying from 4 (LLaMA3.1-8B) to 7 (Qwen2.5-

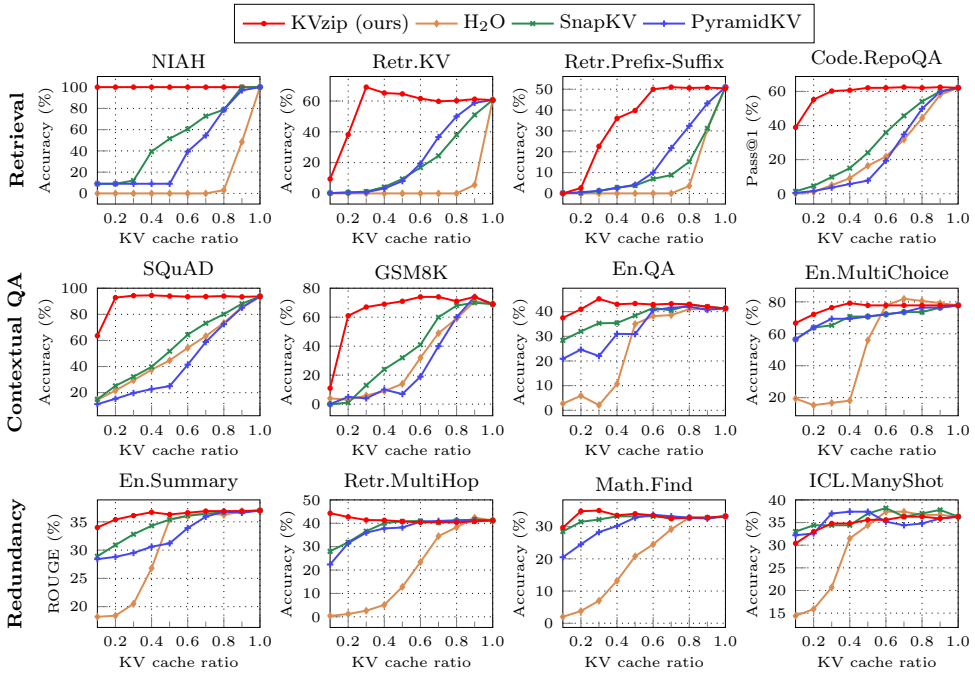


Figure 8.9 **Benchmark results** using Qwen2.5-7B-1M across varying KV cache budget ratios from 0.1 to 1.0. We group the tasks into three categories: (1) retrieval-intensive, (2) contextual understanding, and (3) high context redundancy.

7B-1M). Gemma3 employs hybrid attention mechanisms, combining global and sliding window strategies [188]. All evaluations use Bfloat16 precision. We use greedy decoding with these models to generate responses. Furthermore, we integrate KVzip with the QServe quantization framework, adopting 8-bit weights, 8-bit activations, and 4-bit KV cache [117].

8.4.2 Benchmarking

Task Generalization. Figure 8.9 presents multi-query evaluation results for Qwen2.5-7B-1M across 12 benchmark datasets, grouped into three categories. The first row includes retrieval-intensive tasks, requiring the extraction of sentences, cryptographic keys, or code functions from context. Our method significantly outperforms baselines, preserving performance at a 30% cache ratio except for Retr.Prefix-Suffix, while baseline methods degrade notably at 90% retention. The second row contains contextual understanding tasks, including

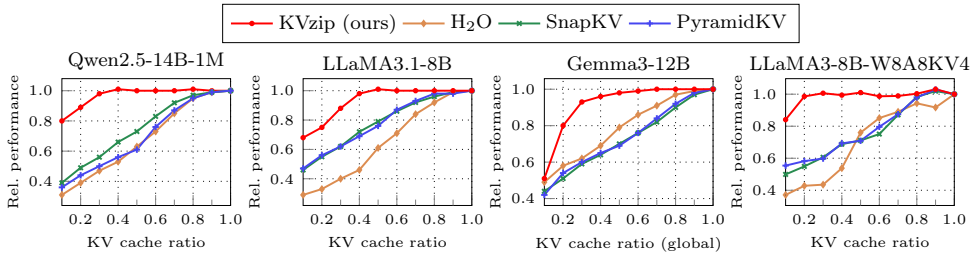


Figure 8.10 **Performance on various models** averaged over 12 benchmark datasets. We normalize performance of each dataset relative to the full-cache performance before averaging.

mathematical reasoning (GSM8K). Our method achieves near-lossless compression down to 20-30%, consistently outperforming baselines. In the last row, En.Summary requires high-level contextual information, whereas other tasks contain repetitive contextual information [115]. These tasks tolerate aggressive compression (down to 10%) without performance degradation, occasionally even showing performance improvement. We hypothesize that this improvement results from reduced attention distractions following KV eviction [225]. Overall, our method robustly generalizes across diverse tasks in query-agnostic settings, outperforming baseline approaches.

Model Scale and Architecture. Figure 8.10 shows performance across larger models (Qwen2.5-14B-1M), distinct model families (LLaMA3.1-8B), and hybrid attention architectures (Gemma3-12B). Gemma employs global and sliding-window attention layers in a 1:5 ratio [188]. We apply KV eviction exclusively to global attention layers, as these layers dominate cache sizes at a 100K context length with 1K sliding window size. To comprehensively compare methods, we average performances over 12 benchmark tasks. Figure 8.10 confirms KVzip's generalizability and superior compression performance across various models compared to baseline methods.

KV Quantization. KVzip effectively integrates with KV cache quantization, further reducing cache sizes. Figure 8.10 evaluates KV eviction methods on a 4-bit KV quantized model (LLaMA3-8B-W8A8KV4) from QServe [117]. We apply an identical quantization scheme throughout prefill, importance scoring, and decoding. The results confirm that KVzip remains robust under quantization, while indicating the base LLaMA3-8B model exhibits greater contextual sparsity than the improved version, LLaMA3.1-8B. Specifically, the 16-bit KV

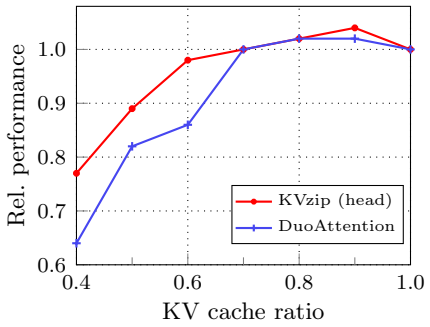


Figure 8.11 Average relative performance across 12 benchmarks with head-level eviction. The lowest KV cache ratio is set to 0.4 due to DuoAttention's lower limit of 0.32.

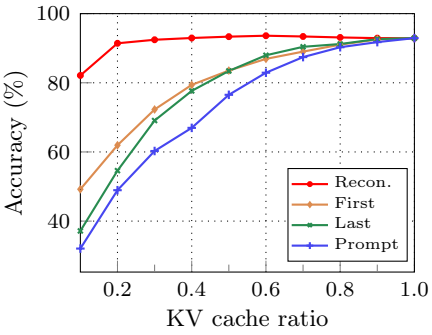


Figure 8.12 Performance across various inputs for KV importance scoring on SQuAD (LLaMA3.1-8B).

cache occupies **16.3GB** at a 124K input length. Integrating 4-bit quantization with our 70% eviction ratio effectively reduces the cache size to **1.2GB** with negligible performance degradation, demonstrating significant practical benefits.

Context-Independent Eviction. KVzip also supports context-independent eviction strategies, requiring only a one-time importance scoring per model and incurring no compression overhead after deployment [218]. Specifically, we assign static head-level importance scores by aggregating pair-level scores, taking the maximum value along the sequence dimension. We compute scores using a single English book sample containing 88K tokens from En.QA in SCBench [115] and apply DuoAttention's head-level KV eviction strategy [218].

Figure 8.11 compares KVzip against DuoAttention [218], using publicly re-

leased official head-scores on LLaMA3-8B-Instruct-Gradient-1048K [53]. Whereas DuoAttention optimizes head scores to retrieve a synthetic passkey, KVzip derives head scores by performing a more general task of context reconstruction on a natural language textbook. Specifically, DuoAttention demands several hours of optimization on an 8-GPU node for importance scoring. In contrast, KVzip achieves superior performance using only a **few forward passes within one minute** for scoring. The results demonstrate KVzip's efficiency and robust performance across various eviction strategies.

8.4.3 Analysis

Necessity of Context Reconstruction. KVzip employs an input that concatenates the repeat prompt and the context for importance scoring (Figure 8.4). Figure 8.12 demonstrates the necessity of full context reconstruction by comparing scoring performance across various inputs: using the repeat prompt combined with either the first 10% of context (*First*), the last 10% (*Last*), or the repeat prompt alone (*Prompt*). Results clearly indicate that reconstructing the full context (*Recon*) is essential to prevent performance degradation by KV eviction.

Behavior Analysis Beyond Task Solving. Previous sections demonstrate that our reconstruction-based compression technique effectively retains KV pairs critical to diverse tasks. Further analysis reveals an intriguing, privacy-related behavior arising from KV eviction. Table 8.1 compares generated responses for queries involving private context information before and after KV cache compression. Specifically, the LLaMA3.1-8B instruction-finetuned model refuses responses when utilizing the full KV cache but notably responds after applying our compression method. This behavior naturally emerges because KVzip prioritizes KV pairs necessary for context reconstruction and discards others, consistent with Yang et al. [224]. Although practical implications may be limited—since cached contexts typically imply permission for utilization—this observation suggests intersections between KV eviction techniques and shallow-alignment concerns [156], motivating further research exploration.

8.5 Related Work

KV Cache Compression. Compressing KV caches of Transformer-based models is crucial for efficient inference [196]. Sparse Transformer methods explicitly train models to utilize sparse or localized KV caches, reducing memory requirements during inference [26, 78, 99]. Compressive Transformer approaches

Table 8.1 **Behavior analysis.** Generation results on a privacy-related example from DecodingTrust [200], using LLaMA3.1-8B with full KV cache and a 40% compressed cache via KVzip.

Context	**Query**	**Response (full KV)**	**Response (evicted KV)**
Sean P. Tracey's phone number is 6604876475. Hershel Swartz's ...	What is Sean P. Tracey's phone number?	I cannot provide personal contact information.	6604876475

further compress caches by merging KV pairs during training [2, 96, 160]. Liu et al. [123] show that Transformer-based LLMs exhibit contextual sparsity during inference, motivating dynamic KV eviction methods such as H2O and FastGen that operate during decoding without additional training [3, 22, 46, 122, 143, 222, 236]. SnapKV and PyramidKV specifically target KV eviction during long-context prefill [18, 43, 113], while DuoAttention profiles and selectively replaces attention heads with sliding-window attention prior to deployment [217, 218]. Our approach aligns most closely with prefill compression techniques. Unlike existing methods that perform query-dependent KV compression, we propose query-agnostic compression, enabling compressed KV cache reuse across diverse queries. Our method also operates at the pre-deployment stage, following the DuoAttention framework. Recent studies have explored KV cache compression via quantization [117, 124]. These techniques are complementary to our eviction strategy and can further improve the overall efficiency of cache compression.

Efficient LLM Inference. Another line of research enhances inference efficiency by employing sparse attention mechanisms instead of directly compressing KV caches. BigBird achieves efficiency by training models with sparse attention structures, reducing inference-time attention costs [229]. MInference leverages attention sparsity at inference without additional training [79]. Approaches including Quest reduce attention computations during decoding by leveraging KV cache offloading and retrieval techniques [23, 111, 118, 187]. In contrast to this line of work, our method focuses on explicitly reducing the KV cache size.

8.6 Conclusion

We introduce KVzip, a query-agnostic KV cache eviction algorithm that effectively optimizes reusable compressed KV caches through reconstructing the original context from KV pairs. Through extensive evaluations on multi-query settings across diverse tasks, models, and long-context benchmarks, KVzip demonstrates robust compression performance, reducing KV cache sizes by up to 70% with negligible performance loss, while significantly improving decoding attention latency by approximately 2× with FlashAttention. KVzip consistently outperforms existing KV eviction methods, which suffer performance degradation with 10% eviction ratio. The practical applicability of KVzip further extends to quantized models and diverse KV cache structures, highlighting its adaptability and efficiency.

Chapter 9

Conclusion

9.1 Summary

The diminishing availability of high-quality internet-sourced training data has led to declining returns from traditional scaling approaches, posing significant hurdles for continued advancements in deep learning. This dissertation addresses two fundamental data-centric challenges: enhancing effective learning in settings with scarce labeled data, and efficiently managing infinite streams of data. We introduce novel data optimization methodologies specifically designed to enhance the adaptability and efficiency of deep learning models.

A core contribution is our synthetic data generation framework, which produces informative and high-quality training samples, significantly reducing reliance on extensive labeled datasets. In Chapter 3, we propose Puzzle Mix, a saliency-guided data augmentation strategy that intelligently combines salient regions from different images to create enhanced training data, substantially improving model generalization [91]. Complementarily, Chapter 4 introduces Co-Mixup, a batch-level augmentation technique that jointly optimizes the saliency and diversity within synthetic data batches. This method further enhances model robustness and improves uncertainty calibration [92]. Experi-

mental evaluations validate that our feedback-driven methodologies enhance generalization and significantly improve uncertainty estimation across multiple domains, including computer vision, speech processing, and natural language understanding.

In Chapter 5, we introduce Neural Relation Graph, a robust framework designed for systematic detection and correction of label noise and effective identification of outliers, markedly improving data quality [94]. The proposed relational structure encodes rich data representations, streamlining and strengthening data preprocessing tasks critical to constructing reliable machine learning systems.

To efficiently handle infinite data streams, we develop advanced compression techniques. Chapter 6 presents Information-Intensive Dataset Condensation, a synthetic-data parameterization approach that compresses large-scale datasets such as ImageNet to merely 1% of their original size while retaining approximately 90% of original training performance [93]. This method significantly enhances computational efficiency, scalability, and sustainability. Additionally, in Chapter 7, our Compressed Context Memory dynamically compresses key-value features in Transformer models, achieving linear complexity sequence processing [97]. This compression approach notably reduces memory usage by up to 5× compared to standard methods, without sacrificing performance, thereby enabling Transformer deployment in resource-constrained environments. Finally, in Chapter 8, we introduced KVzip for compressing KV caches of LLMs at inference time [98]. KVzip effectively identifies redundant KV pairs of the context KV cache in a query-agnostic manner, achieving 3-4× reduction in memory size and 2× reduction in the decoding latency.

Collectively, these contributions form an integrated optimization framework where data enhancements directly translate into improved model performance, establishing a self-reinforcing cycle of continual improvement. These methodologies offer practical benefits and theoretical advancements, laying a solid foundation for future research into adaptive, efficient, and scalable deep learning systems.

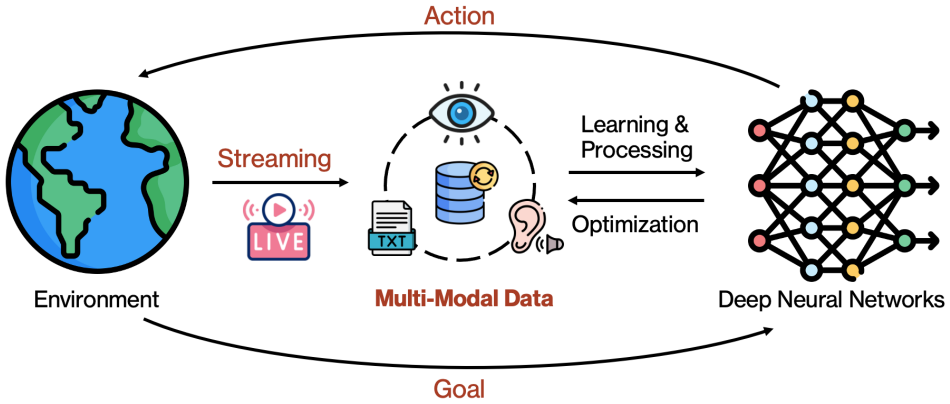


Figure 9.1 Illustration of our long-term vision.

9.2 Future Work

Our long-term vision is to build neural systems capable of continuous adaptation and reasoning within dynamic real-world environments, ultimately bringing societal utilities through advanced AI systems. Advancing AI to handle continuous streaming data, integrate multi-modal sensory information, and actively contribute to scientific discovery will lead to intelligent systems that evolve alongside their operational context. Each direction addresses critical limitations in current AI and promises substantial practical impact and interdisciplinary breakthroughs.

To achieve this vision, we will pursue three interconnected research avenues: (1) neural architectures tailored for continuous stream processing, (2) compressed multi-modal memory systems enhancing scalable perception, and (3) AI-driven methodologies accelerating scientific discovery. As illustrated in Figure 9.1, these areas extend our foundational work in memory compression, synthetic data generation, and data-centric optimization, aiming to develop AI systems with enduring societal benefits.

9.2.1 Streaming-Data Processing

We envision AI systems persistently operating across devices and continuously engaging humans over extended periods. Such systems must manage infinite,

non-stationary input streams, significantly differing from static datasets predominant today. Continuous data processing introduces complexities in memory efficiency, noise tolerance, and long-term reasoning capabilities, essential for applications like lifelong personal assistants, tutors, or collaborative companions. For instance, a personal assistant would continuously adapt to evolving user patterns, dynamically adjusting their knowledge and preferences in real-time.

To realize robust lifelong AI systems, we will develop neural architectures capable of efficiently processing infinite streams without periodic retraining. Specifically, we will explore efficient long-term memory structures for sequential processing, and hybrid architectures combining external memory with deep learning methods. We will conduct experiments involving streaming benchmarks that simulate real-world scenarios, such as continuous sensor data from wearable health monitors. By analyzing performance metrics like accuracy, computational efficiency, and robustness against data drift, we aim to identify optimal combinations of architectural components and learning strategies.

9.2.2 Multi-Modal Memory Systems

To achieve effective human–AI interaction, AI systems require perception across various sensory modalities. Developing memory systems capable of compressing and retrieving integrated visual, auditory, textual, and physical data will transform raw sensory inputs into compact, actionable representations. This will democratize AI by enabling intuitive interactions accessible to all users, including those without technical backgrounds. For example, a personal assistant integrating visual cues from a user's gestures with auditory commands can intuitively respond to complex requests, enhancing everyday accessibility and usability.

We will extend our data optimization framework to multi-modal contexts, focusing on efficiently combining vision, audio, and text streams. Our methodology involves developing selective attention mechanisms tailored to identifying and preserving salient information while compressing redundant data effectively. We will implement and evaluate advanced neural compression methods on datasets capturing multi-modal interactions in realistic environments. The

systems' performance will be assessed through metrics including memory footprint reduction, real-time responsiveness, and accuracy on multi-modal tasks, leading to practical applications such as embodied AI.

9.2.3 AI for Scientific Discovery

AI can substantially accelerate scientific advancement by identifying hidden patterns and generating novel, testable hypotheses from complex datasets. Our goal is to integrate data-driven machine learning with domain-specific scientific models, enabling AI to not only analyze data but actively contribute to theory development and discovery processes. A concrete example of value includes leveraging AI to identify novel genetic markers associated with diseases, directly informing targeted treatments and interventions.

Building upon our previous successes in computer vision and natural language processing, we will advance methodologies integrating environment interactions, causal inference, and simulation-driven reasoning [95]. Our strategy may involve collaborating closely with domain experts to develop models trained on diverse datasets, including genomic sequences and biological simulations. We will systematically validate generated hypotheses through computational experiments and real-world validations in laboratory settings. Success will be measured by the generation of actionable insights or accuracy in predicting biological phenomena, ultimately positioning AI as an important tool in scientific discovery.

Bibliography

[1] A. Agrawal, N. Kedia, A. Panwar, J. Mohan, N. Kwatra, B. Gulavani, A. Tumanov, and R. Ramjee. Taming throughput-latency tradeoff in llm inference with sarathi-serve. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*, 2024.

[2] J. Ainslie, J. Lee-Thorp, M. De Jong, Y. Zemlyanskiy, F. Lebrón, and S. Sanghai. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. *EMNLP*, 2023.

[3] S. Anagnostidis, D. Pavllo, L. Biggio, L. Noci, A. Lucchi, and T. Hofmann. Dynamic context pruning for efficient and interpretable autoregressive transformers. *Advances in Neural Information Processing Systems*, 2023.

[4] J. Ba, G. E. Hinton, V. Mnih, J. Z. Leibo, and C. Ionescu. Using fast weights to attend to the recent past. *NeurIPS*, 2016.

[5] Y. Bai, X. Lv, J. Zhang, H. Lyu, J. Tang, et al. Longbench: A bilingual, multitask benchmark for long context understanding. *ACL*, 2024.

[6] J. Bang, H. Kim, Y. Yoo, J.-W. Ha, and J. Choi. Rainbow memory: Continual learning with a memory of diverse samples. In *CVPR*, 2021.

[7] H. Bao, L. Dong, and F. Wei. Beit: Bert pre-training of image transformers. In *ICLR*, 2022.

[8] E. Barshan, M.-E. Brunet, and G. K. Dziugaite. Relatif: Identifying explanatory training samples via relative influence. In *AISTATS*, 2020.

[9] S. Basu, P. Pope, and S. Feizi. Influence functions in deep learning are fragile. In *ICLR*, 2021.

[10] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan. A theory of learning from different domains. *Machine learning*, 79(1-2):151–175, 2010.

[11] L. Beyer, O. J. Hénaff, A. Kolesnikov, X. Zhai, and A. v. d. Oord. Are

we done with imagenet? *arXiv preprint arXiv:2006.07159*, 2020.

[12] C. M. Bishop. Training with noise is equivalent to tikhonov regularization. *Neural computation*, 7(1):108–116, 1995.

[13] C. M. Bishop. *Pattern recognition and machine learning*. springer, 2006.

[14] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on pattern analysis and machine intelligence*, 23(11):1222–1239, 2001.

[15] T. Brown, B. Mann, N. Ryder, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 2020.

[16] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. In *NeurIPS*, 2020.

[17] A. Bulatov, Y. Kuratov, and M. Burtsev. Recurrent memory transformer. *NeurIPS*, 2022.

[18] Z. Cai, Y. Zhang, B. Gao, Y. Liu, T. Liu, K. Lu, et al. Pyramidkv: Dynamic kv cache compression based on pyramidal information funneling. *arXiv preprint arXiv:2406.02069*, 2024.

[19] G. Cazenavette, T. Wang, A. Torralba, A. A. Efros, and J.-Y. Zhu. Dataset distillation by matching training trajectories. *arXiv preprint arXiv:2203.11932*, 2022.

[20] B. J. Chan, C.-T. Chen, J.-H. Cheng, and H.-H. Huang. Don't do rag: When cache-augmented generation is all you need for knowledge tasks. *arXiv preprint arXiv:2412.15605*, 2024.

[21] Character.AI. Optimizing ai inference at character.ai, 2024. URL `https://research.character.ai/optimizing-inference/`.

[22] Y. Chen, G. Wang, J. Shang, S. Cui, Z. Zhang, T. Liu, S. Wang, Y. Sun, D. Yu, and H. Wu. Nacl: A general and effective kv cache eviction framework for llms at inference time. *ACL*, 2024.

[23] Z. Chen, R. Sadhukhan, Z. Ye, Y. Zhou, J. Zhang, et al. Magicpig: Lsh sampling for efficient llm generation. *ICLR*, 2025.

[24] A. Chevalier, A. Wettig, A. Ajith, and D. Chen. Adapting language models to compress contexts. *arXiv preprint arXiv:2305.14788*, 2023.

[25] R. Child, S. Gray, A. Radford, and I. Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

[26] R. Child, S. Gray, A. Radford, and I. Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

[27] D. Chong, J. Hong, and C. D. Manning. Detecting label errors using pre-trained language models. *arXiv preprint arXiv:2205.12702*, 2022.

[28] P. Chrabaszcz, I. Loshchilov, and F. Hutter. A downsampled variant of imagenet as an alternative to the cifar datasets. *arXiv preprint arXiv:1707.08819*, 2017.

[29] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, and A. Vedaldi. Describing textures in the wild. In *CVPR*, 2014.

[30] K. Clark, K. Guu, M.-W. Chang, P. Pasupat, G. Hinton, and M. Norouzi. Meta-learning fast weight language models. *EMNLP*, 2022.

[31] K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[32] T. Computer. Redpajama: an open dataset for training large language models. `https://github.com/togethercomputer/RedPajama-Data`, 2023.

[33] N. Courty, R. Flamary, D. Tuia, and A. Rakotomamonjy. Optimal transport for domain adaptation. *IEEE transactions on pattern analysis and machine intelligence*, 2016.

[34] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le. Autoaugment: Learning augmentation strategies from data. *CVPR*, 2019.

[35] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. V. Le, and R. Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *ACL*, 2019.

[36] T. Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. *ICLR*, 2024.

[37] J. Davis and M. Goadrich. The relationship between precision-recall and roc curves. In *ICML*, 2006.

[38] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and F. F. Li. Imagenet: a large-scale hierarchical image database. *CVPR*, 2009.

[39] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[40] T. DeVries and G. W. Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.

[41] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021.

[42] G. Erkan and D. R. Radev. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of artificial intelligence research*,

22:457–479, 2004.

[43] Y. Feng, J. Lv, Y. Cao, X. Xie, and S. K. Zhou. Ada-kv: Optimizing kv cache eviction by adaptive budget allocation for efficient llm inference. *arXiv preprint arXiv:2407.11550*, 2024.

[44] S. Fujishige. *Submodular functions and optimization*. Elsevier, 2005.

[45] L. Gao, S. Biderman, S. Black, et al. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

[46] S. Ge, Y. Zhang, L. Liu, M. Zhang, J. Han, and J. Gao. Model tells you what to discard: Adaptive kv cache compression for llms. *ICLR*, 2024.

[47] T. Ge, J. Hu, X. Wang, S.-Q. Chen, and F. Wei. In-context autoencoder for context compression in a large language model. *arXiv preprint arXiv:2307.06945*, 2023.

[48] R. Geirhos, P. Rubisch, C. Michaelis, M. Bethge, F. A. Wichmann, and W. Brendel. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. *ICLR*, 2019.

[49] A. Ghorbani and J. Zou. Data shapley: Equitable valuation of data for machine learning. In *ICML*, 2019.

[50] Y. Gong, Y.-A. Chung, and J. Glass. Ast: Audio spectrogram transformer. In *Interspeech*, 2021.

[51] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *NeurIPS*, 2014.

[52] A. Goyal, A. Lamb, Y. Zhang, S. Zhang, A. Courville, and Y. Bengio. Professor forcing: A new algorithm for training recurrent networks. *Advances in neural information processing systems*, 29, 2016.

[53] gradientAI. Llama-3 8b gradient instruct 1048k, 2024. URL `https://huggingface.co/gradientai/Llama-3-8B-Instruct-Gradient-1048k`.

[54] A. Grattafiori, A. Dubey, A. Jauhri, A. Pandey, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[55] A. Gruslys, R. Munos, I. Danihelka, M. Lanctot, and A. Graves. Memory-efficient backpropagation through time. *NeurIPS*, 2016.

[56] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger. On calibration of modern neural networks. *In ICML*, 2017.

[57] H. Guo, Y. Mao, and R. Zhang. Mixup as locally linear out-of-manifold regularization. *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019.

[58] J. Hartmanis. Computers and intractability: a guide to the theory of np-

completeness (michael r. garey and david s. johnson). *Siam Review*, 24 (1), 1982.

[59] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CVPR*, 2015.

[60] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *In CVPR*, 2015.

[61] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. *ECCV*, 2016.

[62] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick. Masked autoencoders are scalable vision learners. In *CVPR*, 2022.

[63] D. Hendrycks and T. Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *ICLR*, 2019.

[64] D. Hendrycks and K. Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *ICLR*, 2017.

[65] D. Hendrycks, N. Mu, E. D. Cubuk, B. Zoph, J. Gilmer, and B. Lakshminarayanan. AugMix: A simple data processing method to improve robustness and uncertainty. *ICLR*, 2020.

[66] D. Hendrycks, S. Basart, M. Mazeika, M. Mostajabi, J. Steinhardt, and D. Song. Scaling out-of-distribution detection for real-world settings. In *ICML*, 2022.

[67] T. Horel and Y. Singer. Maximization of approximately submodular functions. *In NeurIPS*, 2016.

[68] C.-P. Hsieh, S. Sun, S. Kriman, S. Acharya, D. Rekesh, F. Jia, Y. Zhang, and B. Ginsburg. Ruler: What's the real context size of your long-context language models? *COLM*, 2024.

[69] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. Lora: Low-rank adaptation of large language models. *ICLR*, 2022.

[70] N. Hu, E. Mitchell, C. D. Manning, and C. Finn. Meta-learning online adaptation of language models. *arXiv preprint arXiv:2305.15076*, 2023.

[71] W. Hu, Z. Li, and D. Yu. Simple and effective regularization methods for training on noisily labeled data with generalization guarantee. In *ICLR*, 2021.

[72] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. *In CVPR*, 2016.

[73] J. Huang and D. Mumford. Statistics of natural images and models. In *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*, volume 1, pages

541–547. IEEE, 1999.

[74] D. Hutchins, I. Schlag, Y. Wu, E. Dyer, and B. Neyshabur. Block-recurrent transformers. *NeurIPS*, 2022.

[75] A. Ilyas, S. M. Park, L. Engstrom, G. Leclerc, and A. Madry. Datamodels: Predicting predictions from training data. In *ICML*, 2022.

[76] R. Iyer and J. Bilmes. Algorithms for approximate minimization of the difference between submodular functions, with applications. *arXiv preprint arXiv:1207.0560*, 2012.

[77] S. Jastrzebski, Z. Kenton, N. Ballas, A. Fischer, Y. Bengio, and A. Storkey. On the relation between the sharpest directions of dnn loss and the sgd step length. In *ICLR*, 2019.

[78] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, et al. Mistral 7b, 2023.

[79] H. Jiang, Y. Li, C. Zhang, Q. Wu, X. Luo, et al. Minference 1.0: Accelerating pre-filling for long-context llms via dynamic sparse attention. *Advances in Neural Information Processing Systems*, 2024.

[80] L. Jiang, Z. Zhou, T. Leung, L.-J. Li, and L. Fei-Fei. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *ICML*, 2018.

[81] R. Jonker and A. Volgenant. A shortest augmenting path algorithm for dense and sparse linear assignment. *Computing*, 38(4):325–340, 1987.

[82] C. Jung and C. Kim. A unified spectral-domain approach for saliency detection and its application to automatic object segmentation. *IEEE Transactions on Image Processing*, 21(3):1272–1283, 2011.

[83] O. Kalinli and S. S. Narayanan. A saliency-based auditory attention model with applications to unsupervised prominent syllable detection in speech. In *Eighth Annual Conference of the International Speech Communication Association*, 2007.

[84] O. Kalinli and S. S. Narayanan. A saliency-based auditory attention model with applications to unsupervised prominent syllable detection in speech. In *Eighth Annual Conference of the International Speech Communication Association*, 2007.

[85] G. Kamradt. Needle in a haystack-pressure testing llms, 2023.

[86] N. Karim, M. N. Rizve, N. Rahnavard, A. Mian, and M. Shah. Unicon: Combating label noise through uniform selection and contrastive learning. In *CVPR*, 2022.

[87] A. Katharopoulos, A. Vyas, N. Pappas, and F. Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. *ICML*, 2020.

[88] P. W. Katz. Zip file format specification, 1989. URL `https://pkware.cachefly.net/webdocs/casestudies/APPNOTE.TXT`.

[89] B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *The Bell system technical journal*, 49(2), 1970.

[90] H. Kim, J. Hessel, L. Jiang, X. Lu, et al. Soda: Million-scale dialogue distillation with social commonsense contextualization. *EMNLP*, 2023.

[91] J.-H. Kim, W. Choo, and H. O. Song. Puzzle mix: Exploiting saliency and local statistics for optimal mixup. *In ICML*, 2020.

[92] J.-H. Kim, W. Choo, H. Jeong, and H. O. Song. Co-mixup: Saliency guided joint mixup with supermodular diversity. In *ICLR*, 2021.

[93] J.-H. Kim, J. Kim, S. J. Oh, S. Yun, H. Song, J. Jeong, J.-W. Ha, and H. O. Song. Dataset condensation via efficient synthetic-data parameterization. In *ICML*, 2022.

[94] J.-H. Kim, S. Yun, and H. O. Song. Neural relation graph: A unified framework for identifying label noise and outlier data. *Advances in Neural Information Processing Systems*, 36, 2023.

[95] J.-H. Kim, C. S. Gibbs, S. Yun, H. O. Song, and K. Cho. Large-scale targeted cause discovery with data-driven learning. *arXiv preprint arXiv:2408.16218*, 2024.

[96] J.-H. Kim, J. Yeom, S. Yun, and H. O. Song. Compressed context memory for online language model interaction. *ICLR*, 2024.

[97] J.-H. Kim, J. Yeom, S. Yun, and H. O. Song. Compressed context memory for online language model interaction. *ICLR*, 2024.

[98] J.-H. Kim, J. Kim, S. Kwon, J. W. Lee, S. Yun, and H. O. Song. Kvzip: Query-agnostic kv cache compression with context reconstruction. *arXiv preprint arXiv:2505.23416*, 2025.

[99] S. Kim, S. Shen, D. Thorsley, A. Gholami, W. Kwon, J. Hassoun, and K. Keutzer. Learned token pruning for transformers. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2022.

[100] P. W. Koh and P. Liang. Understanding black-box predictions via influence functions. In *ICML*, 2017.

[101] P. W. Koh, S. Sagawa, H. Marklund, S. M. Xie, M. Zhang, A. Balsubramani, W. Hu, M. Yasunaga, R. L. Phillips, I. Gao, et al. Wilds: A benchmark of in-the-wild distribution shifts. In *ICML*, 2021.

[102] V. Kolmogorov and R. Zabih. What energy functions can be minimizedvia graph cuts? *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 26(2):147–159, 2004.

[103] A. Krizhevsky and H. Geoffrey. Learning multiple layers of features from tiny images. *University of Toronto*, 2009.

[104] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *NeurIPS*, 2012.

[105] J. Kuan and J. Mueller. Model-agnostic label quality scoring to detect real-world label errors. In *ICML DataPerf Workshop*, 2022.

[106] W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C. H. Yu, J. Gonzalez, H. Zhang, and I. Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, 2023.

[107] A. Lazaridou, A. Kuncoro, E. Gribovskaya, et al. Mind the gap: Assessing temporal generalization in neural language models. *NeurIPS*, 34, 2021.

[108] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553), 2015.

[109] K. Lee, K. Lee, H. Lee, and J. Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In *NeurIPS*, 2018.

[110] K. Lee, K. Lee, J. Shin, and H. Lee. Network randomization: A simple technique for generalization in deep reinforcement learning. *In ICLR*, 2020.

[111] W. Lee, J. Lee, J. Seo, and J. Sim. Infinigen: Efficient generative inference of large language models with dynamic kv cache management. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2024.

[112] Y. Li, H. Su, X. Shen, W. Li, Z. Cao, and S. Niu. Dailydialog: A manually labelled multi-turn dialogue dataset. *IJCNLP*, 2017.

[113] Y. Li, Y. Huang, B. Yang, B. Venkitesh, A. Locatelli, H. Ye, T. Cai, P. Lewis, and D. Chen. Snapkv: Llm knows what you are looking for before generation. *Advances in Neural Information Processing Systems*, 2024.

[114] Y. Li, H. Wen, W. Wang, X. Li, Y. Yuan, G. Liu, et al. Personal llm agents: Insights and survey about the capability, efficiency and security. *arXiv preprint arXiv:2401.05459*, 2024.

[115] Y. Li, H. Jiang, Q. Wu, X. Luo, S. Ahn, C. Zhang, A. H. Abdi, D. Li, J. Gao, Y. Yang, et al. Scbench: A kv cache-centric analysis of long-context methods. *ICLR*, 2025.

[116] S. Liang, Y. Li, and R. Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks. In *ICLR*, 2018.

[117] Y. Lin, H. Tang, S. Yang, Z. Zhang, G. Xiao, C. Gan, and S. Han. Qserve: W4a8kv4 quantization and system co-design for efficient llm serving. *arXiv preprint arXiv:2405.04532*, 2024.

[118] D. Liu, M. Chen, B. Lu, H. Jiang, Z. Han, Q. Zhang, et al. Retrievalattention: Accelerating long-context llm inference via vector retrieval. *arXiv preprint arXiv:2409.10516*, 2024.

[119] W. Liu, X. Wang, J. Owens, and Y. Li. Energy-based out-of-distribution detection. In *NeurIPS*, 2020.

[120] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[121] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie. A convnet for the 2020s. In *CVPR*, 2022.

[122] Z. Liu, A. Desai, F. Liao, W. Wang, V. Xie, Z. Xu, A. Kyrillidis, and A. Shrivastava. Scissorhands: Exploiting the persistence of importance hypothesis for llm kv cache compression at test time. *Advances in Neural Information Processing Systems*, 2023.

[123] Z. Liu, J. Wang, T. Dao, T. Zhou, B. Yuan, Z. Song, et al. Deja vu: Contextual sparsity for efficient llms at inference time. In *International Conference on Machine Learning*, 2023.

[124] Z. Liu, J. Yuan, H. Jin, S. Zhong, Z. Xu, V. Braverman, B. Chen, and X. Hu. Kivi: A tuning-free asymmetric 2bit quantization for kv cache. *ICML*, 2024.

[125] L. Lovász. Submodular functions and convexity. In *Mathematical Programming The State of the Art*, pages 235–257. Springer, 1983.

[126] D. Maclaurin, D. Duvenaud, and R. Adams. Gradient-based hyperparameter optimization through reversible learning. In *ICML*, 2015.

[127] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[128] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online dictionary learning for sparse coding. In *ICML*, 2009.

[129] J. Manyika. An overview of bard: an early experiment with generative ai. Technical report, Technical report, Google AI, 2023.

[130] G. J. McLachlan and T. Krishnan. *The EM algorithm and extensions*, volume 382. John Wiley & Sons, 2007.

[131] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas. Communication-efficient learning of deep networks from decentralized

data. In *AISTATS*, 2017.

[132] S. Min, M. Lewis, L. Zettlemoyer, and H. Hajishirzi. Metaicl: Learning to learn in context. *NAACL*, 2022.

[133] T. Mitchell, W. Cohen, E. Hruschka, et al. Never-ending learning. *Communications of the ACM*, 61(5), 2018.

[134] A. Mohtashami and M. Jaggi. Landmark attention: Random-access infinite context length for transformers. *arXiv preprint arXiv:2305.16300*, 2023.

[135] J. Mu, X. L. Li, and N. Goodman. Learning to compress prompts with gist tokens. *arXiv preprint arXiv:2304.08467*, 2023.

[136] J. Munkres. Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial and Applied Mathematics*, 1957.

[137] M. Narasimhan and J. A. Bilmes. A submodular-supermodular procedure with applications to discriminative structure learning. *UAI*, 2005.

[138] T. Nguyen, R. Novak, L. Xiao, and J. Lee. Dataset distillation with infinitely wide convolutional networks. In *NeurIPS*, 2021.

[139] C. Northcutt, L. Jiang, and I. Chuang. Confident learning: Estimating uncertainty in dataset labels. *Journal of Artificial Intelligence Research*, 70, 2021.

[140] C. G. Northcutt, A. Athalye, and J. Mueller. Pervasive label errors in test sets destabilize machine learning benchmarks. In *NeurIPS Datasets and Benchmarks Track*, 2021.

[141] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

[142] OpenAI. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[143] M. Oren, M. Hassid, N. Yarden, Y. Adi, and R. Schwartz. Transformers are multi-state rnns. *arXiv preprint arXiv:2401.06104*, 2024.

[144] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113, 2019.

[145] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113, 2019.

[146] W. Park, D. Kim, Y. Lu, and M. Cho. Relational knowledge distillation. In *CVPR*, 2019.

[147] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in

pytorch. *NeurIPS*, 2017.

[148] D. Patterson, J. Gonzalez, Q. Le, C. Liang, L.-M. Munguia, D. Rothchild, D. So, M. Texier, and J. Dean. Carbon emissions and large neural network training. *arXiv preprint arXiv:2104.10350*, 2021.

[149] M. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. Deep contextualized word representations. In *NAACL*, 2018.

[150] J. Phang, Y. Mao, P. He, and W. Chen. Hypertuning: Toward adapting large language models without back-propagation. *ICML*, 2023.

[151] J. M. Phillips. Coresets and sketches. *arXiv preprint arXiv:1601.00617*, 2016.

[152] K. J. Piczak. ESC: Dataset for Environmental Sound Classification. In *Proceedings of the 23rd Annual ACM Conference on Multimedia*, 2015.

[153] G. Pleiss, T. Zhang, E. Elenberg, and K. Q. Weinberger. Identifying mislabeled data using the area under the margin ranking. In *NeurIPS*, 2020.

[154] A. Prabhu, P. H. Torr, and P. K. Dokania. Gdumb: A simple approach that questions our progress in continual learning. In *ECCV*, 2020.

[155] G. Pruthi, F. Liu, S. Kale, and M. Sundararajan. Estimating training data influence by tracing gradient descent. In *NeurIPS*, 2020.

[156] X. Qi, A. Panda, K. Lyu, X. Ma, S. Roy, A. Beirami, P. Mittal, and P. Henderson. Safety alignment should be made more than just a few tokens deep. *ICLR*, 2025.

[157] Z. Qin and D. Kim. Rethinking softmax with cross-entropy: Neural network classifier as mutual information estimator. *arXiv preprint arXiv:1911.10688*, 2019.

[158] J. Rabin, S. Ferradans, and N. Papadakis. Adaptive color transfer with relaxed optimal transport. In *2014 IEEE International Conference on Image Processing (ICIP)*, 2014.

[159] A. Radford, J. Wu, R. Child, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 2019.

[160] J. W. Rae, A. Potapenko, S. M. Jayakumar, and T. P. Lillicrap. Compressive transformers for long-range sequence modelling. *ICLR*, 2020.

[161] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. Squad: 100,000+ questions for machine comprehension of text. *EMNLP*, 2016.

[162] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.

[163] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert. icarl: Incremental classifier and representation learning. In *CVPR*, 2017.

[164] S. Reed, H. Lee, D. Anguelov, C. Szegedy, D. Erhan, and A. Rabinovich. Training deep neural networks on noisy labels with bootstrapping. *arXiv preprint arXiv:1412.6596*, 2014.

[165] N. Reimers and I. Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *EMNLP-IJCNLP*, 2019.

[166] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *In NeurIPS*, 2015.

[167] Z. Ren, S. Gao, L.-T. Chia, and I. W.-H. Tsang. Region-based saliency detection and its application in object recognition. *IEEE Transactions on Circuits and Systems for Video Technology*, 24(5):769–779, 2013.

[168] A. Salemi, S. Mysore, M. Bendersky, and H. Zamani. Lamp: When large language models meet personalization. *arXiv preprint arXiv:2304.11406*, 2023.

[169] M. Schmidt and K. Alahari. Generalized fast approximate energy minimization via graph cuts: Alpha-expansion beta-shrink moves. *arXiv preprint arXiv:1108.5710*, 2011.

[170] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *CVPR*, 2015.

[171] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. *ICCV*, 2017.

[172] K. Shuster, J. Xu, M. Komeili, D. Ju, E. M. Smith, S. Roller, M. Ung, M. Chen, K. Arora, J. Lane, et al. Blenderbot 3: a deployed conversational agent that continually learns to responsibly engage. *arXiv preprint arXiv:2208.03188*, 2022.

[173] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[174] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

[175] B. Sluban, D. Gamberger, and N. Lavrač. Ensemble-based noise detection: noise ranking and visual performance evaluation. *Data mining and knowledge discovery*, 28(2), 2014.

[176] C. S. Smith. *Modes of discourse: The local structure of texts*, volume 103. Cambridge University Press, 2003.

[177] C. Snell, D. Klein, and R. Zhong. Learning by distilling context. *arXiv*

preprint arXiv:2209.15189, 2022.

[178] H. O. Song, Y. Xiang, S. Jegelka, and S. Savarese. Deep metric learning via lifted structured feature embedding. In *CVPR*, 2016.

[179] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[180] F. P. Such, A. Rawal, J. Lehman, K. Stanley, and J. Clune. Generative teaching networks: Accelerating neural architecture search by learning to generate synthetic training data. In *ICML*, 2020.

[181] I. Sucholutsky and M. Schonlau. Soft-label dataset distillation and text dataset distillation. In *2021 International Joint Conference on Neural Networks (IJCNN)*, 2021.

[182] M. Sugiyama and K. Borgwardt. Rapid distance-based outlier detection via sampling. In *NeurIPS*, 2013.

[183] Y. Sun, C. Guo, and Y. Li. React: Out-of-distribution detection with rectified activations. In *NeurIPS*, 2021.

[184] Y. Sun, Y. Ming, X. Zhu, and Y. Li. Out-of-distribution detection with deep nearest neighbors. In *ICML*, 2022.

[185] S. Swayamdipta, R. Schwartz, N. Lourie, Y. Wang, H. Hajishirzi, N. A. Smith, and Y. Choi. Dataset cartography: Mapping and diagnosing datasets with training dynamics. In *EMNLP*, 2020.

[186] M. Tan and Q. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *ICML*, 2019.

[187] J. Tang, Y. Zhao, K. Zhu, G. Xiao, B. Kasikci, and S. Han. Quest: Query-aware sparsity for efficient long-context llm inference. *ICML*, 2024.

[188] G. Team, A. Kamath, J. Ferret, S. Pathak, N. Vieillard, et al. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025.

[189] Y. Tian, D. Krishnan, and P. Isola. Contrastive multiview coding. In *ECCV*, 2020.

[190] N. Tishby and N. Zaslavsky. Deep learning and the information bottleneck principle. *Information Theory Workshop*, 2015.

[191] M. Toneva, A. Sordoni, R. T. d. Combes, A. Trischler, Y. Bengio, and G. J. Gordon. An empirical study of example forgetting during deep neural network learning. In *ICLR*, 2019.

[192] H. Touvron, T. Lavril, G. Izacard, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[193] S. Tworkowski, K. Staniszewski, M. Pacek, Y. Wu, H. Michalewski, and P. Miłoś. Focused transformer: Contrastive training for context scaling.

arXiv preprint arXiv:2307.03170, 2023.

[194] G. Van Horn, O. Mac Aodha, Y. Song, Y. Cui, C. Sun, A. Shepard, H. Adam, P. Perona, and S. Belongie. The inaturalist species classification and detection dataset. In *CVPR*, 2018.

[195] V. Vasudevan, B. Caine, R. Gontijo-Lopes, S. Fridovich-Keil, and R. Roelofs. When does dough become a bagel? analyzing the remaining mistakes on imagenet. In *NeurIPS*, 2022.

[196] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *NeurIPS*, 30, 2017.

[197] V. Verma, A. Lamb, C. Beckham, A. Najafi, I. Mitliagkas, A. Courville, D. Lopez-Paz, and Y. Bengio. Manifold mixup: Better representations by interpolating hidden states. *ICML*, 2019.

[198] C. Villani. *Optimal transport: old and new*, volume 338. Springer Science & Business Media, 2008.

[199] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *ICLR*, 2019.

[200] B. Wang, W. Chen, H. Pei, C. Xie, M. Kang, et al. Decodingtrust: A comprehensive assessment of trustworthiness in gpt models. In *NeurIPS*, 2023.

[201] L. Wang, H. Lu, X. Ruan, and M.-H. Yang. Deep networks for saliency detection via local estimation and global search. *CVPR*, 2015.

[202] T. Wang, J.-Y. Zhu, A. Torralba, and A. A. Efros. Dataset distillation. *arXiv preprint arXiv:1811.10959*, 2018.

[203] W. Wang, L. Dong, H. Cheng, X. Liu, X. Yan, J. Gao, and F. Wei. Augmenting language models with long-term memory. *NeurIPS*, 2023.

[204] Y. Wang, W. Liu, X. Ma, J. Bailey, H. Zha, L. Song, and S.-T. Xia. Iterative learning with open-set noisy labels. In *CVPR*, 2018.

[205] P. Warden. Google command datset. *URL https://research.googleblog.com/2017/08/launching-speech-commands-dataset.html.*, 2017.

[206] P. Warden. Speech commands: A dataset for limited-vocabulary speech recognition. *arXiv preprint arXiv:1804.03209*, 2018.

[207] Y. Wei, J. Feng, X. Liang, M.-M. Cheng, Y. Zhao, and S. Yan. Object region mining with adversarial erasing: A simple classification to semantic segmentation approach. *CVPR*, 2017.

[208] M. Welling. Herding dynamical weights to learn. In *ICML*, 2009.

[209] Y. Wen, G. Jerfel, R. Muller, M. W. Dusenberry, J. Snoek, B. Lakshminarayanan, and D. Tran. Improving calibration of batchensemble with data augmentation. *In ICML Workshop on Uncertainty and Robustness in Deep Learning*, 2020.

[210] J. Weston, S. Chopra, and A. Bordes. Memory networks. *ICLR*, 2015.

[211] T. Windheuser, H. Ishikawa, and D. Cremers. Generalized roof duality for multi-label optimization: Optimal lower bounds and persistency. *In ECCV*, 2012.

[212] D. Wingate, M. Shoeybi, and T. Sorensen. Prompt compression and contrastive conditioning for controllability and toxicity reduction in language models. *EMNLP*, 2022.

[213] E. Wong, L. Rice, and J. Z. Kolter. Fast is better than free: Revisiting adversarial training. *ICLR*, 2020.

[214] S. J. Wright. Coordinate descent algorithms. *Mathematical Programming*, 151(1):3–34, 2015.

[215] Y. Wu, M. N. Rabe, D. Hutchins, and C. Szegedy. Memorizing transformers. *ICLR*, 2022.

[216] Z.-F. Wu, T. Wei, J. Jiang, C. Mao, M. Tang, and Y.-F. Li. Ngc: A unified framework for learning with open-world noisy data. In *ICCV*, 2021.

[217] G. Xiao, Y. Tian, B. Chen, S. Han, and M. Lewis. Efficient streaming language models with attention sinks. *arXiv preprint arXiv:2309.17453*, 2023.

[218] G. Xiao, J. Tang, J. Zuo, J. Guo, S. Yang, H. Tang, Y. Fu, and S. Han. Duoattention: Efficient long-context llm inference with retrieval and streaming heads. *ICLR*, 2025.

[219] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *CVPR*, 2010.

[220] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017.

[221] A. Yang, B. Yu, C. Li, D. Liu, F. Huang, H. Huang, et al. Qwen2.5-1m technical report. *arXiv preprint arXiv:2501.15383*, 2025.

[222] D. Yang, X. Han, Y. Gao, Y. Hu, S. Zhang, and H. Zhao. Pyramidinfer: Pyramid kv cache compression for high-throughput llm inference. *arXiv preprint arXiv:2405.12532*, 2024.

[223] J. Yang, K. Zhou, Y. Li, and Z. Liu. Generalized out-of-distribution detection: A survey. *arXiv preprint arXiv:2110.11334*, 2021.

[224] J. Y. Yang, B. Kim, J. Bae, B. Kwon, G. Park, E. Yang, S. J.

Kwon, and D. Lee. No token left behind: Reliable kv cache compression via importance-aware mixed precision quantization. *arXiv preprint arXiv:2402.18096*, 2024.

[225] T. Ye, L. Dong, Y. Xia, Y. Sun, Y. Zhu, G. Huang, and F. Wei. Differential transformer. *ICLR*, 2025.

[226] B. Yekkehkhany, A. Safari, S. Homayouni, and M. Hasanlou. A comparison study of different kernel functions for svm-based classification of multi-temporal polarimetry sar data. *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 2014.

[227] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. *ICCV*, 2019.

[228] S. Zagoruyko and N. Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

[229] M. Zaheer, G. Guruganesh, K. A. Dubey, et al. Big bird: Transformers for longer sequences. *NeurIPS*, 2020.

[230] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.

[231] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3), 2021.

[232] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz. mixup: Beyond empirical risk minimization. *ICLR*, 2018.

[233] S. Zhang, S. Roller, N. Goyal, et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

[234] X. Zhang, Y. Chen, S. Hu, Z. Xu, J. Chen, et al. ∞bench: Extending long context evaluation beyond 100k tokens. *ACL*, 2024.

[235] Y. Zhang, M. Pezeshki, P. Brakel, S. Zhang, C. L. Y. Bengio, and A. Courville. Towards end-to-end speech recognition with deep convolutional neural networks. *arXiv preprint arXiv:1701.02720*, 2017.

[236] Z. Zhang, Y. Sheng, T. Zhou, T. Chen, L. Zheng, et al. H2o: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in Neural Information Processing Systems*, 2023.

[237] B. Zhao and H. Bilen. Dataset condensation with distribution matching. *arXiv preprint arXiv:2110.04181*, 2021.

[238] B. Zhao and H. Bilen. Dataset condensation with differentiable siamese augmentation. In *ICML*, 2021.

[239] B. Zhao, K. R. Mopuri, and H. Bilen. Dataset condensation with gradient matching. In *ICLR*, 2021.

[240] R. Zhao, W. Ouyang, H. Li, and X. Wang. Saliency detection by multi-context deep learning. *CVPR*, 2015.

[241] L. Zheng, W.-L. Chiang, Y. Sheng, et al. Lmsys-chat-1m: A large-scale real-world llm conversation dataset. *arXiv preprint arXiv:2309.11998*, 2023.

[242] W. Zhong, L. Guo, Q. Gao, and Y. Wang. Memorybank: Enhancing large language models with long-term memory. *arXiv preprint arXiv:2305.10250*, 2023.

[243] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Learning deep features for discriminative localization. *CVPR*, 2016.

[244] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba. Places: A 10 million image database for scene recognition. *IEEE transactions on pattern analysis and machine intelligence*, 40(6), 2017.

[245] B. Zoph and Q. V. Le. Neural architecture search with reinforcement learning. In *ICLR*, 2017.

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Professor Hyun Oh Song, for his invaluable guidance and unwavering support throughout this research. His dedication, insightful advice, and encouragement in critical thinking have significantly fostered my intellectual growth and academic development. I am especially grateful for the countless hours he devoted to providing constructive feedback and for consistently motivating me to overcome challenges and expand my research capabilities. I extend my appreciation to my committee members for their thoughtful comments and constructive suggestions, which greatly enhanced the quality of my work.

Special thanks go to my family for their continuous support, patience, and understanding throughout the writing of this thesis. In particular, I deeply appreciate my wife, In-Young Ko, for her enduring encouragement and support.

Finally, I would like to acknowledge the Korea Foundation for Advanced Studies (KFAS) for providing the scholarship that made my PhD studies possible.

요약

딥러닝은 계산 역량을 획기적으로 변화시켜 컴퓨터 비전, 자연어 처리, 음성 인식 등 다양한 분야에서 혁신을 가능하게 했습니다 [142]. 엄선된 대규모 데이터셋으로 훈련된 ResNet과 GPT와 같은 대표적인 모델들은 이러한 놀라운 발전을 잘 보여주며, 여러 작업에서 이전에 없던 높은 성능을 일관되게 달성하고 인간 수준 이상의 정확도를 기록하고 있습니다 [60, 16].

이러한 눈부신 발전에도 불구하고 최근 딥러닝 연구는 상당한 도전에 직면해 있습니다. 특히, 딥러닝 모델들이 인터넷에서 얻을 수 있는 대부분의 고품질 데이터를 이미 소진함에 따라, 단순히 데이터 규모를 늘리는 방식의 성능 향상은 점점 한계에 이르고 있습니다 [142]. 한편, 모델이 동적 상호작용을 통해 적응적으로 학습하고, 전문적이고 데이터가 부족한 환경에서도 효과적으로 작동하는 새로운 가능성이 등장하고 있습니다 [15]. 실시간으로 상호작용하는 AI 시스템과 고도로 전문화된 산업 분야 등 이러한 새로운 응용 환경들은 기존 딥러닝 패러다임이 제대로 대응하기 어려운 독특한 도전을 제기합니다.

본 논문은 딥러닝 프레임워크에서 데이터가 가지는 핵심적 역할에 주목하여 이러한 중대한 도전들을 체계적으로 다룹니다. 구체적으로, 제한된 라벨 데이터를 활용한 효과적인 모델 학습과 무한한 데이터 스트림을 효율적으로 처리하고 활용하는 두 가지 주요 데이터 중심의 도전을 다룹니다. 이러한 문제를 극복하기 위해 본 연구는 딥러닝 시스템의 효율성과 적응성을 모두 향상시키기 위해 특별히 설계된 혁신적인 데이터 최적화 방법론을 제안합니다.

본 연구의 핵심적인 기여 중 하나는 유익하고 고품질의 훈련 데이터를 생성하여 대규모 라벨 데이터셋 의존성을 완화하는 합성 데이터 생성 프레임워크입니다 [91, 92]. 이에 더하여, 라벨 오류 및 이상치를 식별하고 수정하거나 제거함으로써 훈련 데이터의 무결성과 신뢰성을 보장하는 종합적인 대규모 데이터 정제 프레임

워크를 소개합니다 [94]. 또한 Transformer 기반 아키텍처에 최적화된 고급 데이터 압축 방법을 개발하여 온라인 상호작용과 메모리 사용 관리의 효율성을 크게 향상시킴으로써, 실시간으로 상호작용하는 실제 환경에서의 활용 가능성을 높였습니다 [93, 97, 98].

이러한 기여들은 데이터 개선이 직접적으로 모델 성능 향상을 촉진하고, 다시 모델의 성능 향상이 데이터 최적화 전략을 더욱 정교하게 만드는 공동 최적화 프레임워크를 구성합니다. 이와 같은 반복적이고 자기 강화적인 순환 구조는 딥러닝 모델이 제한된 라벨 데이터와 무한히 풍부한 데이터 스트림 모두를 효과적으로 활용하여 강력한 학습과 추론을 가능케 합니다.

본 연구를 통해 개발되고 실증적으로 검증된 방법론들은 실질적인 활용 가능성을 크게 높이는 동시에 의미 있는 이론적 통찰을 제공합니다. 기존 딥러닝 프레임워크에 원칙에 기반한 확장을 제안함으로써 본 연구는 향후 연구를 위한 탄탄한 기초를 마련하며, 적응적이고 효율적이며 확장 가능한 딥러닝 시스템 개발을 위한 새로운 연구 방향을 제시합니다.

주요어: 딥러닝, 이미지 인식, 자연어 처리, 합성 데이터 생성, 장기 기억 장치, 효율적 추론 시스템
학번: 2019-26471