UCLA
UCLA Electronic Theses and Dissertations

Title
Towards Algorithm and Data Efficient Deep Learning

Permalink
https://escholarship.org/uc/item/1bs8s6pp

Author
Xiong, Yuanhao

Publication Date
2024

Peer reviewed|Thesis/dissertation

eScholarship.org Powered by the California Digital Library
University of California

UNIVERSITY OF CALIFORNIA

Los Angeles

Towards Algorithm and Data Efficient Deep Learning

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Computer Science

by

Yuanhao Xiong

2024

© Copyright by

Yuanhao Xiong

2024

ABSTRACT OF THE DISSERTATION

Towards Algorithm and Data Efficient Deep Learning

by

Yuanhao Xiong
Doctor of Philosophy in Computer Science
University of California, Los Angeles, 2024
Professor Cho-Jui Hsieh, Chair

Deep Learning has transformed our interaction with the world significantly, with numerous breakthroughs in the fields of computer vision, natural language understanding, autonomous driving and more. We have witnessed great success of large models in capturing intrinsic patterns and representations present in the training data. However, despite prosperous development, we are faced with challenges such as computational constraints and the lack of high-quality annotated data when scaling up models. Consequently, it is necessary to design both algorithm and data efficient methods to address these issues. For algorithm efficiency, we explore techniques including meta-learning and dataset distillation to reduce training time. On the other hand, for data efficiency, we show performance improvement by pre-training on unannotated noisy datasets followed by fine-tuning. Specifically, we investigate representation learning and language modeling, two prevalent frameworks to enhance the utilization of pre-training data. In the meanwhile, we propose an automatic data annotation pipeline to further enable model and data co-development. With all the efforts in efficient deep learning, we make it feasible and practical to train a well-performed model efficiently.

The dissertation of Yuanhao Xiong is approved.

Yizhou Sun

Wei Wang

Kai-Wei Chang

Cho-Jui Hsieh, Committee Chair

University of California, Los Angeles

2024

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

This doctoral journey has been an incredible, challenging and rewarding experience, one that would not have been possible without the support and contributions of many individuals and organizations along the way.

First and foremost, I owe a profound debt of gratitude to my advisor, Professor Cho-Jui Hsieh, whose guidance, wisdom, and unwavering belief in me have been invaluable. His passion for research, attention to detail, and commitment to excellence have shaped my growth as a researcher and scholar. I am truly fortunate to have had the opportunity to work with him during my Ph.D. career.

I extend my sincere appreciation to the members of my dissertation committee – Professor Wei Wang, Professor Yizhou Sun, and Professor Kai-Wei Chang – for their insightful feedback, thought-provoking questions, and constructive critiques throughout this process. Their expertise and perspectives have enriched this work immensely.

My heartfelt thanks go to my colleagues and friends I have met and worked with, whose support and encouragement, stimulating discussions, and collaborative spirit made this journey infinitely more enjoyable.

I am grateful to the Department of Computer Science and the broader University of California, Los Angeles community for providing the necessary resources, facilities, and support that enabled me to pursue my research endeavors.

Words cannot express my gratitude to my parents – whose unconditional love, encouragement, and unwavering faith in me have been my constant source of strength. Their sacrifices and support have made this achievement possible.

Finally, to all those who have contributed, directly or indirectly, to this dissertation, I express my sincerest gratitude. Your support has been the driving force behind this achievement, and I am forever indebted to you.

VITA

2015–2019	B.S. (Information Engineering), Zhejiang University.
2019–2021	M.S. (Computer Science), UCLA.

PUBLICATIONS

Yuanhao Xiong, Cho-Jui Hsieh. Improved Adversarial Training via Learned Optimizer. ECCV. 2020.

Yuanhao Xiong, Li-Cheng Lan, Xiangning Chen, Ruochen Wang, Cho-Jui Hsieh. Learning to Schedule Learning Rate with Graph Neural Networks. ICLR. 2022.

Yuanhao Xiong, Wei-Cheng Chang, Cho-Jui Hsieh, Hsiang-Fu Yu, Inderjit Dhillon. Extreme Zero-Shot Learning for Extreme Text Classification. NAACL. 2022.

Yuanhao Xiong, Cho-Jui Hsieh. Learning to Learn with Smooth Regularization. ECCV. 2022.

Yuanhao Xiong, Ruochen Wang, Minhao Cheng, Felix Yu, Cho-Jui Hsieh. FedDM: Iterative Distribution Matching for Communication-efficient Federated Learning. CVPR. 2023.

Yuanhao Xiong, Long Zhao, Boqing Gong, Ming-Hsuan Yang, Florian Schroff, Ting Liu, Cho-Jui Hsieh, Liangzhe Yuan. FedDM: Structured Video-Language Modeling with Temporal Grouping and Spatial Grounding . ICLR. 2024.

CHAPTER 1

Introduction

1.1 Background

Deep learning [GBC16] has emerged as one of the most transformative technologies in past few decades, revolutionizing fields ranging from computer vision and natural language processing to robotics and healthcare. The success of deep learning models can be largely attributed to their ability to automatically learn complex patterns and representations from large amounts of training data [KMH20, HBM22]. However, the insatiable appetite of these models for massive datasets and immense computational resources [KMH20, HBM22] poses significant challenges to scalability, efficiency, and accessibility, hampering their widespread application and real-world deployment. As we move towards an era of ubiquitous artificial intelligence (AI) systems, there is a pressing need to develop efficient deep learning techniques.

In particular, efficiency in deep learning encompasses a multifaceted approach, addressing not only the computational costs associated with training but also the judicious utilization of data resources [Men23]. Traditional deep learning paradigms often rely on massive datasets and compute-intensive algorithms, which raises concerns in scenarios where where computational resources are constrained, or where annotated data collection is limited or expensive. Therefore, the pursuit of algorithm and data efficiency in deep learning endeavors to mitigate these problems, enabling the development of leaner, more sustainable, and adaptable models.

1.2 Algorithm Efficiency

Algorithm efficiency refers to the ability of models to learn effectively with fewer computational resources, lower memory footprints, and faster training times. By streamlining model architectures, refining optimization algorithms, and leveraging novel training paradigms, researchers aim to reduce the computational burden associated with deep learning workflows, thereby enhancing their algorithm efficiency and applicability across diverse domains. Among these potential directions, our research focuses primarily on automatic optimization techniques [XH20], in which the whole or partial model update rules are learned automatically through a meta-learning process. Specifically, "learning to optimize" [ADG16, CCC22] has been adopted in our works to replace traditional optimization algorithms such as PGD for adversarial learning [GSS14, ZYJ19] and Adam for image classification [KB14], and achieved better generalization and faster convergence at the same time. In addition, to circumvent the problem of instability of learned optimizers in longer training, we have taken an alternative path to automate only learning rate scheduling, one essential part of the complete optimization procedure. Apart from improving efficiency in optimization algorithms, we have resorted to dataset distillation [ZMB20, NCL20, WZT18] to accomplish communication-efficient federated learning, reducing message transmission between the client and the server significantly.

1.3 Data Efficiency

Data efficiency, on the other hand, pertains to the capability of models to learn generalizable representations from limited training data, reducing the reliance on massive, manually labeled datasets. Although deep learning's ability to extract complex patterns from data has fueled breakthroughs in image recognition, natural language processing, and a multitude of other domains, this power comes at a cost. Traditional deep learning models are often data hungry, requiring massive datasets for training, and computationally expensive, demanding significant resources to run [GBC16, Men23]. These limitations hinder deep learning's potential severely. For better data efficiency, there are a number of algorithms to leverage several kinds of information such as unlabeled data, data from another domain, and prior knowledge. Due to the abundance of web-crawled noisy data without annotations, first pre-training the model on such a large-scale dataset followed by downstream adaptation becomes a typical and prevalent approach. In this dissertation, we explore the pre-training and fine-tuning from single modality of language only to multiple ones of vision and language. Models pre-trained on different tasks including contrastive learning and language modeling have been investigated and both of them have demonstrated impressive performance given limited data for fine-tuning in downstream evaluation. Moreover, multimodal large models have made automatic annotation and model-dataset co-development possible, democratizing an efficient collection of high-quality data.

1.4 Overview

This dissertation covers topics from algorithmic innovations that enhance optimization strategies to data efficient learning approaches such as transfer learning. The significance of achieving algorithm and data efficiency in deep learning extends beyond academic curiosity, resonating deeply with practical considerations in industry, academia, and societal domains. By reducing computational burdens and data dependencies, efficient deep learning methodologies pave the way for deploying AI solutions in resource-constrained environments, empowering edge devices, and democratizing access to AI technologies.

The thesis consists of 9 chapters with Chapter 2-8 as the core contents of this dissertation. In particular, we expand on algorithm efficient deep learning in Part I from Chapter 2-5 and concentrate on data efficient methods in Part II from Chapter 6-8. An overview of all these chapters are as follows:

- Chapter 1 introduces the background and landscape of algorithm and data efficient deep learning, and presents an overview of the dissertation.
- Chapter 2 explores the adaptation of learned optimizers to adversarial training, replacing the original PGD for faster training.
- Chapter 3 discusses how to stabilize the training of neural optimizers via adversarial regularization for further improvement on final performance and convergence rate.
- Chapter 4 introduces a reinforcement learning based approach to model the learning rate scheduler, reducing manual labor on hyperparameter tuning notably.
- Chapter 5 describes how to leverage dataset distillation to achieve communication-efficient federated learning.
- Chapter 6 investigates a pre-training method to deal with extreme text classification in zero-shot scenario where limited supervision is provided.
- Chapter 7 refines the contrastive learning in the video-language domain, with a focus on utilizing fine-grained structures in pre-training data.
- Chapter 8 presents a unified causal video-oriented language modeling framework and introduces how to use the obtained model to automatically annotate internet videos with temporally-aligned captions for better data efficiency.
- Chapter 9 concludes this dissertation and outlines future research directions.

Part I

Algorithm Efficient Deep Learning

CHAPTER 2

Improved Adversarial Training via Learned Optimizer

In this chapter, we describe how to learn a neural optimizer for adversarial training to achieve better performance in terms of both model robustness and training speed against the widely-adopted PGD optimizer.

2.1 Introduction

It has been widely acknowledged that deep neural networks (DNN) have made tremendous breakthroughs benefiting both academia and industry. Despite being effective, many DNN models trained with benign inputs are vulnerable to small and undetectable perturbation added to original data and tend to make wrong predictions under such threats. Those perturbed examples, also known as adversarial examples, can be easily constructed by algorithms such as DeepFool [MFF16], Fast Gradient Sign Method (FGSM) [GSS14], and Carlini-Wagner, i.e., C&W attack [CW17]. Moreover, such adversarial attacks can also be conducted in the black-box setting [BRB17, CLC18, CSC20] and can appear naturally in the physical world [HZB19, KGB16a]. This phenomenon can bring about serious consequences in domains such as face recognition and autonomous-driving. Therefore, how to train a model resistant to adversarial inputs has become an important topic.

A variety of defense methods have been proposed to improve the performance of DNNs against adversarial attacks [KGB16b, SKC18, WY19, WZ19, XWM19, ZYJ19]. Among them, adversarial training [KGB16b] stands out for its effectiveness. Moreover, [MMS17] shows that adversarial training can be formulated as a minimax optimization problem, resembling a game between the attacker and the defender. The formulation is so intuitive that the inner problem aims at generating adversarial examples by maximizing the training loss while the outer one guides the network in the direction that minimizes the loss to resist attacks. However, directly obtaining the optimal value of the inner maximization is infeasible, so one has to run an iterative optimization algorithm for a fixed number (often 10) iterations to get an approximate inner maximizer.

Existing adversarial training often uses hand-designed general purpose optimizers, such as PGD attack, to (approximately) solve the inner maximization. However, there is an essential property of adversarial training that is rarely explored: the maximization problems associated with each sample share very similar structure, and a good inner maximizer for adversarial training only needs to work well for this set of data-dependent problems. To be specific, there are a finite of n maximization problems need to be solved (where n is number of training samples), and those maximization problems share the same objective function along with identical network structure and weights, and the only difference is their input x. Based on this observation, can we have a better optimizer that in particular works well for these very similar and data-dependent problems?

Motivated by this idea, we propose a learned optimizer for improved adversarial training. Instead of using an existing optimizer with a fixed update rule (such as PGD), we aim at learning the inner maximizer that could be faster and more effective for this particular set of maximization problems. We have noticed that two works have already put forward algorithms to combine learning to learn with adversarial training [JCS18, JZH19]. Both of them adopt a convolutional neural network (CNN) generator to produce malicious perturbations whereas CNN structure might complicate the training process and cannot grasp the essence of the update rule in the long term. In contrast, we propose an L2L-based adversarial training method with recurrent neural networks (RNN). RNN is capable of capturing long-term dependencies and has shown great potentials in predicting update directions and steps adaptively [LJL17]. Thus, following the framework in [ADG16], we leverage RNN as the optimizer to generate perturbations in a coordinate-wise manner. Based on the properties of the inner problem, we tailor our RNN optimizer with removed bias and weighted loss for further elaborations to ameliorate issues like short-horizon in L2L [WRL18].

Specifically, our main contributions in this chapter are summarized as follows:

- We first investigate and confirm the improvement in the model robustness from stronger attacks by searching a suitable step size for PGD.
- In replacement of hand-designed algorithms like PGD, an RNN-based optimizer based on the properties of the inner problem is designed to learn a better update rule. In addition to standard adversarial training, the proposed algorithm can also be applied to any other minimax defense objectives such as TRADES [ZYJ19].
- Experimental results show that the proposed method can noticeably improve the robust accuracy of both adversarial training [MMS17] and TRADES [ZYJ19]. Furthermore, our RNN-based adversarial training significantly outperforms previous CNN-based one and requires much less number of trainable parameters.

2.2 Related Work

2.2.1 Adversarial Attack and Defense

Model robustness has recently become a great concern for deploying deep learning models in real-world applications. Goodfellow *et al.* [GSS14] succeeded in fooling the model to make wrong predictions by Fast Gradient Sign Method (FGSM). Subsequently, to produce adversarial examples, IFGSM and Projected Gradient Descent (PGD) [GSS14, MMS17] accumulate attack strength through running FGSM iteratively, and Carlini-Wagner (C&W) attack [CW17] designs a specific objective function to increase classification errors. Besides these conventional optimization-based methods, there are several algorithms focusing on generating malicious perturbations via neural networks [ROG18, XLZ18]. For instance, Xiao *et al.* [XLZ18] exploit GAN, which is originally designed for crafting deceptive images, to output corresponding noises added to benign iuput data. The appearance of various attacks has pushed forward the development of effective defense algorithms to train neural networks that are resistant to adversarial examples. The seminal work of adversarial training has significantly improved adversarial robustness [MMS17]. It has inspired the emergence of various advanced defense algorithms: TRADES [ZYJ19] is designed to minimize a theoretically-driven upper bound and GAT [LHL17] takes generator-based outputs to train the robust classifier. All these methods can be formulated as a minimax problem [MMS17], where the defender makes efforts to mitigate negative effects (outer minimization) brought by adversarial examples from the attacker (inner maximization). Whereas, performance of such an adversarial game is usually constrained by the quality of solutions to the inner problem [JZH19, JCS18]. Intuitively, searching a better maxima for the inner problem can improve the solution of minimax training, leading to improved defensive models.

2.2.2 Learning to Learn

Recently, learning to learn emerges as a novel technique to efficiently address a variety of problems such as automatic optimization [ADG16], few-shot learning [FAL17], and neural architecture search [EMH18]. In this chapter, we emphasize on the subarea of L2L: how to learn an optimizer for better performance. Rather than using human-defined update rules, learning to learn makes use of neural networks for designing optimization algorithms automatically. It is developed originally from [CC90] and [YHC01], in which early attempts are made to model adaptive algorithms on simple convex problems. More recently, [ADG16] proposes an LSTM optimizer for some complex optimization problems, such as training a convolutional neural network classifier. Based on this work, elaborations in [LJL17] and [WMH17] further improve the generalization and scalability for learned optimizers. Moreover, [RXR19] demonstrates that a zeroth order optimizer can also be learned using L2L. Potentials of learning-to-learn motivates a line of L2L-based defense which replaces hand-designed methods for solving the inner problem with neural network optimizers. [JCS18] uses a CNN generator mapping clean images into corresponding perturbations. Since it only makes one-step and deterministic attack like FGSM, [JZH19] modifies the algorithm and produces stronger and more diverse attacks iteratively. Unfortunately, due to the large number of parameters and the lack of ability to capture the long-term dependencies, the CNN generator adds too much difficulty in the optimization, especially for the minimax problem in adversarial training. Therefore, we adopt an RNN optimizer in our method for a more stable training process as well as a better grasp of the update rule.

2.3 Preliminaries

2.3.1 Notations

We use bold lower-case letters $\boldsymbol{x}$ and $\boldsymbol{y}$ to represent clean images and their corresponding labels. An image classification task is considered in this chapter with the classifier f parameterized by θ. $\operatorname{sign}(\cdot)$ is an element-wise operation to output the sign of a given input with $\operatorname{sign}(0) = 1$. $\mathbb{B}(\boldsymbol{x}, \epsilon)$ denotes the neighborhood of $\boldsymbol{x}$ as well as the set of admissible perturbed images: $\{\boldsymbol{x}' : \|\boldsymbol{x}' - \boldsymbol{x}\|_\infty \leq \epsilon\}$, where the infinity norm is adopted as the distance metric. We denote by Π the projection operator that maps perturbed data to the feasible set. Specifically, $\Pi_{\mathbb{B}(\boldsymbol{x},\epsilon)}(\boldsymbol{x}') = \max(\boldsymbol{x} - \epsilon, \min(\boldsymbol{x}', \boldsymbol{x} + \epsilon))$, which is an element-wise operator. $\mathcal{L}(\cdot, \cdot)$ is a multi-class loss like cross-entropy.

2.3.2 Adversarial Training

In this part, we present the formulation of adversarial training, together with some hand-designed optimizers to solve this problem. To obtain a robust classifier against adversarial attacks, an intuitive idea is to minimize the robust loss, defined as the worst-case loss within a small neighborhood $\mathbb{B}(\boldsymbol{x}, \epsilon)$. Adversarial training, which aims to find the weights that minimize the robust loss, can be formulated as a minimax optimization problem in the following way [MMS17]:

$$\min_{\theta} \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim D} \left\{ \max_{\boldsymbol{x}' \in \mathbb{B}(\boldsymbol{x},\epsilon)} \mathcal{L}(f(\boldsymbol{x}'), \boldsymbol{y}) \right\}, \tag{2.1}$$

where D is the empirical distribution of input data. However, Equation (2.1) only focuses on accuracy over adversarial examples and might cause severe over-fitting issues on the training set. To address this problem, TRADES [ZYJ19] investigates the trade-off between natural and robust errors and theoretically puts forward a different objective function for adversarial training:

$$\min_{\theta} \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim D} \left\{ \mathcal{L}(f(\boldsymbol{x}), \boldsymbol{y}) + \max_{\boldsymbol{x}' \in \mathbb{B}(\boldsymbol{x},\epsilon)} \mathcal{L}(f(\boldsymbol{x}), f(\boldsymbol{x}'))/\lambda \right\}. \tag{2.2}$$

Note that Equation (2.1) and Equation (2.2) are both defined as minimax optimization problems, and to solve such saddle point problems, a commonly used approach is to first get an approximate solution $\boldsymbol{x}'$ of inner maximization based on the current θ, and then use $\boldsymbol{x}'$ to conduct updates on model weights θ. The adversarial training procedure then iteratively runs this on each batch of samples until convergence. Clearly, the quality and efficiency of inner maximization is crucial to the performance of adversarial training. The most commonly used inner maximizer is the projected gradient descent algorithm, which conducts a fixed number of updates:

$$\boldsymbol{x}'_{t+1} = \Pi_{\mathbb{B}(\boldsymbol{x},\epsilon)}(\alpha \, \mathrm{sign}(\nabla_{\boldsymbol{x}'}\mathcal{L}(\boldsymbol{x}'_t)) + \boldsymbol{x}'_t). \tag{2.3}$$

$\mathcal{L}(\boldsymbol{x}'_t)$ represents the maximization term in Equation (2.1) or 2.2 with abuse of notation.

2.3.3 Effects of Adaptive Step Sizes

We found that the performance of adversarial training crucially depends on the optimization algorithm used for inner maximization, and the current widely used PGD algorithm may not be the optimal choice. Here we demonstrate that even a small modification of PGD and without any change to the adversarial training objective can boost the performance of model robustness. We use the CNN structure in [ZYJ19] to train a classifier on MNIST dataset. When 10-step PGD (denoted by PGD for simplicity) is used for the inner maximization, a constant step size is always adopted, which may not be suitable for the subsequent update. Therefore, we make use of backtracking line search (BLS) to select a step size adaptively for adversarial training (AdvTrain as abbreviation). Starting with a maximum candidate step size value α_0, we iteratively decrease it by $\alpha_t = \rho\alpha_{t-1}$ until the following condition is satisfied:

$$\mathcal{L}(\boldsymbol{x}' + \alpha_t \boldsymbol{p}) \geq \mathcal{L}(\boldsymbol{x}') + c\alpha_t \boldsymbol{p}^{\mathrm{T}}\boldsymbol{p} \tag{2.4}$$

where $\boldsymbol{p} = \nabla_{\boldsymbol{x}'}\mathcal{L}(\boldsymbol{x}')$ is a search direction. Based on a selected control parameter $c \in (0, 1)$, the condition tests whether the update with step size α_t leads to sufficient increase in the objective function, and it is guaranteed that a sufficiently small α will satisfy the condition so line search will always stop in finite steps. This is standard in gradient ascent (descent) optimization, and see more discussions in [NW06]. Following the convention, we set $\rho = 0.5$ and $c = 10^{-4}$. As shown in Table 2.1, defense with AdvTrain+BLS leads to

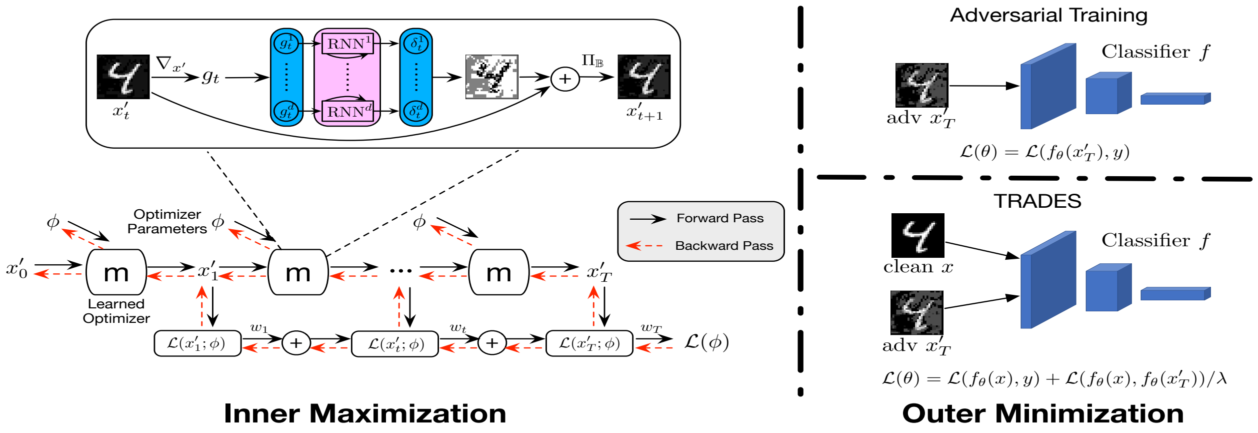


Figure 2.1: Model architecture of our defense method with an RNN optimizer

a more robust model than solving the inner problem only by PGD (88.71% vs 87.33%). At the same time the attacker combined with BLS generates stronger adversarial examples: the robust accuracy of the model trained from vanilla adversarial training drops over 1.2% with PGD+BLS, compared to merely PGD attack. This experiment motivates our efforts to find a better inner maximizer for adversarial training.

Table 2.1: Effects of the inner solution quality on robust accuracy (%)

Defense \ Attack	Natural	PGD	PGD+BLS
AdvTrain	96.43	87.33	86.09
AdvTrain+BLS	**96.70**	**88.71**	**88.00**

2.4 Proposed Algorithm

2.4.1 Learning to Learn for Adversarial Training

As mentioned in the previous section, it can be clearly seen that the inner maximizer plays an important role in the performance of adversarial training. However, despite the effectiveness of BLS introduced in Section 2.3.3, it is impractical to combine it with adversarial training as multiple line searches together with loss calculation in this algorithms increase its computational burden significantly. Then a question arises naturally: is there any automatic way for determining a good step size for inner maximization without too much computation overhead? Moreover, apart from the step size, the question can be extended to whether such a maximizer can be learned for a particular dataset and model in replacement of a general optimizer like PGD. Recently, as a subarea of learning-to-learn, researchers have been investigating whether it is possible to use machine learning, especially neural networks, to learn improved optimizer to replace the hand-designed optimizer [ADG16, LJL17, WMH17]. However, it is commonly believed that those ML-learned general-purpose optimizers are still not practically useful due to several unsolved issues. For instance, the exploded gradient [MMN19] in unrolled optimization impedes generalization of these learned optimizers to longer steps and truncated optimization on the other hand induces short-horizon bias [WRL18].

In this chapter, we show it is possible and practical to learn an optimizer for inner maximization in adversarial training. Note that in adversarial training, the maximization problems share very similar form: $\max_{\boldsymbol{x}' \in \mathbb{B}(\boldsymbol{x},\epsilon)} \mathcal{L}(f(\boldsymbol{x}'), \boldsymbol{y})$, where they all have the same loss function $\mathcal{L}$ and the same network (structure and weights) f, and the only difference is their input $\boldsymbol{x}$ and label $\boldsymbol{y}$. Furthermore, we only need the maximizer to perform well on a fixed set of n optimization problems for adversarial training. These properties thus enable us to learn a better optimizer that outperforms PGD.

To allow a learned inner maximizer, we parameterize the learned optimizer by an RNN network. This is following the literature of learning-to-learn [ADG16], but we propose several designs as shown below that works better for our inner maximization problem which is a constrained optimization problem instead of a standard unconstrained training task in [ADG16]. We then jointly optimize the classifier parameters (θ) as well as the parameters of the inner maximizer (ϕ). The overall framework can be found in Figure 2.1.

Specifically, the inner problem is to maximize vanilla adversarial training loss in Equation (2.1) or TRADES loss in Equation (2.2), with a constraint that $\boldsymbol{x}' \in \mathbb{B}(\boldsymbol{x}, \epsilon)$. With an RNN optimizer m parameterized by ϕ, we propose the following parameterized update rule to mimic the PGD update rule in Equation (2.3):

$$\boldsymbol{\delta}_t, \boldsymbol{h}_{t+1} = m_\phi(\boldsymbol{g}_t, \boldsymbol{h}_t), \quad \boldsymbol{x}'_{t+1} = \Pi_{\mathbb{B}(\boldsymbol{x},\epsilon)}\left(\boldsymbol{x}'_t + \boldsymbol{\delta}_t\right). \tag{2.5}$$

Here, $\boldsymbol{g}_t$ is the gradient $\nabla_{\boldsymbol{x}'}\mathcal{L}(f(\boldsymbol{x}'), \boldsymbol{y})$ and $\boldsymbol{h}_t$ is the hidden state representation. It has to be emphasized that our RNN optimizer generates perturbations coordinate-wisely, in contrast to other L2L based methods which take as input the entire image. This property reduces trainable parameters significantly, making it much easier and faster for training. In addition, note that the hidden state of our RNN optimizer plays an important role in the whole optimization. A separate hidden state for each coordinate guarantees the different update behavior. And it contains richer information like the trajectory of loss gradients mentioned in [JZH19] but can produce a recursive update with a simpler structure.

For the RNN design, we mainly follow the structure in [ADG16] but with some modifications to make it more suitable to adversarial training. We can expand the computation of perturbation for each step as:

$$\boldsymbol{\delta}_t = \tanh(\boldsymbol{V}\boldsymbol{h}_t + \boldsymbol{b}_1), \tag{2.6}$$

$$\boldsymbol{h}_{t+1} = \tanh(\boldsymbol{U}\boldsymbol{g}_t + \boldsymbol{W}\boldsymbol{h}_t + \boldsymbol{b}_2) \tag{2.7}$$

where $\boldsymbol{h}_t \in \mathbb{R}^d$, $\boldsymbol{V} \in \mathbb{R}^{1\times d}$, $\boldsymbol{U} \in \mathbb{R}^{d\times 1}$, $\boldsymbol{W} \in \mathbb{R}^{d\times d}$, $\boldsymbol{b}_1 \in \mathbb{R}$ and $\boldsymbol{b}_2 \in \mathbb{R}^d$ in the coordinate-wise update manner. As the optimization proceeds, the gradient will become much smaller when approaching the local maxima. At that time, a stable value of the perturbation is expected without much change between two consecutive iterations. However, from Equation (2.6) and Equation (2.7), we can clearly see that despite small $\boldsymbol{g}_t$, the update rule will still produce an update with magnitude proportional to $\tanh(\boldsymbol{b}_1)$. Imagine the case where the exact optimal value is found with an all-zero hidden state ($\boldsymbol{b}_2$ needs to be zero as well), $\boldsymbol{\delta}_t = \tanh(\boldsymbol{b}_1)$ with a non-zero bias will push the adversarial example away from the optimal one. Thus, two bias terms $\boldsymbol{b}_1$ and $\boldsymbol{b}_2$ are problematic for optimization close to the optimal solution. Due to the short horizon of the inner maximization in adversarial training, it is unlikely for the network to learn zero bias terms. Therefore, to ensure stable training, we remove the bias terms in the vanilla RNN in all implementations.

With an L2L framework, we simultaneously train the RNN optimizer parameters ϕ and the classifier weights θ. The joint optimization problem can be formulated as follows:

$$\min_{\theta} \ \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim D} \left\{ \mathcal{L}(f_\theta(\boldsymbol{x}'_T(\phi^*)), \boldsymbol{y}) \right\} \tag{2.8}$$

$$\text{s.t. } \ \phi^* = \arg\max \mathcal{L}(\phi) \tag{2.9}$$

where $\boldsymbol{x}'_T(\phi^*)$ is computed by running Equation (2.5) T times iteratively. Since the learned optimizer aims at finding a better solution to the inner maximization term, the objective function for training it in the horizon T is defined as:

$$\mathcal{L}(\phi) = \sum_{t=1}^{T} w_t \mathcal{L}(f_\theta(\boldsymbol{x}'_t(\phi)), \boldsymbol{y}). \tag{2.10}$$

If we set $w_t = 0$ for all $t < T$ and $w_T = 1$, then Equation (2.10) implies that our learned maximizer m_ϕ will maximize the loss after T iterations. However, in practice we found that considering intermediate iterations can further improve the performance since it will make the maximizer converge faster even after conducting one or few iterations. Therefore in the experiments we set an increased weights $w_t = t$ for $t = 1, \ldots, T$. Note that [MMN19] showed that this kind of unrolled optimization may lead to some issues such as exploded gradients which is still an unsolved problem in L2L. However, in adversarial training we only need to set a relative small T (e.g., $T = 10$) so we do not encounter that issue.

While updating the learned optimizer, corresponding adversarial examples are produced together. We can then train the classifier by minimizing the loss accordingly. The whole algorithm is presented in Algorithm 1.

Algorithm 1 RNN-based adversarial training

```
1: Input: clean data $\{(\boldsymbol{x}, \boldsymbol{y})\}$, batch size $B$, step sizes $\alpha_1$ and $\alpha_2$, number of inner iterations $T$, classifier parameterized by $\theta$, RNN optimizer parameterized by $\phi$
2: Output: Robust classifer $f_\theta$, learned optimizer $m_\phi$
3: Randomly initialize $f_\theta$ and $m_\phi$, or initialize them with pre-trained configurations
4: repeat
5:     Sample a mini-batch $M$ from clean data.
6:     for $(\boldsymbol{x}, \boldsymbol{y})$ in $B$ do
7:         Initialization: $\boldsymbol{h}_0 \leftarrow 0$, $\mathcal{L}_\theta \leftarrow 0$, $\mathcal{L}_\phi \leftarrow 0$
8:         Gaussian augmentation: $\boldsymbol{x}'_0 \leftarrow \boldsymbol{x} + 0.001 \cdot \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
9:         for $t = 0, \ldots, T-1$ do
10:            $\boldsymbol{g}_t \leftarrow \nabla_{\boldsymbol{x}'} \mathcal{L}(f_\theta(\boldsymbol{x}'_t), \boldsymbol{y})$
11:            $\boldsymbol{\delta}_t, \boldsymbol{h}_{t+1} \leftarrow m_\phi(\boldsymbol{g}_t, \boldsymbol{h}_t)$, where coordinate-wise update is applied
12:            $\boldsymbol{x}'_{t+1} \leftarrow \Pi_{\mathbb{B}(\boldsymbol{x},\epsilon)}(\boldsymbol{x}'_t + \boldsymbol{\delta}_t)$
13:            $\mathcal{L}_\phi \leftarrow \mathcal{L}_\phi + w_{t+1}\mathcal{L}(f_\theta(\boldsymbol{x}'_{t+1}), \boldsymbol{y})$, where $w_{t+1} = t + 1$
14:        end for
15:        $\mathcal{L}_\theta \leftarrow \mathcal{L}_\theta + \mathcal{L}(f_\theta(\boldsymbol{x}'_T), \boldsymbol{y})$
16:    end for
17:    Update $\phi$ by $\phi \leftarrow \phi + \alpha_1 \nabla_\phi \mathcal{L}_\phi / B$
18:    Update $\theta$ by $\theta \leftarrow \theta - \alpha_2 \nabla_\theta \mathcal{L}_\theta / B$
19: until training converged
```

2.4.2 Advantages over Other L2L-based Methods

Previous methods have proposed to use a CNN generator [JZH19, JCS18] to produce perturbations in adversarial training. However, CNN-based generator has a larger number of trainable parameters, which makes it hard to train. In Table 2.2, the detailed properties including the number of parameter and training time per epoch are provided for different learning-to-learn based methods. We can observe that our proposed RNN approach stands out with the smallest parameters as well as efficiency in training. Specifically, our RNN optimizer only has 120 parameters, almost 5000 times fewer than L2LDA while the training time per epoch is 268.50s (RNN-TRADES only consumes 443.52s per training epoch) v.s. 1972.41s. Furthermore, our method also leads to better empirical performance, as shown in our main comparison in Table 2.3, 2.4 and 2.5.

Table 2.2: Comparion among different L2L-based methods

	Number of parameters	Training time per epoch (s)
RNN-Adv	**120**	**268.50**
RNN-TRADES	**120**	443.52
L2LDA	500944	1972.41

2.5 Experimental Results

In this section, we present experimental results of our proposed RNN-based adversarial training. We compare our method with various baselines on both white-box and black-box attack. In addition, different datasets and network architectures are evaluated.

2.5.1 Experimental Settings

- **Datasets and classifier networks.** We mainly use MNIST [LBB98] and CIFAR-10 [KNH10] datasets for performance evaluation in our experiments. For MNIST, the CNN architecture with four convolutional layers in [CW17] is adopted as the classifier. For CIFAR-10, we use the standard VGG-16 [SZ14] and Wide ResNet [ZK16], which has been used in most of the previous defense papers including adversarial training [MMS17] and TRADES [ZYJ19].

- **Baselines for Comparison.** Note that our method is an optimization framework irrelevant to what minimax objective function is used. Thus we choose two popular minimax formulations, AdvTrain[1] [MMS17] and TRADES[2] [ZYJ19], and substitute

[1]https://github.com/xuanqing94/BayesianDefense

[2]https://github.com/yaodongyu/TRADES

the proposed L2L-based optimization for their original PGD-based algorithm. Moreover, we also compare with a previous L2L defense mechanism L2LDA[3] [JZH19] which outperforms other L2L-based methods for thorough comparison. We use the source code provided by the authors on github with their recommended hyper-parameters for all these baseline methods.

- **Evaluation and implementation details.** Defense algorithms are usually evaluated by classification accuracy under different attacks. Effective attack algorithms including PGD, C&W and the attacker of L2LDA are used for evaluating the model robustness, with the maximum ℓ_∞ perturbation strength $\epsilon = 0.3$ for MNIST and $\epsilon = 8/255$ for CIFAR-10. For PGD, we run 10 and 100 iterations (PGD-10 and -100) with the step size $\eta = \epsilon/4$, as suggested in [JZH19]. C&W is implemented with 100 iterations in the infinity norm. For L2LDA attacker, it is learned from L2LDA [JZH19] under different settings with 10 attack steps. In addition, we also uses the learned optimizer of RNN-Adv to conduct 10-step attacks.

For our proposed RNN-based defense, we use a one-layer vanilla RNN with the hidden size of 10 as the optimizer for the inner maximization. Since we test our method under two different minimax losses, we name them as RNN-Adv and RNN-TRADES respectively. The classifier and the optimizer are updated alternately according to the Algorithm 1. All algorithms are implemented in PyTorch-1.1.0 with four NVIDIA 1080Ti GPUs. Note that all adversarial training methods adopt 10-step inner optimization for fair comparison. We run each defense method five times with different random seeds and report the lowest classification accuracy.

[3]https://github.com/YunseokJANG/l2l-da

2.5.2 Performance on White-box Attacks

We demonstrate the robustness of models trained from different defense methods under the white-box setting in this part. Experimental results are shown in Table 2.3, 2.4 and 2.5. From these three tables, we can observe that our proposed L2L-based adversarial training with RNN always outperforms its counterparts.

Table 2.3: Robust accuracy under white-box attacks (MNIST, 4-layer CNN)

Defense \ Attack	Natural	PGD-10	PGD-100	CW100	L2LDA	RNN-Adv	Min
Plain	99.46	1.04	0.42	83.63	5.94	0.79	0.42
AdvTrain	99.17	94.89	94.28	98.38	95.83	94.39	94.28
TRADES	**99.52**	95.77	95.50	98.72	96.03	95.50	95.50
L2LDA	98.76	94.73	93.22	97.69	95.28	93.16	93.16
RNN-Adv	99.20	95.80	95.62	98.75	96.05	95.51	95.51
RNN-TRADES	99.46	**96.09**	**95.83**	**98.85**	**96.56**	**95.80**	**95.80**

Table 2.4: Robust accuracy under white-box attacks (CIFAR-10, VGG-16)

Defense \ Attack	Natural	PGD-10	PGD-100	CW100	L2LDA	RNN-Adv	Min
Plain	**93.66**	0.74	0.09	0.08	0.89	0.43	0.08
AdvTrain	81.11	42.32	40.75	42.26	43.55	41.07	40.75
TRADES	78.08	48.83	48.30	45.94	49.94	48.38	45.95
L2LDA	77.47	35.49	34.27	35.31	36.27	34.54	34.27
RNN-Adv	81.22	44.98	42.89	43.67	46.20	43.21	42.89
RNN-TRADES	80.76	**50.23**	**49.42**	**47.23**	**51.29**	**49.49**	**47.23**

To be specific, our method achieves 95.80% robust accuracy among various attacks on MNIST dataset. On CIFAR-10, RNN-TRADES reaches 47.23% and 54.11 for VGG-16 and Wide ResNet with 1.28% and 1.43% gain over other baselines. It should be stressed that our method surpasses L2LDA (the previous CNN-based L2L method) noticeably. For conventional defense algorithms, our L2L-based variant improves the original method by $1\% - 2\%$

Table 2.5: Robust accuracy under white-box attakcs (CIFAR-10, WideResNet)

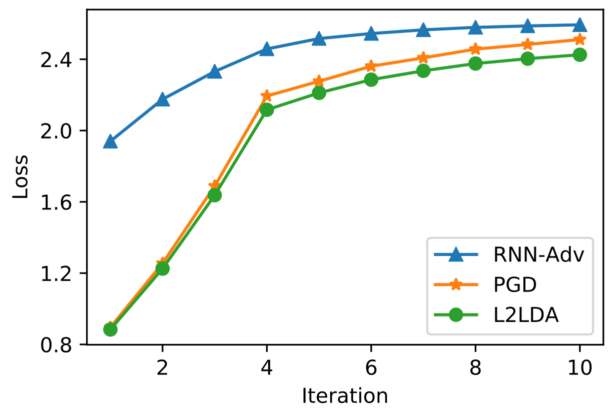
Defense \ Attack	Natural	PGD-10	PGD-100	CW100	L2LDA	RNN-Adv	Min
Plain	**95.14**	0.01	0.00	0.00	0.02	0.00	0.00
AdvTrain	86.28	46.64	45.13	46.64	48.46	45.41	45.13
TRADES	85.89	54.28	52.68	53.68	56.49	53.00	52.68
L2LDA	85.30	45.47	44.35	44.19	47.16	44.54	44.19
RNN-Adv	85.92	47.62	45.98	47.26	49.40	46.23	45.98
RNN-TRADES	84.21	**56.35**	**55.68**	**54.11**	**58.86**	**55.80**	**54.11**

percents under different attacks from comparison of robust accuracy in AdvTrain and RNN-Adv. A similar phenomenon can also be observed in TRADES and RNN-TRADES. Since previous works of L2L-based defense only concentrate on PGD-based adversarial training, the substantial performance gain indicates that the learned optimizer can contribute to the minimax problem in TRADES as well. Furthermore, apart from traditional attack algorithms, we leverage our RNN optimizer learned from adversarial training as the attacker (the column RNN-Adv). Results in three experiments show that compared with other general attackers when conducting 10 iterations such as PGD-10 and L2LDA, ours is capable of producing much stronger perturbations which lead to low robust accuracy.

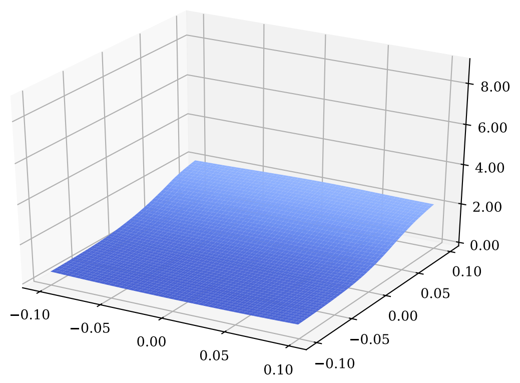
2.5.3 Analysis

Learned Optimizer. As mentioned in Section 2.5.2, the optimizer learned from PGD-based adversarial training can be regarded as an special attacker. Thus, we primarily investigate the update trajectories of different attackers to obtain an in-depth understanding of our RNN optimizer. For VGG-16 models trained from four defense methods, three attacker are used to generate perturbations in 10 steps respectively and losses are recorded as shown in Figure 2.2.

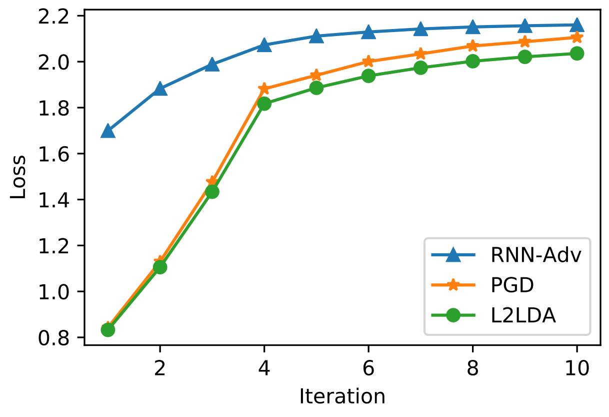
We can see clearly from these four figures that the losses obtained from RNN-Adv are



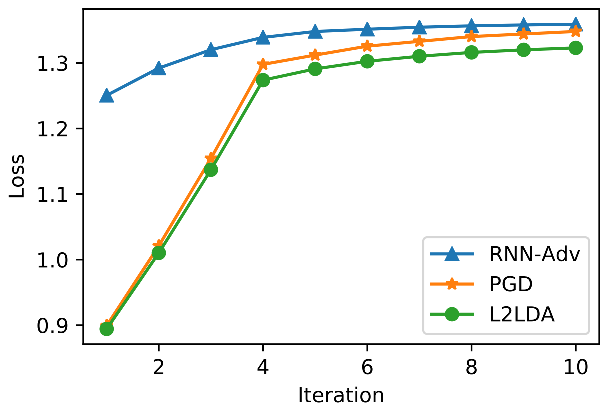
(a) AdvTrain



(b) TRADES



(c) RNN-Adv



(d) RNN-TRADES

Figure 2.2: Comparison of optimization trajectories among various attack algorithms. We evaluate four defense mechanisms, AdvTrain TRADES, RNN-Adv and RNN-TRADES, under three attackers including PGD, L2LDA and our proposed RNN-Adv. All attackers conduct 10-step perturbing process

always larger than others within 10 iterations, reflecting stronger attacks produced by our proposed optimizer. Moreover, it should be noted that the loss gap between RNN-Adv and other attackers is much more prominent at some very beginning iterations. This in fact demonstrates an advantage of the L2O framework that the optimizer can converge faster than hand-designed algorithms.

Generalization to more attack steps. Although our learned RNN optimizer is only trained under 10 steps, we show that it can generalize to more steps as an attacker. From

Table 2.6, we can observe that the attacker is capable of producing much stronger adversarial examples by extending its attack steps to 40. Performance of our attacker is even comparable with that of PGD-100, which further demonstrates the superiority of our proposed method.

2.5.4 Performance on Black-box Transfer Attacks

We further test the robustness of the proposed defense method under transfer attack. As suggested by [ACW18], this can be served as a sanity check to see whether our defense leads to obfuscated gradients and gives a false sense of model robustness. Following procedures in [ACW18], we first train a surrogate model with the same architecture of the target model using a different random seed, and then generate adversarial examples from the surrogate model to attack the target model.

Table 2.6: Generalization to more steps of learned optimizer

Step / Defense	10	40
Plain	0.43	**0.03**
AdvTrain	41.07	**40.70**
TRADES	48.38	**48.27**
L2LDA	34.54	**34.19**
RNN-Adv	43.21	**42.89**
RNN-TRADES	49.49	**49.28**

Specifically, we choose VGG-16 models obtained from various defense algorithms as our target models. In the meanwhile, we train two surrogate models: one is Plain-Net with natural training and the other is PGD-Net with 10-step PGD-based adversarial training. Results are presented in Table 2.7. We can observe that our method outperforms all other baselines, with RNN-PGD and RNN-TRADES standing out in defending attacks from Plain-Net and PGD-Net respectively. It suggests great resistance of our L2L defense

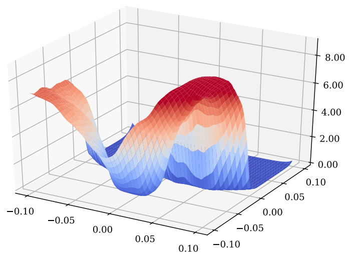
Table 2.7: Robust accuracy under black-box attack settings

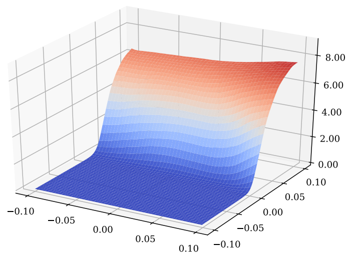
Defense \ Surrogate	Plain-Net	PGD-Net
AdvTrain	79.94	62.57
TRADES	77.01	65.41
L2LDA	76.37	60.32
RNN-Adv	**80.58**	63.17
RNN-TRADES	79.54	**67.09**

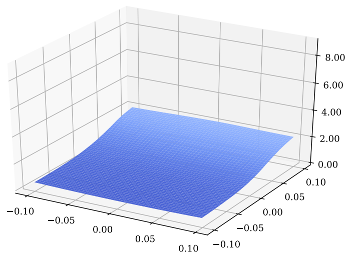
to transfer attacks.

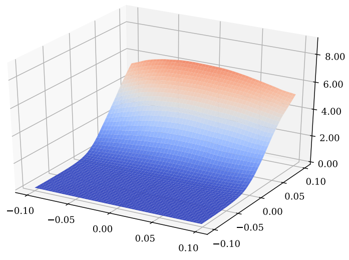
2.5.5 Loss Landscape Exploration

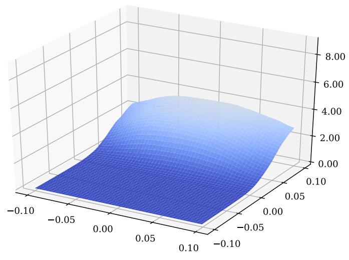
To further verify the superior performance of the proposed algorithm, we visualize the loss landscapes of VGG-16 models trained under different defense strategies, as shown in Figure 2.3. According to the implementation in [EIA18], we modify the input along a linear space defined by the sign of the gradient and a random Rademacher vector, where the x and y axes represent the magnitude of the perturbation added in each direction and the z axis represents the loss. It can be observed that loss surfaces of models trained from RNN-Adv and RNN-TRADES in Figure 2.3e and 2.3f are much smoother than those of their counterparts in Figure 2.3b and 2.3c. Besides, our method significantly reduces the loss value of perturbed data close to the original input. In particular, the maximum loss decreases roughly from 7.61 in adversarial training to 3.66 in RNN-Adv. Compared with L2LDA in Figure 2.3d, the proposed RNN optimizer can contribute to less bumpier loss landscapes with smaller variance, which further demonstrates the stability and superiority of our L2L-based adversarial training.


(a) Plain


(b) Adversarial Training


(c) TRADES


(d) L2LDA


(e) RNN-Adv

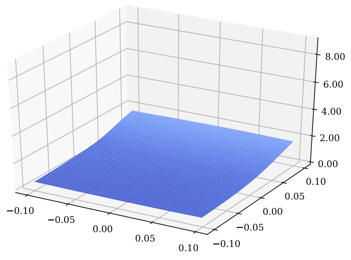

(f) RNN-TRADES

Figure 2.3: Comparison of loss landscapes among different training methods. The color gradually changes from blue (low loss) to red (high loss)

2.6 Conclusion

For defense mechanisms that can be formulated as a minimax optimization problem, we propose to replace the inner PGD-based maximizer with a automatically learned RNN maximizer, and show that jointly training the RNN maximizer and classifier can significantly improve the defense performance. Empirical results demonstrate that the proposed approach can be combined with several minimax defense objectives, including adversarial training and TRADES. For future work, it can be a worthwhile direction to address the inadequacy of L2L in dealing with a long-horizon problem. Then we can substitute the learned optimizer for hand-designed algorithms in both inner and outer problems, which enables an entirely automatic process for adversarial training.

CHAPTER 3

Learning to Learn with Smooth Regularization

Despite impressive performance against traditional optimization algorithms, those learned ones suffer from instability severely. To mitigate the issue, we propose a smooth regularization term via adversarial training to further improve effectiveness and efficiency of learned optimizers.

3.1 Introduction

Optimization is always regarded as one of the most important foundations for deep learning, and its development has pushed forward tremendous breakthroughs in various domains including computer vision and natural language processing [CRK20, LJL17]. Effective algorithms such as SGD [RM51], Adam [KB14] and AdaBound [LXL19] have been proposed to work well on a variety of tasks. In parallel to this line of hand-designed methods, Learning to Learn (L2L) or Learning to Optimizer (L2O) [ADG16, WMH17, MMN18, LJL17, CRK20], a novel framework aimed at an automatic optimization algorithm (optimizer), provides a new direction to performance improvement in updating a target function (optimizee). Typically, the optimizer, modeled as a neural network, takes as input a certain state representation of the optimizee and outputs corresponding updates for parameters. Then such a neural optimizer can be trained like any other network based on specific objective functions.

Empirical results have demonstrated that these learned optimizers can perform better optimization in terms of the final loss and convergence rate than general hand-engineered ones [ADG16, WMH17, MMN18, LJL17, CRK20]. In addition, such advantages in faster training make the learned optimizer a great fit for few-shot learning (FSL) [RL17, HMX20], where only a limited number of labelled examples per class are available for generalizing a classifier to a new task.

However, instability concealed behind the algorithm impedes its development significantly. There are some unsolved issues such as gradient explosion in unrolled optimization [MMN18] and short-horizon bias [WRL18] challenging the promotion of neural optimizers. One of the most essential problems is that contrary to traditional optimizers, the learned ones modeled as neural networks cannot guarantee smoothness with respect to input data. Specifically, an ideal optimizer is expected to conduct similar updates given similar states of the target optimizee. For instance, SGD updates a parameter by a magnitude proportional to its original gradient. Current meta learners neglect this property and suffer from the issue that they would produce a quite different output while merely adding a small perturbation to the input state.

Such a phenomenon has been widely observed in other problems like image classification [GSS14], where the perturbed image can fool the classifier to make a wrong prediction. Inspired by the progress in adversarial training [MMS17] where the worst-case loss is minimized, we propose an algorithm that takes the smoothness of the learned optimizer into account. Through penalizing the non-smoothness by a regularization term, the neural optimizer is trained to capture a smooth update rule with better performance.

In summary, we are the first to consider the smoothness of neural optimizers, and our main contributions include:

- A smoothness-inducing regularizer is proposed to improve the existing training of learned optimizers. This term, representing the maximal distance of updates from the current state to the other in the neighborhood, is minimized to narrow the output gap for similar states.

- We evaluate our proposed regularization term on various classification problems and the learned optimizer outperforms hand-engineered methods even if transferring to tasks with different architectures and data.
- In addition, we also conduct experiments on few-shot learning based on a Meta-LSTM optimizer [RL17] and SIB [HMX20]. Results show that our smoothness-inducing regularizer consistently improves the accuracy on FSL benchmark datasets for 5-way few shot learning.

3.2 Related Work

Gradient-based optimization has drawn extensive attention due to its significance to deep learning. Various algorithms have been proposed to improve training of deep neural networks, including SGD [RM51], Adam [KB14], RMSProp [HSS12], and the like. On the other hand, a profound thought of updating the optimizee automatically rather than using hand-engineered algorithms has broken the routine and shown great potentials in improving specific problems. Early attempts can be dated back to 1990s when [CC90] leveraged recurrent neural networks to model adaptive optimization algorithms. The idea was further developed in [YHC01] where neural optimizers were trained to tackle fundamental convex optimization problems. Recently a seminal work of [ADG16] designed a learning-to-learn framework with an LSTM optimizer, which obtained better performance than traditional optimizers for training neural networks. Follow-up work in [LJL17] and [WMH17] have improved the generalization and scalability of learned optimizers. L2L has also been extended to various applications such as few-shot learning [RL17], zeroth-order optimization [RXR19] and adversarial training [XH20].

Our work is the first to investigate the smoothness of neural optimizers in the L2L

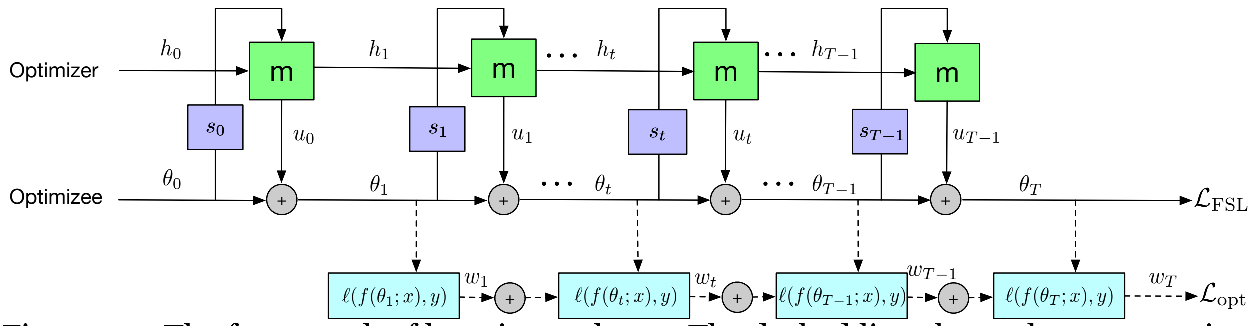


Figure 3.1: The framework of learning-to-learn. The dashed line shows the computation graph of the objective function $\mathcal{L}_{\text{opt}}$ for training the optimizer to learn a general update rule while the horizontal full line is the one for few-shot learning. Note that m is the neural optimizer parameterized by ϕ, and s_t is the state of the optimizee taking the form of $s_t = (\theta_t, \dots, \nabla_\theta \ell)^T$.

framework. It is related to the notion of adversarial robustness in classification models. As observed in [GSS14], neural network based models are vulnerable to malicious perturbations. In particular, for image classification the classifier would be fooled by adversarial examples to make a wrong prediction [GSS14], while for reinforcement learning the agent is likely to act differently under perturbed states [SLJ20]. Our learned optimizers might be affected by this issue as well. In other domains some algorithms have been proposed to mitigate the non-smooth property of neural networks such as adversarial training [MMS17], and SR^{2}L [SLJ20]. In this chapter, our method utilizes the idea of minimizing the worst-case loss to regularize training of neural optimizers towards smoothness. In contrast to previous algorithms targeted at classification, we design a specific regularizer to neural optimizers.

3.3 Background on the L2L Framework

In this section, we present the framework of L2O with neural optimizers for tackling problems of general optimization for classification and few-shot learning.

3.3.1 Optimization Process

As shown in Figure 3.1, like any traditional optimization methods, we can apply the learned optimizer as follows:

(a) At each time step t, feed a batch of training examples $\{(x, y)\}$ from the distribution $\mathcal{D}$ into the target classifier f parameterized by θ, and the state of the optimizee s_t can be described by several values such as the current parameter value, its gradient, or the exponentially weighted moving averages of the gradient.

(b) Given the current state s_t and the hidden state h_t, the neural optimizer m parameterized by ϕ accordingly outputs the increment of the parameter and the next hidden state by $u_t, h_{t+1} = m(s_t, h_t)$.

(c) The optimizer updates the parameter by $\theta_{t+1} = \theta_t + u_t$.

All above operations are coordinate-wise, which means the parameters are updated by a shared neural optimizer independently and maintain their individual hidden states.

The exploitation of the learned optimizer is straightforward but how can we train it? Following [ADG16], since parameters of the optimizee depend implicitly on the optimizer, which can be written as $\theta_t(\phi)$, the quality of the optimizer can be reflected by performance of the optimizee for some horizon T, leading to the objective function below to evaluate the optimizer:

$$\mathcal{L}_{\text{opt}}(\phi) = \mathbb{E}_{(x,y)\sim\mathcal{D}} \left[\sum_{t=1}^{T} w_t \ell(f(\theta_t(\phi); x), y) \right]. \tag{3.1}$$

$\ell(\cdot, \cdot)$ represents cross-entropy and w_t is the weight assigned for each time step.

3.3.2 Few-shot Learning

Apart from optimization, the superiority of learned optimizers is a natural fit for few-shot learning. Generally, FSL is a type of machine learning problems with only a limited number of labeled examples for a specific task [WYK19]. In this chapter, we mainly focus on FSL targeted at image classification, specifically N-way-K-shot classification. We deal with a group of meta-sets $\mathcal{D}_{\text{meta}}$ in this task. Each element in $\mathcal{D}_{\text{meta}}$ is a meta-set $D = (D_{\text{train}}, D_{\text{test}})$, where D_{train} is composed of K images for each of the N classes (thus $K \cdot N$ images in total) and D_{test} contains a number of examples for evaluation. The goal is to find an optimization strategy that trains a classifier leveraging D_{train} with only a few labeled examples to achieve good learning performance on D_{test}. All meta-sets are further divided into three separate sets: meta-training set $\mathcal{D}_{\text{meta-train}}$, meta-validation set $\mathcal{D}_{\text{meta-val}}$, and meta-testing set $\mathcal{D}_{\text{meta-test}}$. More concretely, $\mathcal{D}_{\text{meta-train}}$ is utilized to learn an optimizer and $\mathcal{D}_{\text{meta-val}}$ is for hyperparameter optimization. After the optimizer is determined, we conduct evaluation on $\mathcal{D}_{\text{meta-test}}$: we first update the classifier with the learned optimizer on the training-set in $\mathcal{D}_{\text{meta-test}}$; then we use the average accuracy on the test set in $\mathcal{D}_{\text{meta-test}}$ to evaluate the performance of the optimizer.

The N-way-K-shot classification problem can be simply incorporated into the L2L framework, where the optimization strategy is modeled by the learned optimizer. As we aim at training a classifier with high average performance on the testing set, instead of harnessing the whole optimization trajectory, the objective can be modified to attach attention only to the final testing loss:

$$\mathcal{L}_{\text{FSL}} = \mathbb{E}_{D \sim \mathcal{D}_{\text{meta}}} \mathbb{E}_{(x,y) \sim D_{\text{test}}} \left[\ell \left(f(\theta_T(\phi); x), y \right) \right], \tag{3.2}$$

where θ_T is updated by a procedure described in Section 3.3.1 under examples from D_{train}.

3.4 Method

3.4.1 Motivation

Despite great potentials of neural optimizers in improving traditional optimization and few-shot learning, there exists a significant problem impeding the development of L2L. In contrast to classical hand-engineered optimization methods, those learned ones cannot guarantee a smooth update of parameters, i.e., producing similar outputs for similar states, where by state we mean the gradient or parameters of the optimizee. In Figure 3.2, we demonstrate the non-smoothness of the learned optimizer explicitly. This is a typical phenomenon in various neural-network-based algorithms such as image classification and reinforcement learning. [Ham74] and [SLJ20] have pointed out advantages of smoothness of a function to mitigate overfitting, improve sample efficiency and stabilize the overall training procedure. Thus, enforcing the smoothness of the learned optimier can be crucial to improve its performance and stability.

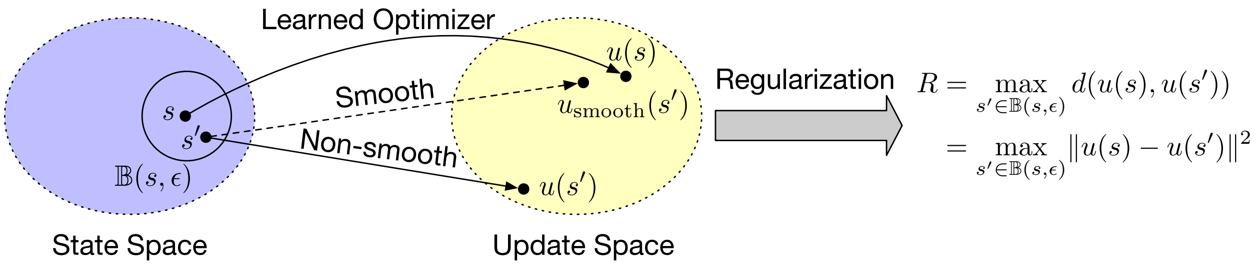


Figure 3.2: An illustration of non-smoothness.

3.4.2 Smoothness Regularization

Some techniques, such as ℓ_2 regularization and gradient clipping, have been developed and utilized in training neural optimizers to enforce smoothness but they are shown insufficient to reduce non-smoothness [ADG16, LJL17, MMN18]. We propose to robustify the learned optimizer through a smoothness-inducing training procedure where a regularization term is introduced to narrow the gap between outputs of two similar input states. It is also known as an effective method to constrain the Lipschitz constant of neural networks to boost smoothness. To describe our method clearly, we first denote two states before updating the optimizee at the time step $t+1$ by s_t and s'_t. Note that s_t and s'_t are distinct but similar states, i.e., $s'_t \in \mathbb{B}(s_t, \epsilon)$, where $\mathbb{B}(s_t, \epsilon)$ represents the neighborhood of s within the ϵ-radius ball in a certain norm and ϵ is perturbation strength. In this chapter, we just use ℓ_∞ norm without loss of generality. Fix the hidden state h_t, then u_t and u'_t, which are the corresponding parameter increments of s_t and s'_t, can be written as functions of the state $u(s_t)$ and $u(s'_t)$ explicitly.

An ideal optimizer is expected to produce similar updates and thus to attain such an optimizer, our goal is to minimize the discrepancy $d(\cdot, \cdot)$ between $u(s_t)$ and $u(s'_t)$. Inspired by adversarial training [MMS17], it is intuitive to find the gap under the worst-case as the targeted difference that takes the form of $\max d(u(s_t), u(s'_t))$ and minimize this term directly. However, the optimizer that takes the state of optimizee as input and the update as output, is different from the classifier whose input is an image and output is a vector of softmax logits. There is no classification for the optimizer so distance metrics such as cross-entropy and KL-divergence are not applicable to our problem. Since the output is a scalar value, we measure the distance with the squared difference and the gap at the time step $t+1$ becomes

$$R_{t+1}(\phi) = \max_{s'_t \in \mathbb{B}(s_t, \epsilon)} d(u(s_t), u(s'_t)) = \max_{s'_t \in \mathbb{B}(s_t, \epsilon)} \|u(s_t) - u(s'_t)\|^2. \tag{3.3}$$

After the regularization term is determined, we can add it to the original objective function of L2L as a regularizer. For each time step, the objective becomes

$$\ell_t(\phi) = \ell(f(\theta_t(\phi); x), y) + \lambda R_t(\phi), \tag{3.4}$$

where λ is the regularization coefficient and the optimizer's parameters ϕ are updated by

$$\min_{\phi} \mathcal{L}_{\text{opt}}(\phi) = \mathbb{E}_{(x,y)\sim\mathcal{D}} \left[\sum_{t=1}^{T} w_t \ell_t(\phi) \right]. \tag{3.5}$$

In few-shot learning, we store regularization terms during training with D_{train} and add the accumulated items to Eq. 3.2, leading to the training objective as

$$\begin{aligned} \min_{\phi} \mathcal{L}_{\text{FSL}}(\phi) =& \mathbb{E}_{D\sim\mathcal{D}_{\text{meta}}} \Bigg[\mathbb{E}_{(x,y)\sim D_{\text{test}}} \ell\left(f(\theta_T(\phi); x), y\right) \\ &+ \mathbb{E}_{(x,y)\sim D_{\text{train}}} \lambda \sum_{t=1}^{T} R_t(\phi) \Bigg]. \end{aligned} \tag{3.6}$$

Note that our proposed regularizer can be applied to any neural optimizer-based algorithms in meta-learning, such as methods in [ADG16, LJL17, HMX20].

3.4.3 Training the Optimizer

The key component for training the optimizer is the calculation of the regularization term in Eq. 3.3. As stated in [SLJ20], in practice we can effectively approximate the solution of the inner maximization by a fixed number of Projected Gradient Descent (PGD) steps:

$$s' = \Pi_{\mathbb{B}(s,\epsilon)}(\eta \, \mathbf{sign}(\nabla_{s'} d(u(s), u(s'))) + s'), \tag{3.7}$$

where Π is the projection operator to control the state located within the given radius of the neighborhood. Note that we use truncated Backpropagation Through Time (BPTT) to update our RNN optimizer in case of a too long horizon. For the predefined weight in Eq. 3.5, to make best use of the optimization trajectory and concentrate more on the loss of the last step at the same time [CRK20], we adopt a linearly-increasing schedule that $w_t = t \bmod T$ where T is the number of step in each truncation. We present the whole training procedure in Algorithm 2.

Specifically, since our aim is to find a perturbed state in the neighborhood of the original state, we can obtain it as follows: a) Starting from the original state s, we add an imperceptible noise to initialize s'; b) Compute the current value of $d(u(s), u(s'))$, backpropagate the gradient back to s' to calculate $\nabla_{s'} d(u(s), u(s'))$, and then adjust the desired state by a small step η in the direction, i.e., $\text{sign}(\nabla_{s'} d(u(s), u(s')))$, that maximizes the difference; (c) Run K steps in Eq. 3.7 to approximate the regularization term in Eq. 3.3.

Algorithm 2 L2L with Smooth-inducing regularization

```
1:  Input: training data $\{(x, y)\}$, step sizes $\eta_1$ and $\eta_2$, inner steps $K$, total steps $T_{\text{total}}$,
    truncated steps $T$, classifier parameterized by $\theta$, optimizer parameterized by $\phi$
2:  repeat
3:      Initialize $\theta$ randomly, reset RNN hidden state
4:      $\mathcal{L} \leftarrow 0$
5:      for $t = 0, \ldots, T_{\text{total}} - 1$ do
6:          Sample a batch of data $(x, y)$, feed it to the classifier, obtain state $s_t$
7:          Update $\theta$ as demonstrated in Section 3.1
8:          $s'_t \leftarrow s_t + 0.05 * \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
9:          for $k = 1, \ldots, K$ do                                    ▷ Find the perturbed state
10:             $s'_t \leftarrow \Pi_{\mathbb{B}(s_t, \epsilon)}(\eta_1 \, \text{sign}(\nabla_{s'_t} d(u(s_t), u(s'_t))) + s'_t)$
11:         end for
12:         $R_{t+1} \leftarrow \|u(s_t) - u(s'_t)\|^2$                    ▷ Regularization term
13:         $\mathcal{L} \leftarrow \mathcal{L} + w_{t+1}\ell_{t+1}$             ▷ $\ell_{t+1}$ is computed by Eq. 3.4
14:         if $t \bmod T - 1 == 0$ then
15:             Update $\phi$ by $\mathcal{L}$ using Adam with $\eta_2$
16:             $\mathcal{L} \leftarrow 0$
17:         end if
18:     end for
19: until converged
```

3.5 Experimental Results

We implemented comprehensive experiments for evaluation of our proposed regularizer. Detailed results are presented in Section 3.5.1 for neural network training and Section 3.5.2 for few-shot learning. All algorithms are implemented in PyTorch-1.2.0 with one NVIDIA

1080Ti GPU.

3.5.1 L2L for neural network training

In this part, we evaluate our method through the task of learning the general update rule for training neural networks. The performance of different optimization algorithms is primarily displayed in learning curves of both training and testing loss, as suggested in previous studies [ADG16, LJL17, MMN18, CZJ20]. As loss and accuracy do not necessarily correlate, we also report accuracy curves for thoroughness.

3.5.1.1 Experiment settings

We consider image classification on MNIST [LeC98] and CIFAR10 [KNH14]. Our learned optimizer with regularization is compared with hand-designed methods including SGD, SGD with momentum (SGDM), Adam, AMSGrad, and RMSProp, as well as neural optimizers DMOptimizer [ADG16] and SimpleOptimizer [CRK20]. For hand-designed optimizers, we tune the learning rate with grid search over a logarithmically spaced range $[10^{-4}, 1]$ and report the performance with the best hyperparameters. As to baseline neural optimizers, we use recommended hyperparameters, optimizer structures, and state definitions in [ADG16] and [CRK20] respectively. We have tried different hyperparameters for baselines and found that recommended ones are the best in our experiments. Our smoothed optimizers adopt original settings, except for two extra hyperparemeters for training, the perturbation strength ϵ and the regularization coefficient λ. In particular, ϵ and λ in our method are also determined by a logarithmic grid search with the range $\epsilon \in [10^{-2}, 10]$ and $\lambda \in [10^{-1}, 10^{2}]$. Neural optimizers are learned with Adam of the learning rate 10^{-4} with the number of total steps $T_{\text{total}} = 200$ and truncated steps $T = 20$. Note that for all neural optimizers we only tune the hyperparameters during training and directly apply them to a new optimization problem, while for hand-engineered algorithms, the learning rate is always tuned for the specific task. Experiments for each task are conducted five times with different seeds and the batch size is 128.

3.5.1.2 Compatibility of the proposed regularizer

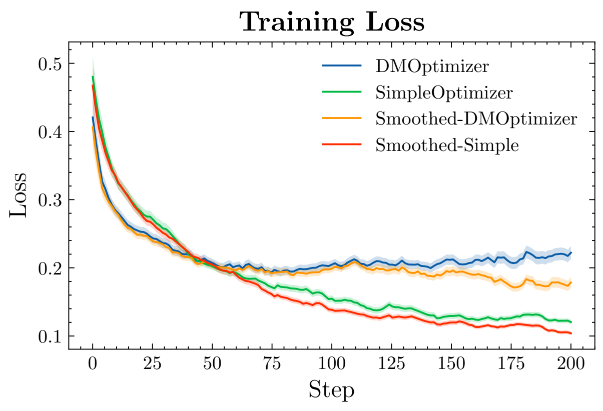
We first conduct an experiment to demonstrate that the proposed regularization term can be combined with various L2L structures. We demonstrate the performance of learned optimizers including training loss and testing loss for training a 2-layer MLP on MNIST. As can be seen in Figure 3.3a and 3.3c, two L2L architectures, DMOptimizer [ADG16] and SimpleOptimizer [CRK20], are compared. With the regularizer, the smoothed version of both optimizers make an improvement in the final training and testing loss, and obtain a faster convergence rate at the same time. In addition, since SimpleOptimizer performs better than DMOptimizer, which is consistent with the observation in [CRK20], we will apply it as our base optimizer in the following experiments.

3.5.1.3 Training on MNIST

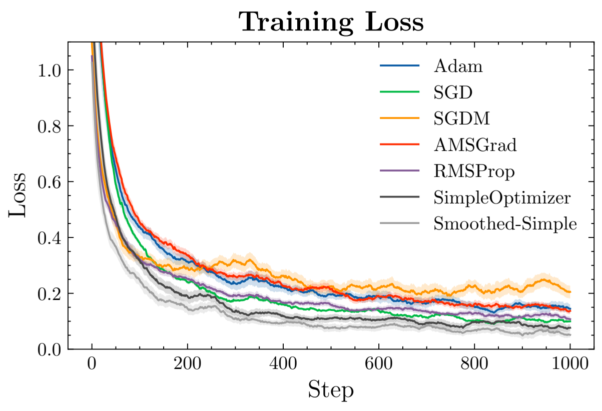
In this part, we train the neural optimizers for a 200-step optimization of LeNet on MNIST. Following [ADG16], we make an evaluation on a longer trajectory with 1000 steps. Although the neural optimizer is only trained within 200 steps, it is capable of updating the optimizee until 1000 steps with faster convergence rate and better final loss consistently, as shown in Figure 3.3b and 3.3d.

3.5.1.4 Training on CIFAR-10

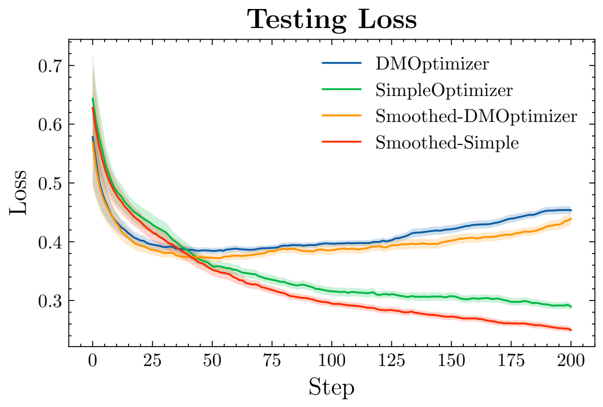
It is insufficient to merely test different optimizers on MNIST, whose size is relatively small. Therefore, we add to the difficulty of the targeted task and focus on image classification on CIFAR-10. The classifier of interest is a 3-layer convolutional neural network with 32 units per layer and the learned optimizer is employed to update the optimizee for 10000



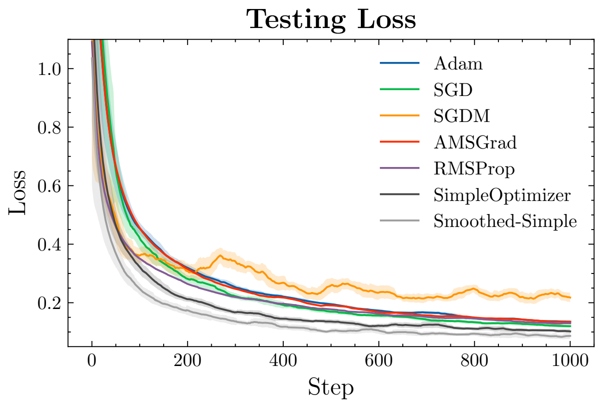
(a)



(b)



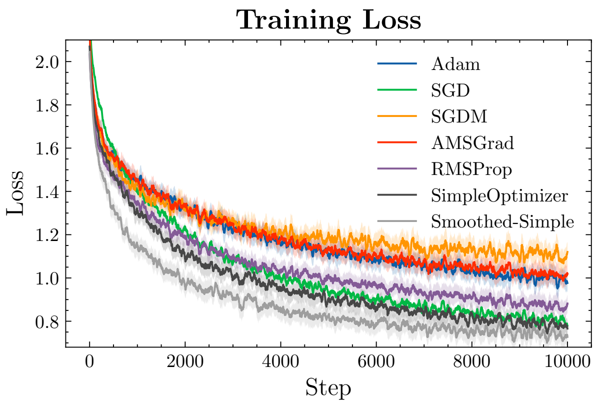
(c)



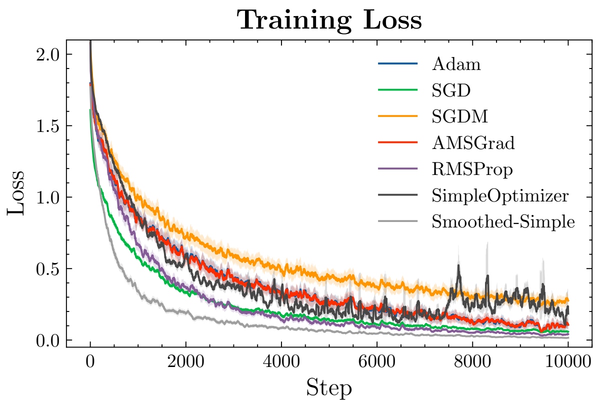
(d)

Figure 3.3: Learning curves on MNIST. Training loss is shown in the first row and testing loss is in the second row. (a) and (c) are results of two neural optimizer structures to show the compatibility of our proposed regularizer; (b) and (d) demonstrate performance of different optimizers for training LeNet of 1000 steps.

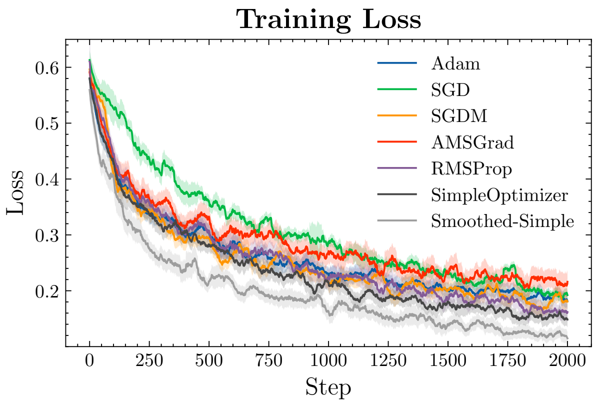
steps. It should be pointed out that the neural optimizer is still trained within 200 steps and the optimization step for evaluation is 50 times larger than what it has explored during training. Figure 3.4a and 3.4e demonstrate its great generalization ability: the smooth version of the learned optimizer can converge faster and better than hand-engineered algorithms such as SGD and Adam, even though it only observes the optimization trajectory in the limited steps at the very beginning. Our smoothed variant also outperforms the original learned optimizer.



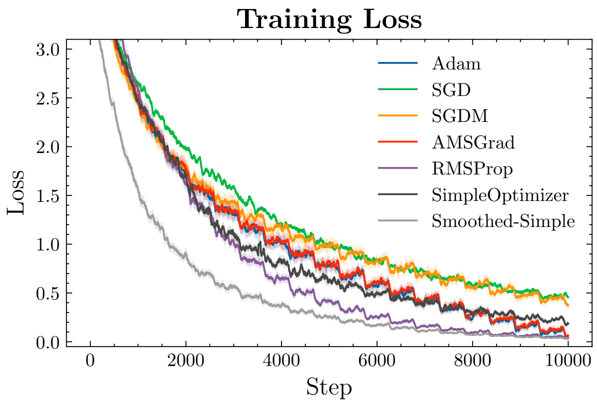
(a)



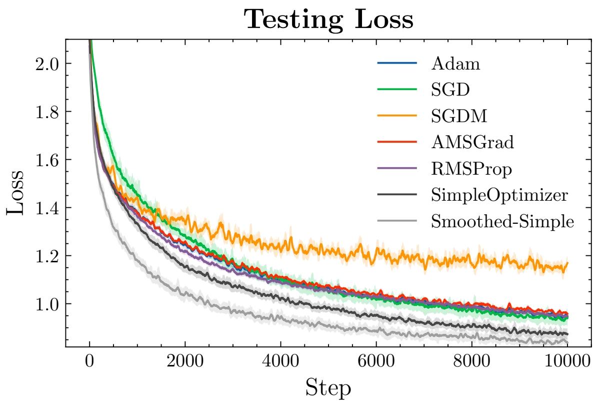
(b)



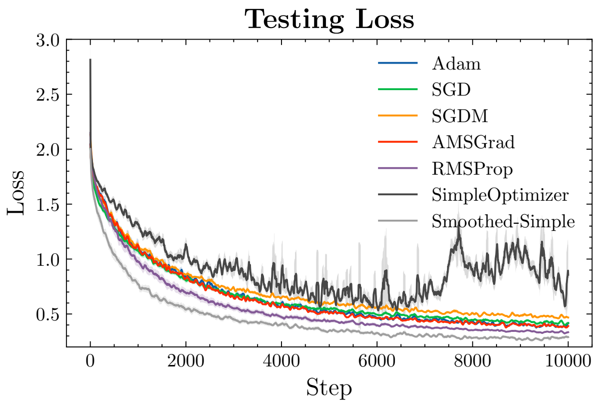
(c)



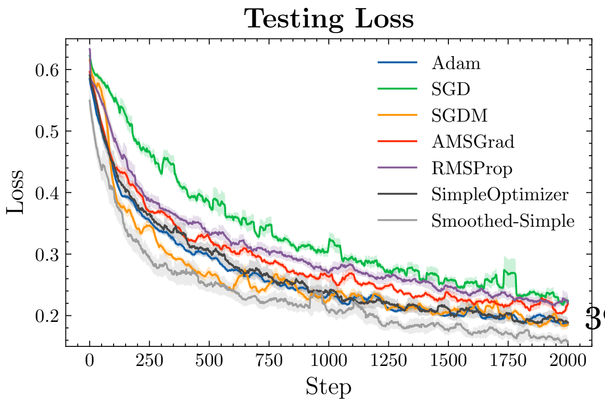
(d)

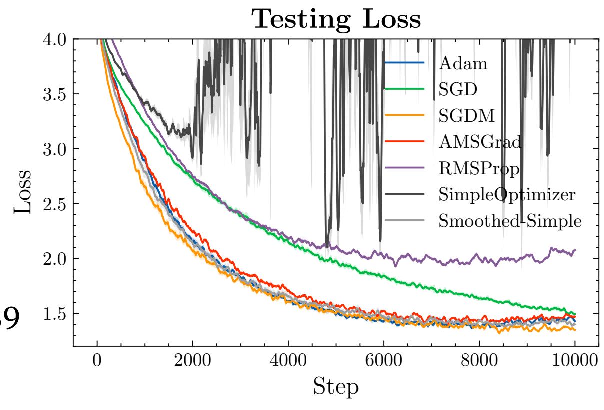


(e)



(f)





(g)

(h)

Figure 3.4: Learning curves on CIFAR-10 and CIFAR-100.

3.5.1.5 Transferrablity to different settings

After obtaining a neural optimizer trained on CIFAR-10 with a 3-layer CNN, we evaluate its transferrability in multiple aspects. Specifically, we first transfer the optimizer to training another network structure, ResNet-18 [HZR16] for 10000 steps. In Figure 3.4b and 3.4f, Smoothed-Simple without finetuning can still beat all hand-designed methods. Besides, it should be emphasized that SimpleOptimizer oscillates violently at the end of training and loses its advantages over traditional methods for this transferring task, while the performance of our Smoothed-Simple is consistent and robust.

Moreover, since we have shown that our neural optimizer can generalize to longer training horizon and different network architectures on optimizees with the same dataset, this naturally leads to the following question: can our neural optimizer learn the intrinsic update rule so that it can generalize to unseen data? To answer this question, we modify the experimental setting to evaluate our proposed optimizer with respect to optimization on unseen data. We split the original CIFAR-10 dataset into three different sets: a training set containing 6 classes, a validation set and a testing set with 2 classes respectively. When training the optimizer, we sample 2 classes from training set and minimize the objective function for a binary classification problem. Images in the validation set are exploited to select the optimizer which achieves best final testing loss in the 200-step optimization. A comparison of learning curves among our smoothed optimizer, SimpleOptimizer, and the rest hand-designed methods for updating the classifier on two unseen classes is shown in Figure 3.4c and 3.4g. We can observe that the smoothed optimizer learns much more quickly and better than other algorithms.

Finally, we test the performance of our optimizer in the most difficult setting: training a ResNet-18 on CIFAR-100, where both the network structure and the dataset are modified. It can be observed in Figure 3.4d and 3.4h that Smoothed-Simple has comparable performance with fine-tuned hand-engineered optimizers in terms of testing loss. On the contrary, SimpleOptimizer is incapable of dealing with this scenario with a large dataset and a complicated network.

3.5.1.6 Comparison with other regularization methods.

As the proposed method serves as a novel regularization term, for the completeness of experiments, we compare our adversarial regularization with three representative techniques: ℓ_2 regularization [Sch15], orthogonal regularization [BCW18], and spectral normalization [MKK18]. In detail, we train a 3-layer CNN on CIFAR-10 for 10000 steps, following the setting in Section 3.5.1. Results of training and testing loss can be found in Figure 3.5, and we also report the test accuracy for reference in Table 3.1. It can be observed that orthogonal regularization even worsens the learned optimizer, which cannot provide meaningful updates and only leads to a random-guess classifier with 10.00% on test accuracy. While ℓ_2 regularization and spectral normalization can improve the SimpleOptimizer to 69.69% and 69.35% respectively, our proposed Smoothed-Simple still outperforms them significantly with 72.50%. This experiment shows that the smoothness-inducing regularization obtained by PGD can achieve performance gain against other popular regularization techniques and is more suitable in training a neural optimizer.

Table 3.1: Test accuracy of different regularizers.

Regularizer	**Test accuracy**
SimpleOptimizer	69.02 ± 0.58%
Simple-Smoothed	**72.50** ± **0.49**%
ℓ_2 Regularization	69.69 ± 0.56%
Orthogonal Regularization	10.00 ± 0.04%
Spectral Normalization	69.35 ± 1.23%

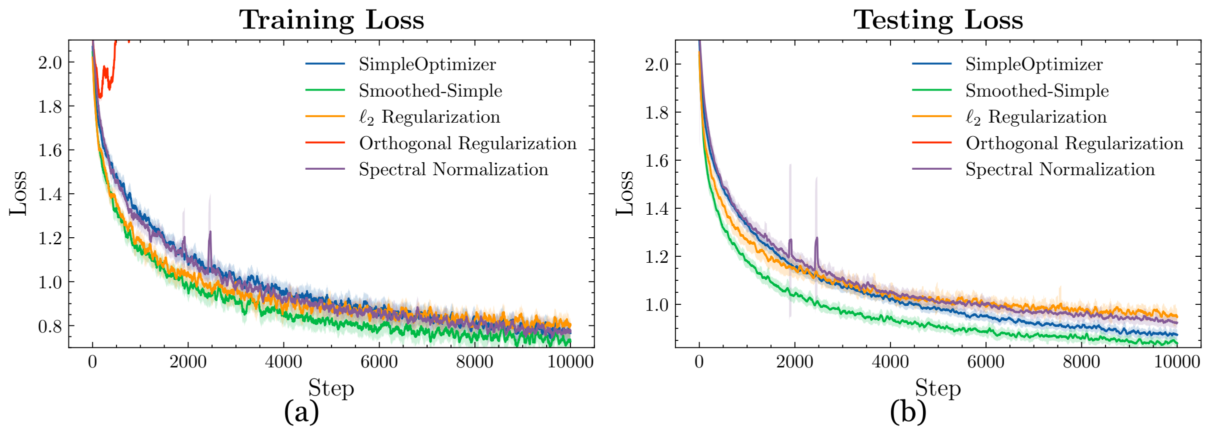


Figure 3.5: Learning curves of neural optimizers with different regularization methods.

3.5.1.7 Smoothness with Perturbation

In this section, we analyze the optimizer's smoothness property with perturbation. Detailedly, we sample 1000 points from $\mathcal{N}(0, 0.1)$ to form a set of perturbed states around 0. Then these states are fed into the simple and smoothed optimizer respectively, and corresponding outputs are shown in Figure 3.6 (x-axis in (a) is the state number while (b) sorts the specific state values.) Around the zero state with zero gradient, the update with a small magnitude is expected for an ideal optimizer. We can see that the smooth version can produce much more stable updates around zero state, while Simple-Optimizer suffers from non-smoothness.

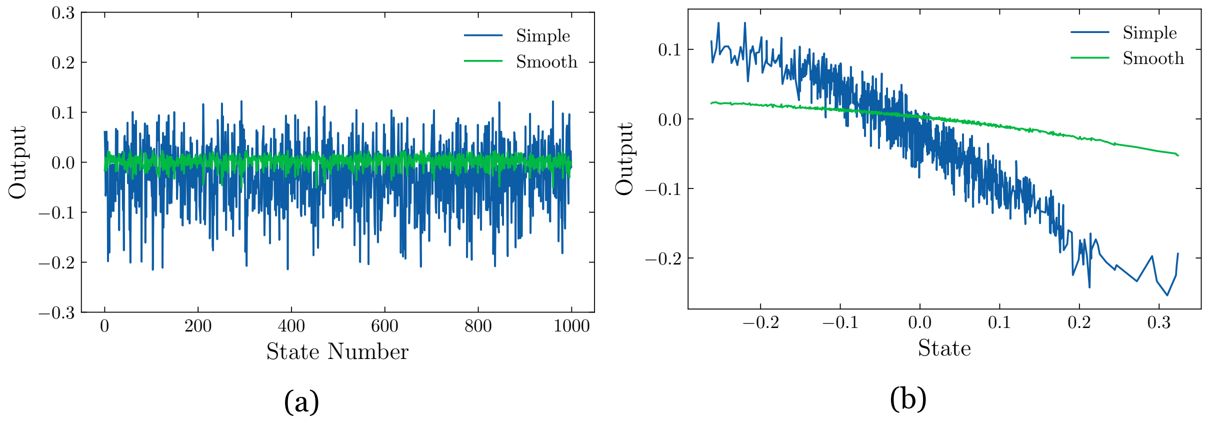


Figure 3.6: A comparison of smoothness between simple and smoothed optimizer.

3.5.1.8 Additional Evaluation

Besides the metric of loss, we explore classification accuracy which is another important performance indicator, to show advantages of our smoothed neural optimizer. In Figure 3.7, we present curves of training and testing accuracy, for MNIST with LeNet and CIFAR10 with the 3-layer CNN. We can observe that the relative ranks of all optimizers do not change if the evaluation metric is switched to accuracy, and our smoothed optimizer still outperforms other algorithms with best final training and testing accuracy as well as convergence rate.

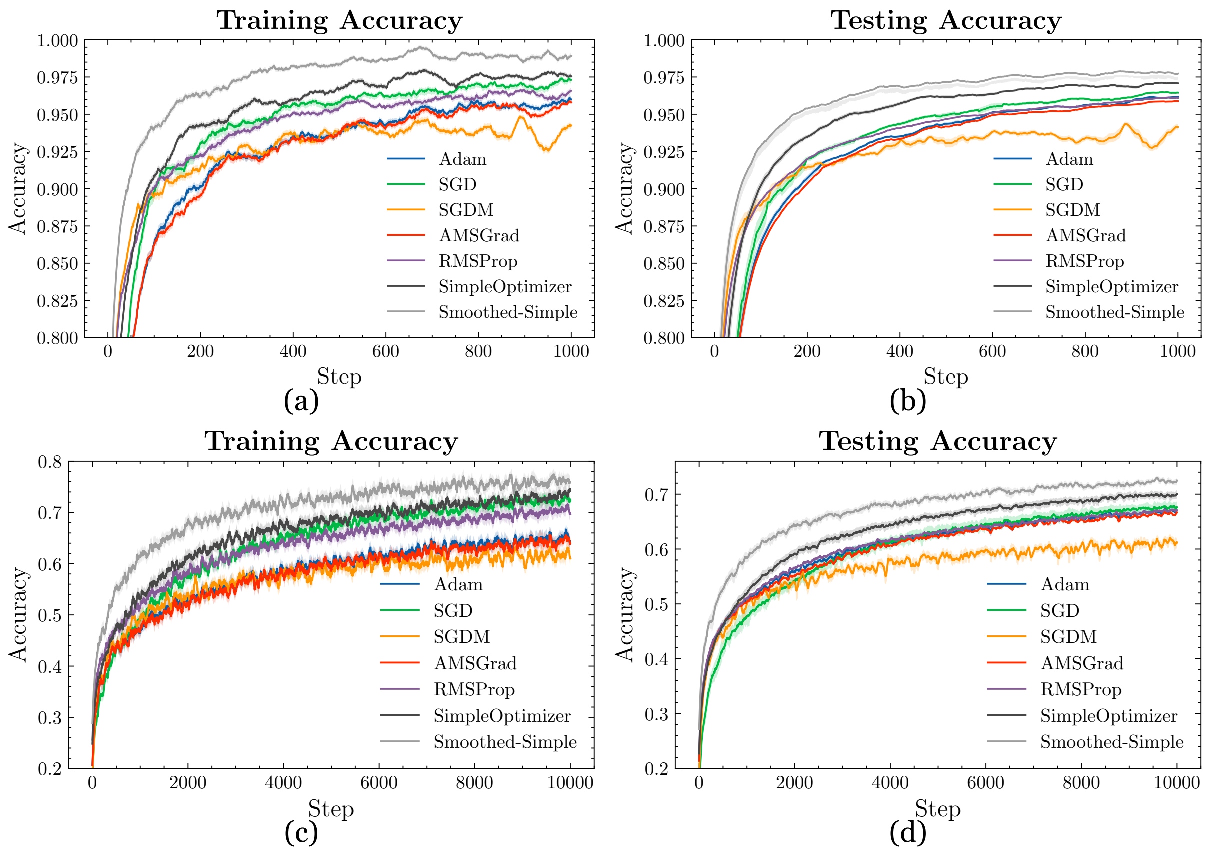


Figure 3.7: Learning curves of different optimizers in training and testing accuracy. (a)-(b) for MNIST with LeNet and (c)-(d) for CIFAR-10 with a 3-layer CNN.

3.5.2 Few-Shot Learning with LSTM

Apart from improving the training procedures, L2L can be applied to few-shot learning as well. Therefore, in this part we primarily explore the effectiveness of our smoothed neural optimizer in FSL, in particular, N-way-K-shot learning. We consider 5-way-1-shot and 5-way-5-shot problems on two benchmark datasets, miniImageNet [VBL16] and tiered-ImageNet [RTR18]. The base structure we utilize is Meta-LSTM, proposed in [RL17] to train an LSTM-based meta learner to learn the optimization rule in the few-shot regime. We compare it with our smoothed version. We keep all hyperparameters the same as reported in [RL17] and only tune ϵ and λ in a manner introduced in Section 3.5.1.1. Statistical results of 5 experiments with different random seeds are reported in Table 3.2. Our smoothed Meta-LSTM attains 2% percents improvement over all scenarios against the baseline. It should be emphasized that the performance boost is purely credited to the regularizer since we apply our regularization term to the exactly same structure as Meta-LSTM. Since the official code for Meta-LSTM is written in lua and is out-of-date, we use the latest PyTorch implementation in [Don19]. Thus, our results might lead to inconsistency with the original paper but do not affect the conclusion.

In addition, we integrate our proposed regularizer into one of the most recent methods involving a neural optimizer, SIB [HMX20] on miniImageNet and CIFAR-FS. Results are presented in Table 3.3 and with regularization, SIB performs consistently better especially for 5-shot tasks.

Table 3.2: Average accuracy of 5-way few shot learning on miniImageNet and tieredIamgeNet.

Model	miniImageNet		tieredImageNet	
	1-shot	5-shot	1-shot	5-shot
Meta-LSTM	38.20 ± 0.73%	56.56 ± 0.65%	36.43 ± 0.65%	53.45 ± 0.61%
Smoothed Meta-LSTM	**40.42 ± 0.68%**	**58.90 ± 0.61%**	**36.74 ± 0.76%**	**55.14 ± 0.60%**

Table 3.3: Average accuracy of 5-way few shot learning problems on miniImageNet and CIFAR-FS.

Model	Backbone	miniImageNet		CIFAR-FS	
		1-shot	5-shot	1-shot	5-shot
SIB($\eta = 1e^{-3}$, $K = 3$)	WRN-28-10	69.60 ± 0.60%	78.90 ± 0.40%	78.40 ± 0.60%	85.30 ± 0.40%
Smoothed SIB	WRN-28-10	**70.03** ± **0.51%**	**80.85** ± **0.32%**	**79.24** ± **0.40%**	**86.13** ± **0.42%**

3.6 Conclusion and Discussion

This work first investigates the smoothness of learned optimizers and leverage it to achieve performance improvement. Specifically, we propose a regularization term for neural optimizers to enforce similar parameter updates given similar input states. Extensive experiments show that the regularizer can be combined with different L2L structures involving neural optimizers, and verify its effectiveness of consistently improving current algorithms for various tasks in classification and few-shot learning. Despite promising results, currently the learned optimizer is constrained to a group of problems with a moderate number of optimization steps and cannot replace hand-crafted ones in such settings. Training a powerful optimizer that can generalize to longer horizon still remains a challenge and can be a potential future direction. Besides, how to design a neural optimizer to deal with language tasks with RNNs or even more complex models like Transformers is also an interesting problem to be explored.

CHAPTER 4

Learning to Schedule Learning Rate with Graph Neural Networks

Even after regularization, learned optimizers are likely to cause divergence when the training horizon is very long. Therefore, we study to make only part of the updating procedure automatic, in particular, learning rate scheduling in this chapter. We design a neural scheduluer based on reinforcement learning and demonstrate it can outperform other hand-crafted ones in representative downstream tasks.

4.1 Introduction

Stochastic optimization [RM51] has achieved great success in training deep learning models, due to the tremendous data size and non-convexity of neural networks [XLL20]. However, in contrast to classical and hyperparameter-free optimizers, performance of widely used stochastic algorithms such as SGD and Adam significantly depends on the choice of hyperparameters including batch size, learning rate and momentum coefficients [XLL20, SMV20, CLD21, CSN19].

Learning rate scheduling is one of the most important factors which stochastic optimizers are very sensitive to [CLD21, XDK19]. A fixed and static learning rate throughout the whole training is insufficient to attain an optimal model. Thus, various scheduling algorithms [GKX18] including polynomial decay, cosine decay and warmup have been developed and each of them is designed in a distinct form. In practice, models trained with an appropriate scheduling mechanism can outperform those with a constant learning rate, in terms of convergence rate as well as the final performance [SHR13, GKX18, LH16].

As the learning rate scheduling is of vital importance to the model performance, how to tune it for a given stochastic optimizer effectively and automatically has been investigated intensively. Current approaches tackle this problem by assuming a particular learning rate scheduling rule (e.g., polynomial or cosine decay, warmup and restart) based on empirical studies and domain knowledge [XQW17, GKX18], and apply traditional hyperparameter optimization methods [LJD17, SLA12, BB12] to tune the parameters of the rule. However, searching within those pre-defined principles is restricted since the best learning rate scheduling for a particular training problem may not exactly follow any existing rule.

To overcome this restriction, finding an optimal scheduler can be viewed as a learning-to-learn problem [ADG16, CZJ20], where empirical scheduling principles are replaced by a data-driven neural network alternative. For instance, attempts such as [XDK19] and [XQW17] model the learning rate scheduling as a sequential decision process where the learning rate is modified in every decision step, and train a policy network with reinforcement learning algorithms. However, these methods sacrifice rich state representations in order to maintain generalization. This implies that problem-dependent information, such as the architecture of neural networks to be optimized, and the state of intermediate neurons is not fully exploited. Therefore, state-of-the-art algorithms such [CZJ20] only represent the current state by the statistics of final layer neurons. A straightforward way to leverage such information is to concatenate states in every layer as a super vector. However, it would lead to the explosion of the input dimension and fail to work in models with different sizes. Therefore it is nontrivial to obtain a representation of training state that is both **informative** (being able to encode rich information of architecture and intermediate neurons) and **generalizable** (can be applied to different architectures).

In this work, we propose a Graph Network-based Scheduler (GNS) to overcome the above-mentioned challenges. Different from previous attempts, our GNS makes full use of the problem structure: since deep neural networks can be represented as directed graphs intrinsically [ZRU18], we leverage graph networks to encode states into latent features with richer information and such a design allows GNS to be easily transferred to different network architectures at the same time. Moreover, an action definition for stabling training and an efficient reward signal collection procedure are designed for GNS.

Our main contributions are summarized as following:

- We develop the Graph Network-based Scheduler (GNS) to search for the best learning rate scheduling rule by representing the target network as a directed graph. Our dynamic learning rate scheduler can thoroughly capture the intermediate layer information in both image classification and language understanding tasks. Also, an efficient reward collection procedure is designed to speedup our method.

- We apply the proposed GNS to multiple training tasks. In particular, GNS achieves average scores of 85.2 and 87.8 on GLUE with RoBERTa-base and -large respectively and outperforms 84.7 and 86.9 from the best existing learning rate schedule. Unlike previous learning-to-learn methods that mostly are applied to toy examples such as CIFAR-10, we are the first to apply an automatic scheduler to challenging tasks like RoBERTa training and improve the performance of those realistic large models.

- We demonstrate that the GNS framework is able to generalize to problems of different datasets (from CIFAR10 to CIFAR100) and model structures (from ResNet to VGG, and from RoBERTa-base to RoBERTa-large).

4.2 Related Work

Learning rate scheduling. This topic has been an open problem in stochastic optimization. In [ALA98] and [BCR17], learning rate was regarded as a trainable parameter and modified using its gradient with respect to the objective function. [Sut92] and

[SYA06] considered a meta-descent algorithm to adjust local learning rate with a meta one. In the era of deep learning, a number of adaptive optimization algorithms have been proposed, such as RMSProp [TH12], AdaGrad [DHS11], and Adam [KB14]. They conduct parameter-wise adaptation to learning rate based on heuristics of the geometry of the traversed objective. For instance, RMSProp normalizes the learning rate with the magnitude of recent gradients while AdaGrad generates a large learning rate for infrequently-updated parameters. Complimentary to these parameter-wise adaptations, some predefined learning rate schedulers are also observed beneficial to training performance [YLW19, GKK19, SZL13, SHR13]. Mechanisms like polynomial [MS19] and cosine decay [LH16] together with a warmup process [GKX18] improve the convergence rate and final performance of deep learning models empirically. However, they still hold additional hyperparameters to be tuned. At the same time, these parametric schedules have limited flexibility and may not be optimized for the training dynamics of different high dimensional and non-convex optimization problems [XQW17, XDK19].

Reinforcement learning. The goal of reinforcement learning (RL) is to find an agent that produces an optimal policy to maximize a cumulative reward [SB18]. Recently, deep reinforcement learning with neural networks as approximation functions has shown great potentials in many applications, such as Atari games [GLS16, HVP18, SWD17], Go [SHM16, SSS17], and training deep learning models [ZL16, CZM18]. There are also several attempts to apply RL to learning rate controlling. For example, [DTN16] proposed to improve the stability of training w.r.t. the initial learning rate via reinforcement learning. [LM16], [XQW17] and [XDK19] leveraged RL agents to adjust the learning rate to reach a final performance whereas their selection of state features, reward and action leads to an inefficient training process, and is difficult to scale up to large models such as BERT [DCL18] and RoBERTa [LOG19].

Graph Neural Networks. Graph Neural Networks (GNN) is a type of neural networks that operate on graph data and structure their computations accordingly [BHB18]. A graph network, or a graph-to-graph module, can be interpreted as a mapping from an input graph to an output graph with the same structure but potentially different node, edge, and graph-level representations [BHB18, SGP20]. The most important component in GNN, Message Passing Neural Networks (MPNN) [GSR17], have been shown effective in learning dynamics for graph-structured settings. Generally, MPNN utilizes the graph structure to propagate information between nodes via edges, and extract latent features for downstream tasks. Variations such as GCN [KW16] and Interaction Networks [BPL16] have previously demonstrated promising results on simulation tasks in the field of physics [BPL16, SGP20] and chemistry [GSR17, YLY18].

4.3 Reinforcement Learning for Learning Rate Scheduling

In this section, we formulate learning rate scheduling as an RL problem, with the goal to find an optimal control policy to maximize the accumulated reward in the current environment which is defined by the target training problem.

4.3.1 Reinforcement Learning

Reinforcement learning aims at solving a group of problems, known as Markov Decision Process (MDP) with an agent choosing optimal actions [SB18]. Typically, the MDP problem is formulated by a tuple $M = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$. Given the state set $\mathcal{S}$ and action set $\mathcal{A}$, the reward function $\mathcal{R}$ is a mapping of $\mathcal{S} \times \mathcal{A} \to \mathbb{R}$. The discounted return is denoted by $R_t = \sum_{l=0}^{\infty} \gamma^l r_{t+l}$ where γ is a discount factor for future rewards. $\mathcal{P}$ is the state transition function, which specifies the probability $p(s_{t+1}|s_t, a_t)$ and is usually determined by the environment itself. To encourage more exploration, we focus on a stochastic policy $\pi(a|s)$, which maps states to action probabilities.

4.3.2 Learning rate scheduling as a MDP

For most problems involved in training deep neural networks, a standard approach for the objective function minimization is stochastic gradient descent on a certain form [RM51, KB14, DHS11], which sequentially updates the parameters using gradients by $w_{k+1} = w_k - \alpha_k \cdot g_k$. Specifically, w_k represents network parameters to be trained and α_k is the global learning rate. g_k is defined by the optimizer (e.g., g_k can be the SGD or Adam direction). For learning rate scheduling, we primarily consider generating a sequence for the global learning rate α_k.

We can formulate learning rate scheduling into a sequential decision process. In each environment of a particular optimization task, states can be described by dynamics of the target network such as a combination of weight and gradient values [XDK19, DTN16]. Actions can be any operation that might change the learning rate. As to rewards, since the ultimate goal is to achieve higher final performance, metrics like loss or accuracy can serve as a signal to guide the agent training. Besides, as training a deep learning model usually requires a large number of iterations, rather than a single optimization step, we execute an action every K steps to avoid instability from too frequent learning rate modifications. By training such an agent modeled as a policy network, we can obtain a learning rate scheduler to adjust learning rate dynamically.

4.4 Method

4.4.1 State Encoding with Graph Neural Networks

The selection of features to represent the environment is of vital significance. Previous work such as [XQW17] and [XDK19] aims to learn a generalized agent that can be applicable to any network architecture, so they were only able to utilize statistics of the final layer to describe the state. It might work for small scale models, but when dealing with large ones like RoBERTa [LOG19], solely using information of the last layer is insufficient to describe the environment.

A straightforward way to obtain a more thorough representation is to utilize dynamics of all layers in a neural network and concatenate them into one super vector. However, such a representation cannot guarantee promising results as shown in Section 4.5.6 of ablation study. The problem is that naively concatenating all features increases the input dimension and does not explore the correlation between layers. For instance, dynamics between nearby layers should have a higher correlation than layers that are far away. This concatenation also loses transferrablity to different network structures.

To maintain a rich state description and generalization in the meanwhile, we construct a computational graph to depict the state. We represent a network as a directed graph $G = (\mathcal{E}, \mathcal{V})$. Each node v denotes an operator $f_v(\cdot; w_v)$ (convolution, linear, etc.) parameterized by w_v which generates an output activation tensor. An edge $e_{uv} \in \mathcal{E}$ represents the computation flow from node u to node v. The output for node v is computed by applying its associated computational operator taking all source nodes as the input. Examples of the graph structure of a neural network are presented in Figure 4.1(a) and (b). For each node v with trainable parameters w_v, the raw feature x_v includes the mean and variance of weights, biases, and absolute gradient values of w_v, as well as global dynamics of the previous learning rate and the average training loss in the decision step. In the end, the environment can be represented by a graph with a feature matrix $\mathbf{X} \in \mathbb{R}^{N \times F}$ and an adjacency matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$, where N is the number of nodes and F is the feature dimension.

Given an input architecture, we describe the state as a graph G with the feature matrix $\mathbf{X}$ and the adjacency matrix $\mathbf{A}$. Then we can feed G into an encoder E, modeled as a graph convolutional neural network [KW16], one of the most effective techniques to achieve message propagation. More concretely, hidden features can be extracted by

$$\mathbf{H}^{l+1} = E(\mathbf{H}^l, \mathbf{A}) = \sigma(\tilde{\mathbf{A}}\mathbf{H}^l\mathbf{W}^l), \tag{4.1}$$

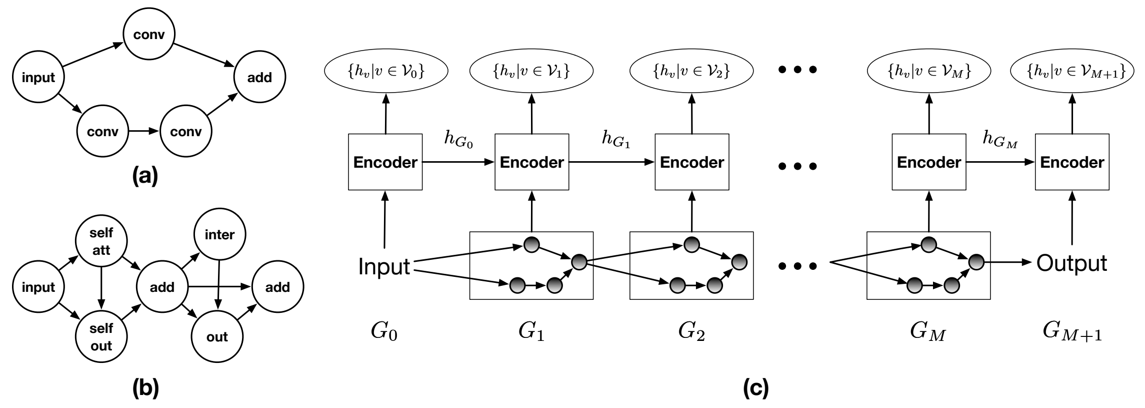

conv
input
add
conv
conv
(a)
self att
inter
input
add
add
self out
out
(b)
$\{h_v|v \in \mathcal{V}_0\}$
$\{h_v|v \in \mathcal{V}_1\}$
$\{h_v|v \in \mathcal{V}_2\}$
$\{h_v|v \in \mathcal{V}_M\}$
$\{h_v|v \in \mathcal{V}_{M+1}\}$
Encoder
h_{G_0}
h_{G_1}
h_{G_M}
Input
Output
G_0
G_1
G_2
G_M
G_{M+1}
(c)

Figure 4.1: (a) A repeated block in ResNet models. (b) A repeated block in Transformer models. (c) Illustration of the hierarchical GCN message passing.

where $\mathbf{H}^0$ is initialized as $\mathbf{X}$, and $\tilde{\mathbf{A}}$ is a normalized adjacency matrix with a self-loop.

However, a deep learning model usually adopts hundreds of operations and results in a large computation graph which is hard to process. We notice that repeated blocks are stacked in commonly-used structures such as ResNet [HZR16] and Transformer [DCL18, VSP17] to boost the expressiveness of the network [ZRU18, ZVS18], shown in Figure 4.1(a) and (b). Instead of passing messages through the whole graph directly, we separate the graph into a number of blocks as $\{G_i = (\mathcal{V}_i, \mathcal{E}_i)\}_{i=0}^{M+1}$ and each block has a group of nodes. M is the number of repeated blocks, and G_0 stands for the input block (e.g., embedding layers) while G_{M+1} represents the output block (e.g., classification layers). We can then use a shared encoder across different blocks as shown in Figure 4.1(c) to conduct information propagation. This is implemented by a hierarchical GCN as follows: (a) compute latent features of nodes in each block by Eq. 4.1; (b) summarize the i-th block as $h_{G_i} = \text{AGG}(\{h_v|v \in \mathcal{V}_i\})$; (c) then use h_{G_i} as the input node feature for the next block: (d) Finally, to obtain the graph level information to describe the whole network, we just take the aggregation over each node by $h_G = \text{AGG}(\{h_v|v \in \mathcal{V}\})$. Here, we use the mean function as aggregation function to allow generalization to different structures.

Compared with a concatenated state with dynamics in all nodes, representing the environment as a graph and then encoding it via a hierarchical GCN have two primary advantages: (1) GCN can capture the correlation between neighboring nodes and generate a meaningful latent feature for action prediction; (2) a shared encoder can learn message passing within one block and between blocks in the meanwhile and is not constrained to a specific graph structure. In detail, the dimension of a concatenated representation varies between different network architectures, for example, RoBERTa-base and -large models. A scheduler learned from such concatenated states fails to work in another environment due to the input mismatch. On the contrary, by stacking the same encoder along the depth dimension, we can transfer our graph-network-based scheduler trained on one network to the new environment of another different network directly or with slight fine-tuning.

4.4.2 Environment design

We discuss the rest components of the environment of the target problem involved in neural network optimization in this part besides the state. An effective set of reward functions and actions is defined to facilitate an efficient and generalizable agent.

Action. We adopt the action as scaling the previous learning rate by a factor [XDK19] considering the wide range of α. Through that operation, the learning rate can be changed stably and consistently. More concretely, after each action execution, $\alpha_{t+1} = \alpha_t \cdot (1 + a_t)$.

State transition. As mentioned in Section 4.3.2, once we modify the learning rate to α_{t+1}, we execute gradient descent for K batches to get the next state s_{t+1}.

Reward design. With the goal of the agent to achieve higher final performance, there are a series of metrics that can be leveraged, such as validation loss and accuracy. In [XDK19], validation loss at each decision step serves as an immediate reward. However, loss might not be an ideal indicator for task specific metrics which we are ultimately concerned with, like accuracy, F1 score and correlation [VTD21]. Thus, we choose to use such task-related metrics directly instead of the loss. On the other hand, since defining an intermediate reward can utilize training information thoroughly and provide direct feedback [XDK19], we take the performance difference between two steps as the reward signal:

$$r_t = P_{t+1} - P_t, \tag{4.2}$$

where P_t represents the validation result under the given metric at the t-th decision step. Whereas, computing a reward in Eq. 4.2 indicates that we have to conduct a validation procedure every decision step, which is a non-trivial burden. In order to make training more efficient, we turn to reduce the frequency of validation with a probability p, i.e., the validation is only carried out if the number randomly sampled from Uni$(0, 1)$ is less than p. Specifically, suppose under our principle, we calculate the validation metric P_{t_1} and P_{t_2} at the decision step t_1 and t_2 respectively. The reward signal for each step can be derived by linearly distributing the difference as follows:

$$r'_t = \frac{P_{t_2} - P_{t_1}}{t_2 - t_1}, \forall t \in [t_1, t_2). \tag{4.3}$$

4.4.3 Graph Network-based Scheduler

With illustration of the environment design, we can then move to the general framework of our Graph Network-based Scheduler (GNS). As shown in Figure 4.2, at each decision step, GNS takes the observation s_t as input, encodes s_t into h_t, sample an action a_t based on the policy network, evaluates the value of s_t, then execute the action to render the next state s_{t+1}.

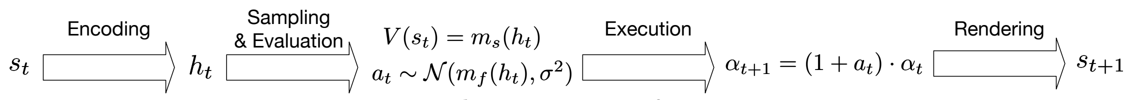


Figure 4.2: The transition from s_t to s_{t+1}.

In the procedure of sampling and evaluation, we use the hidden feature h_t to evaluate the current state and predict the action. Denote m_s and m_f as two different MLPs. Specifically, the state value can be computed by $V(s_t) = m_s(h_t)$. Then we calculate the mean of the action and the policy π samples a_t from the corresponding normal distribution in Eq. 4.4 with a fixed deviation σ:

$$a_t \sim \mathcal{N}(\mu_t, \sigma^2),\ \mu_t = 0.1 \cdot \textbf{tanh}(m_f(h_t)). \tag{4.4}$$

By Eq. 4.4, we can constrain that the scale factor $(1 + a_t)$ is within the range of $[0.9, 1.1]$ at most of the times in order to train the target network stably [XDK19].

As to the agent training, we first parameterize the value network V (critic), and policy network π (actor) by ϕ, φ respectively for clarity. Each environment of the target problem has a fixed time horizon T, i.e., the number of total training steps, which is called as one episode. We adopt Proximal Policy Optimization (PPO) [SWD17] in the actor-critic style to optimize our scheduling policy, which is the state of the art on-policy RL methods in many continuous control tasks [Ach18, BBC19]. On-policy RL methods only use the experience generated by current policy. Therefore, the training will be more stable due to the training data distribution. The key objective function of the policy network is defined as

$$L^{\text{clip}}(\varphi) = \mathbb{E}_t\left[\textbf{min}(\rho_t(\varphi)\hat{A}_t, \textbf{clip}(\rho_t(\varphi), 1-\epsilon, 1+\epsilon)\hat{A}_t)\right], \rho_t(\varphi) = \frac{\pi_\varphi(a_t|s_t)}{\pi_{\varphi_{\text{old}}}(a_t|s_t)}, \tag{4.5}$$

where $\rho_t(\varphi)$ is the probability ratio and $\hat{A}_t$ is the estimated advantage estimator. Besides, a learned state-value function $V_\phi(\cdot)$ is leveraged to reduce variance in advantage estimators and can be trained by minimizing the following squared-error loss:

$$L(\phi) = \mathbb{E}_t[(V_\phi(s_t) - V_t^{\text{targ}})^2], \tag{4.6}$$

where $V_t^{\text{targ}} = r_t + \gamma V(s_{t+1})$. During the training phase, we run the agent in the environment for one episode, and then update the value and policy network at the end. We reset environment at the beginning of each episode and repeat the process for multiple episodes. During inference, we first sample a number of initial learning rates, run each setting for

T_{pre} steps, compute corresponding $V(s_{T_{\text{pre}}})$, and choose the learning rate with the largest value to complete the training.

4.5 Experimental Results

To validate the effectiveness of GNS, we evaluate our method on various tasks in image classification and language understanding, and compare it with popular learning rate scheduling rules. We further investigate the generalization of GNS on different transfer tasks. In addition, we conduct an ablation study to analyze the state representation and reward collection.

4.5.1 Experimental Settings

In this section, we present our experimental settings.

Image classification. We consider two benchmark datasets in image classification, Fashion-MNIST [XRV17] and CIFAR10 [KNH14]. These two datasets are first split into the standard training and test sets. Then we randomly sample 10k images for each dataset from the training set to construct a validation set. A ResNet-18 [HZR16] model is trained on Fashion-MNIST while a ResNet-34 [HZR16] is trained on CIFAR10. We use Adam [KB14] with the batch size of 128 for 200 epochs to train these two image classification tasks.

Language understanding. For language understanding, we conduct experiments on GLUE [WSM19], a benchmark consisting of eight tasks. They are divided into training, validation and test sets and we have no access to ground truth labels of test sets. On each dataset, we fine-tune a pre-trained RoBERTa model [LOG19], use it to generate predictions on the test set, and obtain the corresponding performance through the official GLUE platform. Both RoBERTa-base and -large models are evaluated. No ensembling techniques are adopted for purely validating the effectiveness of our proposed scheduler. The AdamW [LH17] optimizer is adopted to train RoBERTa models.

Baselines. Three hand-designed scheduling mechanisms (constant, polynomial decay, cosine decay), one gradient-based method Hypergradient [BCR17], and one learning-based method, SRLS [XDK19] are compared with our GNS framework as baselines. For traditional schedulers with the warmup process, we use the following equation to summarize them:

$$\alpha_t = \begin{cases} \frac{t_{\text{cur}}}{T_{\text{warm}}}\alpha_{\text{max}}, & \text{warmup;} \\ (\alpha_{\text{max}} - \alpha_{\text{min}})\left(1 - \frac{t_{\text{cur}} - T_{\text{warm}}}{T_{\text{decay}} - T_{\text{warm}}}\right)^p + \alpha_{\text{min}}, & \text{polynomial;} \\ \left[\frac{1}{2}(\alpha_{\text{max}} - \alpha_{\text{min}})\left(1 + \cos(\frac{t_{\text{cur}} - T_{\text{warm}}}{T_{\text{decay}} - T_{\text{warm}}}\pi)\right) + \alpha_{\text{min}}\right], & \text{cosine;} \\ \alpha_{\text{min}}, & \text{constant.} \end{cases} \tag{4.7}$$

In detail, α_{max} and $\alpha_{\text{min}} = \mu\alpha_{\text{max}}$ are the maximum and minimum learning rate respectively during the training. T_{total} is the number of total training steps, $T_{\text{warm}} = \eta_1 T_{\text{total}}$ is the number of warmup steps and $T_{\text{decay}} = \eta_2 T_{\text{total}}$ is the number of decay steps. We tune all hyperparameter combinations within a specific search space and select the configuration with the best validation performance. Then we run the experiment 5 times and report the average test performance. For Hypergraident and SRLS, we follow training details in [BCR17] and [XDK19] respectively.

GNS implementation. To encode raw observations into latent features, we use a 2-layer GCN [KW16] with 64 dimensional node embeddings in all hidden layers, and apply batch normalization for each layer. After a shared encoder, each of the value network and policy network has an MLP with a hidden layer of size 32. Our GNS is trained with PPO [SWD17] for 30 episodes for ResNet models and RoBERTa-base while 10 episodes for RoBERTa-large by fine-tuning the agent learned from RoBERTa-base, by an Adam optimizer of the learning rate 0.005 and 0.001 for the critic and actor respectively.

4.5.2 Results on Image Classification

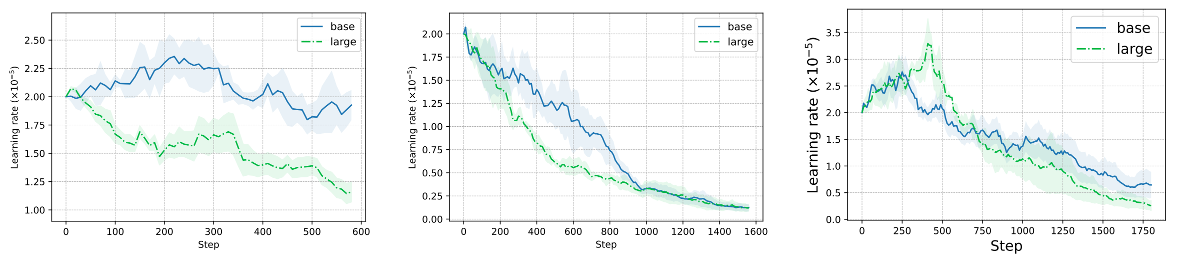
In this section, we present experimental results of image classification tasks. Test accuracy obtained by different schedulers are shown in Table 4.1. It can be observed that the constant learning rate scheduling is insufficient to lead to a satisfactory result and is beaten by all other methods. Although hand-designed schedulers like polynomial and cosine decay and the gradient-based method Hypergradient can make an improvement over a constant learning rate, learning-based methods can still outperform them. In particular, GNS achieves much better accuracy on both Fashion-MNIST and CIFAR10 than SRLS, due to a richer state representation from the utilization of the model's graph structure.

Table 4.1: Test accuracy on Fashion-MNIST and CIFAR10 of different schedulers

Scheduler	**Fashion-MNIST**	**CIFAR10**
Constant	93.0 ± 0.2	93.0 ± 0.4
Polynomial	93.4 ± 0.1	93.5 ± 0.3
Cosine	93.4 ± 0.1	93.6 ± 0.4
Hypergradient	93.5 ± 0.2	93.7 ± 0.6
SRLS	93.6 ± 0.3	93.6 ± 0.6
GNS	$\mathbf{94.6 \pm 0.2}$	$\mathbf{94.3 \pm 0.5}$

4.5.3 Results on Language Understanding

Apart from image classification, we also focus on language understanding tasks on eight GLUE benchmark datasets. Two models with different sizes, RoBERTa-base and RoBERTa-large, are evaluated and detailed results are reported in Table 4.2. As expected, adjusting learning rate rather than keeping a constant one contributes to the test performance. In addition, a consistent performance boost can be observed in GNS over all eight datasets for both base and large models. Specifically, it achieves a score of 85.2 and 87.8 on average in RoBERTa-base and -large model respectively, while the best baseline in each scenario only reaches 84.5 and 86.9. It should be noticed that SRLS performs poorly and even cannot surpass the constant scheduling. It is likely that the simple state design of SRLS (only



(a) MRPC (b) RTE (c) STS-B

Figure 4.3: Learning rate trajectories generated by GNS for three tasks.

using information in the last layer) constrains its ability to adjust the learning rate dynamically, especially for large models like RoBERTa. Thus, SRLS fails to generate a good scheduling to improve the performance.

Table 4.2: Test performance of RoBERTa on GLUE benchmarking of different schedulers.

Scheduler	**CoLA**	**SST-2**	**MRPC**	**QQP**	**STS-B**	**MNLI**	**QNLI**	**RTE**	**Avg.**
Base model:									
Constant	60.5 ± 0.4	95.1 ± 0.2	86.9 ± 0.6	89.0 ± 0.0	88.6 ± 0.2	87.0 ± 0.1	92.9 ± 0.1	73.0 ± 0.3	84.1
Polynomial	61.2 ± 0.3	95.3 ± 0.3	87.2 ± 0.4	89.2 ± 0.1	89.3 ± 0.3	87.3 ± 0.2	93.0 ± 0.1	73.1 ± 0.4	84.4
Cosine	61.2 ± 0.3	95.4 ± 0.3	87.2 ± 0.5	89.2 ± 0.1	89.4 ± 0.1	87.2 ± 0.1	93.0 ± 0.2	73.5 ± 0.3	84.5
Hypergradient	61.0 ± 0.4	95.3 ± 0.2	87.0 ± 0.4	89.1 ± 0.2	89.4 ± 0.2	87.1 ± 0.1	93.0 ± 0.1	73.4 ± 0.5	84.4
SRLS	58.4 ± 1.2	95.3 ± 0.3	86.8 ± 0.5	88.4 ± 0.1	89.4 ± 0.3	86.7 ± 0.2	92.3 ± 0.2	70.8 ± 0.7	83.5
GNS	$\mathbf{63.3 \pm 0.5}$	$\mathbf{95.8 \pm 0.2}$	$\mathbf{87.7 \pm 0.5}$	$\mathbf{89.6 \pm 0.0}$	$\mathbf{89.7 \pm 0.1}$	$\mathbf{87.7 \pm 0.2}$	$\mathbf{93.2 \pm 0.0}$	$\mathbf{74.4 \pm 0.3}$	**85.2**
Large model:									
Constant	63.1 ± 0.7	95.8 ± 0.2	87.7 ± 0.7	89.2 ± 0.0	90.1 ± 0.3	90.0 ± 0.2	94.0 ± 0.2	80.6 ± 0.2	86.3
Polynomial	64.2 ± 0.8	95.9 ± 0.4	88.4 ± 0.7	89.2 ± 0.1	91.1 ± 0.2	90.0 ± 0.0	94.5 ± 0.1	82.0 ± 0.2	86.9
Cosine	63.3 ± 0.6	95.8 ± 0.3	88.5 ± 0.6	89.3 ± 0.0	91.2 ± 0.2	89.9 ± 0.2	94.3 ± 0.2	81.0 ± 0.3	86.7
Hypergradient	63.3 ± 0.7	95.7 ± 0.4	88.2 ± 0.8	89.2 ± 0.1	91.0 ± 0.3	89.9 ± 0.3	94.4 ± 0.2	81.4 ± 0.4	86.6
SRLS	62.3 ± 0.5	95.9 ± 0.3	87.5 ± 1.2	88.2 ± 0.2	91.2 ± 0.4	88.8 ± 0.1	93.4 ± 0.2	79.8 ± 0.3	85.9
GNS	$\mathbf{65.8 \pm 0.7}$	$\mathbf{96.7 \pm 0.1}$	$\mathbf{89.0 \pm 0.9}$	$\mathbf{89.7 \pm 0.1}$	$\mathbf{92.1 \pm 0.2}$	$\mathbf{90.4 \pm 0.1}$	$\mathbf{94.7 \pm 0.1}$	$\mathbf{84.0 \pm 0.2}$	**87.8**

We plot learning rate scheduling trajectories for three tasks in Figure 4.3: MRPC, RTE, and STS-B. It can be observed that different mechanisms are deployed for distinct tasks. The learning rate is generally adjusted in a similar trend for both RoBERTa-base and large models within a task. Specifically, we can see that the learning rate for MRPC is not modified aggressively. For RTE, a decayed scheduling is learned while there is a warmup-like procedure followed by a decreasing tendency for STS-B. Besides, it should be emphasized that the learning rate does not change monotonically, and there are ups and downs during the training to react to the environment observations dynamically.

4.5.4 Generalization of GNS

Due to the utilization of hierarchical GCN as the learning rate scheduler described in Section 4.4.3, our GNS can be transferred to different but similar graphs, e.g., ResNet and VGG [CSB16] structures on CIFAR10. Thus, we conduct an experiment to evaluate the generalization of the learned scheduler. We select the policy network trained under the environment of ResNet-34, and apply it directly to the task with VGG-16. For all other baselines, we simply adopt the best hyperparameter configuration of ResNet-34 to train the VGG-16 network. Results are shown in the second column of Table 4.3. We can see that GNS can still achieve the best performance on the VGG model while the hyperparameter settings of other methods are not suitable for the VGG model. It indicates that GNS is able to learn the fundamental rule to conduct message passing between nodes and is generalizable to various graphs. The excellent transferred performance from RoBERTa-base model to large model also explains why only a few episodes are needed when fine-tuning the agent under RoBERTa-large models.

Table 4.3: Transferred performance of GNS.

Scheduler	**ResNet → VGG**	**CIFAR10 → CIFAR100**
Constant	91.4 ± 0.3	69.3 ± 0.3
Polynomial	91.8 ± 0.3	70.5 ± 0.4
Cosine	91.7 ± 0.4	71.2 ± 0.2
Hypergradient	91.5 ± 0.5	70.9 ± 0.3
SRLS	91.7 ± 0.3	71.0 ± 0.6
GNS	**92.5 ± 0.4**	**72.3 ± 0.3**

In addition, GNS can be utilized to the same network structure but on different datasets. Specifically, we apply the graph scheduler trained on ResNet-34 for CIFAR10 to training a ResNet-34 network for CIFAR100. It can be observed in the third column of Table 4.3 that GNS achieves 72.3% on test accuracy and outperforms 71.2% of the best baseline scheduler with a cosine decay. Such a performance further validates that GNS is generalizable to different datasets.

4.5.5 Running time comparison

Compared with SRLS in which validation is required in every decision step, GNS can reduce a considerable amount of time with the validation probability $p = 0.2$. We consider training RoBERTa-base in this section. For instance, when running on MRPC for 5 epochs, we need to make 58 decisions with the number of network updates $K = 10$. The average time of one episode with one NVIDIA 1080Ti GPU of SRLS is 405s while GNS only takes 259s, which decreases the original cost by 30%.

For hand-designed rules, there are 40 hyperparameter settings shown to be considered for RoBERTa fine-tuning while it takes 30 episodes to train the learned scheduler. The time for running one setting is 223s on average. Thus, a thorough grid search among these 40 settings consumes approximately 2.5 hours. On the contrary, GNS takes 2.2 hours and even achieves better results. Furthermore, fine-tuning the scheduler on RoBERTa-large from the base model only requires 10 episodes and costs much less time than grid search in all configurations.

4.5.6 Ablation Study

In this part, an ablation study is carried out to corroborate that the design of state representations and reward collection play an important role in the GNS framework.

Table 4.4: Test performance of three scheduler variants for RoBERTa-base.

Scheduler	CoLA	SST-2	MRPC	QQP	STS-B	MNLI	QNLI	RTE	Avg.
SRLS	58.4 ± 1.2	95.3 ± 0.3	86.8 ± 0.5	88.4 ± 0.1	89.4 ± 0.3	86.7 ± 0.2	92.3 ± 0.2	70.8 ± 0.7	83.5
CSS	60.0 ± 0.5	94.9 ± 0.1	87.1 ± 0.3	89.2 ± 0.2	88.7 ± 0.3	86.4 ± 0.1	92.3 ± 0.2	72.2 ± 0.5	83.8
FRS	$\mathbf{63.5 \pm 0.3}$	$\mathbf{95.9 \pm 0.3}$	$\mathbf{87.7 \pm 0.4}$	89.4 ± 0.0	89.4 ± 0.2	87.6 ± 0.2	$\mathbf{93.2 \pm 0.1}$	$\mathbf{74.6 \pm 0.4}$	**85.2**
GNS	$\mathbf{63.3 \pm 0.5}$	$\mathbf{95.8 \pm 0.2}$	$\mathbf{87.7 \pm 0.5}$	$\mathbf{89.6 \pm 0.0}$	$\mathbf{89.7 \pm 0.1}$	$\mathbf{87.7 \pm 0.2}$	$\mathbf{93.2 \pm 0.0}$	$\mathbf{74.4 \pm 0.3}$	**85.2**

State representations with graph networks. To show advantages in encoding the state by a graph neural network, we compare our GNS with a variant called concatenated-state scheduler (CSS), where features of all nodes in the state graph are concatenated to form a super representation and then fed into subsequent neural networks for controlling the learning rate. Compared with the basline SRLS that only uses the final layer state, CSS uses all layers but without using GCN to exploit correlation between layers and cannot be applied to different target models. We can observe in Table 4.4 that using a concatenated state slightly outperforms SRLS but is still insufficient to render as a competitive scheduler as GNS, which shows the importance of encoding by graph neural networks.

Reward collection with a probability. We also compare the difference between collecting a reward signal at every decision step (the variant is denoted as full-reward scheduler, FRS) and collecting it with a probability. We report the evaluation metric on the test set in Table 4.4. We can see that an instant reward cannot necessarily improve the final performance of the learned scheduler. It is likely that the performance does not improve monotonically and typically oscillates during the training. For example, in two consecutive decision steps, the latter one gives us a worse validation result, leading to a negative reward computed by Eq. 4.2 and indicating that the action is not good. However, this information might be misleading and impede agent training. Thus, to alleviate such a phenomenon, we design a stochastic reward collection procedure for more useful signals.

4.6 Conclusion

This work proposes a novel graph-network-based scheduler to control the learning rate dynamically by reinforcement learning. We are the first to take into consideration the structure of the target optimization problem with neural networks, and hereby leverage message passing via the graph to encode the state for predicting the action. Comprehensive experiments have shown that GNS can consistently achieve performance boosts on benchmark datasets in both image classification and language understanding. However, in this work we only focus on problems with a moderate time horizon. How to efficient train a learning rate scheduler for a problem with hundreds of thousands optimization steps such as BERT pre-training can be a potential future direction.

CHAPTER 5

FedDM: Iterative Distribution Matching for Communication-Efficient Federated Learning

Besides optimization techniques, algorithm efficiency can be improved by dataset distillation, which we will delve into in the following chapter. Specifically, we integrate dataset distillation into existing federated learning pipelines to reduce communication rounds between the server and the clients while maintaining comparble or even better performance.

5.1 Introduction

Traditional machine learning methods are designed with the assumption that all training data can be accessed from a central location. However, due to the growing data size together with the model complexity [DBK20, KT19, KSH12], distributed optimization [SSZ14, DCM12, CSA14] is necessary over different machines. This leads to the problem of Federated Learning [MMR17] (FL) – multiple clients (e.g. mobile devices or local organizations) collaboratively train a global model under the orchestration of a central server (e.g. service provider) while the training data are kept decentralized and private. Such a practical setting poses two primary challenges [KMA21, MMR17, LWW21, KMY16, LSZ20]: **training data of the FL system are highly unbalanced and non-i.i.d. across downstream clients** and **more efficient communication with fewer costs is expected** because of unreliable devices with limited transmission bandwidth.

Most of the existing FL methods [MMR17, LSZ20, WLL20, KKM20, LDC22] adopt an iterative training procedure from FedAvg [MMR17], in which each round takes the following steps: 1) The global model is synchronized with a selected subset of clients; 2) Each client trains the model locally and sends its weight or gradient back to the server; 3) The server updates the global model by aggregating messages from selected clients. This framework works effectively for generic distributed optimization while the difficult and challenging setting of FL, unbalanced data partition in particular, would result in statistical heterogeneity in the whole system [LST20] and make the gradient from each client inconsistent. It poses a great challenge to the training of the shared model, which requires a substantial number of communication rounds to converge [LDC22]. Although some improvements have been made over FedAvg [MMR17] including modifying loss functions [LSZ20], correcting client-shift with control variates [KKM20] and the like, the reduced number of communication round is still considerable and even the amount of information required by the server rises [ZPM20].

In our paper, we propose a different iterative surrogate minimization based method, **FedDM**, referred to **Fed**erated Learning with iterative **D**istribution **M**atching. Instead of the commonly-used scheme where each client maintains a locally trained model respectively and sends its gradient/weight to the server for aggregation, we take a distinct perspective at the client's side and attempt to build a local surrogate function to approximate the local training objective. By sending those local surrogate functions to the server, the server can then build a global surrogate function around the current solution and conduct the update by minimizing this surrogate. The question is then how to build local surrogate functions that are informative and with a relative succinct representation. Inspired by recent progresses in data condensation [ZMB20, ZB21b] we build local surrogate functions by learning a synthetic dataset to replace the original one to approximate the objective. It can be achieved by matching the original data distribution in the embedding space [GBR12]. After the optimization of synthesized data, the client transmits them to the server, which then leverages the synthetic dataset to recover the global objective function for training. Our method enables the server to have implicit access to the global objective defined by the whole balanced dataset from all clients, and thus outperforms previous algorithms involved in training a local model with unbalanced data in terms of communication efficiency and effectiveness. We also show that our method can be adapted to preserve differential privacy, an important factor to the deployment of FL systems.

Our contributions are primarily summarized as follows:

- We propose FedDM by iterative distribution matching to learn a surrogate function. It sends synthesized data to the server rather than commonly-used local model updates and improves communication efficiency and effectiveness significantly.
- We analyze how to protect privacy of client's data for our method and show that it is able to guarantee (ϵ, δ)-differential privacy with the Gaussian mechanism and train a better model under the same privacy budget.
- We conduct experiments on three tasks and demonstrate that FedDM outperforms its FL counterparts in communication efficiency and the final model performance.

5.2 Related Work

Federated Learning. Federated learning [MMR17, KMA21] has aroused heated discussion nowadays from both research and applied areas. With the goal to train the model collaboratively, it incorporates the principles of focused data collection and minimization [KMA21]. FedAvg [MMR17] was proposed along with the concept of FL as the first effective method to train the global model under the coordination of multiple devices. Since it is based on iterative model averaging, FedAvg suffers from heterogeneity in the FL system, especially the non-i.i.d. data partitioning, which degrades the performance of the global model and adds to the burden of communication [LST20]. To mitigate the issue, some variants have been developed upon FedAvg including [LSZ20, WLL20, KKM20].

For instance, FedProx [LSZ20] modifies the loss function. FedNova [WLL20] and SCAFFOLD [KKM20] leverage auxiliary information to balance the distribution shift. Apart from better learning algorithms with faster convergence rate, another perspective at improving efficiency is to reduce communication costs explicitly [CSP21, SWM19, XKG19, CSJ19, RMH20]. An intuitive approach is to quantize and sparsify the uploaded weights directly [RMH20]. Efforts have also been made towards one-shot federated learning, expecting to obtain a satisfactory model through only one communication round [ZPM20, GTS19, SSG21, SSG19].

Differential Privacy. To quantify information disclosure about individuals, researchers usually adopt the state-of-the-art model, differential privacy (DP) [DMN06, DR14, Dwo11]. DP describes the patterns of groups while withholding information about individuals in the dataset. There are many scenarios in which DP guarantee is necessary [DKM06, DL09, ACG16, PS21, ASY18, McS09, KOV15]. For example, [ACG16] developed differentially private SGD (DP-SGD) which enabled training deep neural networks with non-convex objectives under a certain privacy budget. It was further extended to settings of federated learning, where various techniques have been designed to guarantee client-level or instance-level differential privacy [MRT17, PAE16, YIL22]. Recently, DP has been taken into account for hyperparameter tuning [PS21].

Dataset Distillation. With the explosive growing of the size of training data, it becomes much more challenging and costly to acquire large datasets and train a neural network within moderate time [NCL20, NNX21]. Thus, constructing smaller but still informative datasets is of vital importance. The traditional way to reduce the size is through coreset selection [BMK20], which select samples based on particular heuristic criteria. However, this kind of method has to deal with a trade-off between performance and data size [NCL20, ZMB20]. To improve the expressiveness of the smaller dataset, recent approaches consider learning a synthetic set from the original data, or data distillation for simplicity. Along this line, different methods are proposed using meta-learning [WZT18, SS21], gradient matching [ZMB20, ZB21a], distribution matching [ZB21b, WZP22], neural kernels [NCL20, NNX21] or generative models [SRL20]. These methods demonstrate great potentials in datasets such as CIFAR10 but face challenges in scaling up to larger ones like ImageNet. Besides, a recent work [DZL22] has analyzed the privacy property of dataset distillation methods, focusing on membership inference attacks. It provides a complementary prospective to (ϵ, δ)-differential privacy discussed in our paper.

5.3 Methodology

In this part, we first present the iterative surrogate minimization framework in Section 5.3.1, and then expand on the details of our implementation of FedDM in Section 5.3.2. In addition, we discuss preserving differential privacy of our method through Gaussian mechanism in Section 5.3.3.

5.3.1 Iterative surrogate minimization framework for federated learning

Neural network training can be formulated as solving a finite sum minimization problem:

$$\min_{w} f(\mathcal{D}; w) \quad \text{where} \quad f(\mathcal{D}; w) = \frac{1}{n} \sum_{i=1}^{n} \ell(x_i, y_i; w), \tag{5.1}$$

where $w \in \mathbb{R}^d$ is the parameter to be optimized, $\mathcal{D}$ is the dataset and $\ell(x_i, y_i; w)$ is the loss of the prediction on sample $(x_i, y_i) \in \mathcal{D}$ w.r.t. w such as cross entropy. We will abbreviate these terms as $f(w)$ and $\ell_i(w)$ for simplicity. Equation (5.1) is typically solved by stochastic optimizers when training data are gathered in a single machine. However, the scenario is different under the setting of federated learning with K clients. In detail, each client k has access to its local dataset of the size n_k with the set of indices $\mathcal{I}_k$ ($n_k = |\mathcal{I}_k|$), and we can

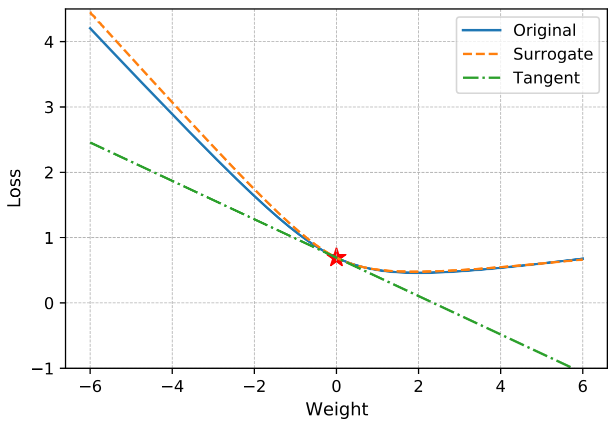


Figure 5.1: A 1-D example showing advantages of minimizing the surrogate function.

rewrite the objective as

$$f(w) = \sum_{k=1}^{K} \frac{n_k}{n} f_k(w) \quad \text{where} \quad f_k(w) = \frac{1}{n_k} \sum_{i \in \mathcal{I}_k} l_i(w). \tag{5.2}$$

Since information can only be communicated between the server and clients, previous methods [MMR17, LSZ20, WLL20, KKM20] train the global model by aggregation of local model updates, as introduced in Section 5.1. However, as each client only sees local data which could be biased and skewed, the local updates is often insufficient to capture the global information. Further, since local weight update consists limited information, it is hard for the server to obtain better joint update direction by considering higher order interactions between different clients. We are motivated to leverage the surrogate function by the example in Figure 5.1. Specifically, we synthesize a 1-D binary classification problem and learn a surrogate for the objective function. We learn the surrogate function via distribution matching introduced in Section 5.3.2 around the weight of 0. Compared with the tangent line computed by the gradient, the surrogate function in orange matches the original one accurately and minimizing it leads to a satisfactory solution. More details can be checked in Appendix B.Thus, we hope to develop a novel scheme such that each client can send a **local surrogate function** instead of a single gradient or weight update to the server, so the server has a more global view to loss landscape to obtain a better update instead of pure averaging.

To achieve this goal, we propose to conduct federated learning with an iterative surrogate minimization framework. At each round, let w_r be the current solution, we build a surrogate training objective $\hat{f}_r(\cdot)$ to approximate the original training objective in the local area around w_r, and then update the model by minimizing the local surrogate function. The update rule can be written as

$$w_{r+1} = \min_{w \in B_\rho(w_r)} \hat{f}_r(w), \tag{5.3}$$

where $\hat{f}_r(w) \approx f(w), \forall w \in B_\rho(w_r)$ and $B_\rho(w_r)$ is a ρ-radius ball around w_r. We do not expect to build a good surrogate function in the entire parameter space; instead, we only construct it near w_r and obtain the update by minimizing the surrogate function within this space. Many optimization algorithms can be described under this framework. For instance, if $\hat{f}_r(w) = \nabla f(w_r)^T(w - w_r)$ (based on the first-order Taylor expansion), then Equation (5.3) leads to the gradient descent update where ρ controls the step size.

To apply this framework in the federated learning setting, we consider the decomposition of Equation (5.2) and try to build surrogate functions to approximate each $f_k(w)$ on each client. More specifically, each client aims to find

$$\hat{f}_{r,k}(w) \approx f_k(w), \ \ \forall w \in B_\rho(w_r) \tag{5.4}$$

and send the **local surrogate function** $\hat{f}_{r,k}(\cdot)$ instead of gradient or weights to the server. The server then form the **aggregated surrogate function**

$$\hat{f}_r(w) = \hat{f}_{r,1}(w) + \cdots + \hat{f}_{r,K}(w) \tag{5.5}$$

and then use Equation (5.3) to obtain the update. Again, if each $\hat{f}_{r,k}$ is the Taylor expansion based on local data, it is sufficient for the client to send local gradient to the server, and the update will be equivalent to (large batch) gradient descent. However, we will show that there exists other ways to build local approximations to make federated learning more communication efficient.

5.3.2 Local distribution matching

Inspired by recent progresses in data distillation [ZMB20, ZB21b, ZB21a, NCL20, NNX21], it is possible to learn a set of synthesized data for each client to represent original data in terms of the objective function. Therefore, we propose to build local surrogate models based on the following approximation for the r-th round:

$$f_k(w) \approx \frac{1}{n_k^s} \sum_{j \in \mathcal{I}_k^{\mathcal{S}}} \ell_j(\tilde{x}_j, \tilde{y}_j; w) = \hat{f}_{r,k}(\mathcal{S}; w), \ \ \forall w \in B_\rho(w_r), \tag{5.6}$$

where $\mathcal{S}$ denotes the set of synthesized data and $\mathcal{I}_k^{\mathcal{S}}$ is the corresponding set of indices. Note that we aim to approximate f_k only in a local region around w_r instead of finding the approximation globally, which is hard as demonstrated in [ZMB20, ZB21b]. To form the approximation function in Equation (5.6), we solve the following minimization problem:

$$\min_{\mathcal{S}} \mathbb{E}_{w \sim \mathcal{P}_w} \| f_k(w) - \hat{f}_{r,k}(\mathcal{S}; w) \|^2 \tag{5.7}$$

where w is sampled from distribution $\mathcal{P}_w$, which is a Gaussian distribution truncated at radius ρ. A different perspective at Equation (5.7) is that we can just match the distribution between the real data and synthesized ones given $f_k(w)$ and $\hat{f}_{r,k}(\mathcal{S}; w)$ are just empirical risks. A common way to achieve this is to estimate the real data distribution in the latent space with a lower dimension by maximum mean discrepancy (MMD) [WZP22, ZB21b]: $\sup_{\|h_w\|_{\mathcal{H} \le 1}} (\mathbb{E}[h_w(\mathcal{D})] - \mathbb{E}[h_w(\mathcal{S})])$. Here $\mathcal{H}$ is reproducing kernel Hilbert space and h_w is the embedding function that maps the input into the hidden representation. We use the empirical estimate of MMD in [ZB21b] since the underlying distribution is inaccessible. To make our approximation more accurate and effective, we match outputs of the logit layer which corresponds with Equation (5.7), together with the preceding embedding layer:

$$\begin{aligned}\mathcal{L}_{\text{DM}} = &\mathbb{E}_{w \in B_\rho(w_r)} \| \frac{1}{|\mathcal{D}|} \sum_{(x,y)\in\mathcal{D}} h_w(x) - \frac{1}{|\mathcal{S}|} \sum_{(\tilde{x},\tilde{y})\in\mathcal{S}} h_w(\tilde{x}) \|^2 \\ &+\mathbb{E}_{w \in B_\rho(w_r)} \| \frac{1}{|\mathcal{D}|} \sum_{(x,y)\in\mathcal{D}} z_w(x) - \frac{1}{|\mathcal{S}|} \sum_{(\tilde{x},\tilde{y})\in\mathcal{S}} z_w(\tilde{x}) \|^2,\end{aligned} \tag{5.8}$$

where $h_w(x)$ again denotes intermediate features of the input while $z_w(x) \in \mathbb{R}^C$ represents the output of the final logit layer. Note that we learn synthesized data for each class respectively, which means samples in $\mathcal{D}$ and $\mathcal{S}$ belong to the same class. For training, we adopt mini-batch based optimizers to make it more efficiently. Specifically, a batch of real data and a batch of synthetic data are sampled randomly for each class independently by $B_c^{\mathcal{D}_k} \sim \mathcal{D}_k$ and $B_c^{\mathcal{S}_k} \sim \mathcal{S}_k$. We plug these two batches into Equation (5.8) to compute $\mathcal{L}_c$ and $\mathcal{L} = \sum_{c=0}^{C-1} \mathcal{L}_c$. $\mathcal{S}_k$ can be updated with SGD by minimizing $\mathcal{L}$ for each client.

Then we aggregate all synthesized data from K clients at the server's side to approximate the global objective function, which is computed as

$$f(w) = \sum_{k=1}^{K} \frac{n_k}{n} f_k(w) \approx \sum_{k=1}^{K} \frac{n_k^{\mathcal{S}}}{n} \hat{f}_{r,k}(\mathcal{S}_k; w), \;\; \forall w \in B_\rho(w_r). \tag{5.9}$$

Moreover, since synthesized data are trained based on a specific distribution around the current value of w, we need to iteratively synchronize the global weights with all the clients and obtain proper $\mathcal{S}$ according to the latest w for the next communication round.

Therefore, instead of transmitting information such as parameters or gradients in previous FL algorithms, we propose federated learning with iterative distribution matching (FedDM) in Algorithm 3 following the steps below to train the global model:

(a) At each communication round, for each client, we adopt Equation (5.8) as the objective function to train synthesized data for each class.

(b) The server receives synthesized data and leverages them to update the global model.

(c) The current weight is then synchronized with all the clients and a new communication round starts by repeating step (a) and (b).

It should be noticed that through estimating the local objective, FedDM extracts richer information than existing model averaging based methods, and enables the server to explore the loss landscape from a more global view. It reduces communication rounds significantly. On the other hand, the explicit message uploaded to the server, or the number of float parameters, is relatively smaller. This is especially true when training large neural network models, where the size of neural network parameters (and therefore gradient update) is much larger than the size of the input. Take CIFAR10 as an example, when training data are distributed obeying $\mathrm{Dir}_{10}(0.5)$, the average number of classes per client (cpc) is 9. When we adopt the number of images per class (ipc) of 10 for the synthetic set, the total number of float parameters uploaded to the server is: the number of clients $\times$ cpc $\times$ ipc $\times$ image size $= 10 \times 9 \times 10 \times 3 \times 32 \times 32 \approx 2.8 \times 10^6$. For those iterative model averaging model methods, the number of float parameters is equal to the product of weight size and the number of clients, which is $320010 \times 10 \approx 3.2 \times 10^6$ for ConvNet [ZMB20] and comparably larger than FedDM. An extensive comparison is presented in Appendix C.

5.3.3 Differential privacy of FedDM

An important factor to evaluate a federated learning algorithm is whether it can preserve differential privacy. Before analyzing our method, we first review some fundamentals of differential privacy.

Definition 5.3.1 (Differential Privacy [DKM06]) *A randomized mechanism* $\mathcal{M}$:

Algorithm 3 FedDM: Federated Learning with Iterative Distribution Matching

```
1: **Input:** Training set $\mathcal{D}$, set of synthetic samples $\mathcal{S}$, deep neural network parameterized with $w$, probability distribution over parameters $\mathcal{P}_w$, training iterations of distribution matching $T$, learning rate $\eta_c$ and $\eta_s$.
2: **Server executes:**
3: **for** each round $r = 1, \ldots, R$ **do**
4:     **for** client $k = 1, \ldots, K$ **do**
5:         $\mathcal{S}_k \leftarrow$ ClientUpdate($k, w_r$)
6:         Transmit $\mathcal{S}_k$ to the server
7:     **end for**
8:     Aggregate synthesized data from each client and build the surrogate function by Equation (5.9)
9:     Update weights to $w_{r+1}$ on $\mathcal{S}$ by SGD with the learning rate $\eta_s$
10: **end for**
11: **ClientUpdate**($k, w_r$):
12: Initialize $\mathcal{S}_k$ from random noise or real examples.
13: **for** $t = 0, \cdots, T-1$ **do**
14:     Sample $w \sim P_w(w_r)$
15:     Sample mini-batch pairs $B_c^{\mathcal{D}_k} \sim \mathcal{D}_k$ and $B_c^{\mathcal{S}_k} \sim \mathcal{S}_k$ for each class $c$
16:     Compute $\mathcal{L}_c$ based on Equation (5.8), $\mathcal{L} \leftarrow \sum_{c=0}^{C-1} \mathcal{L}_c$
17:     Update $\mathcal{S}_k \leftarrow \mathcal{S}_k - \eta_c \nabla_{\mathcal{S}_k} \mathcal{L}$
18: **end for**
```

$\mathcal{D} \rightarrow \mathcal{R}$ with domain $\mathcal{D}$ and range $\mathcal{R}$ satisfies (ϵ, δ)-differential privacy if for any two adjacent datasets D_1, D_2 and any measurable subset $S \subseteq \mathcal{R}$,

$$\boldsymbol{Pr}(\mathcal{M}(D_1) \in S) \leq e^{\epsilon} \boldsymbol{Pr}(\mathcal{M}(D_2) \in S) + \delta. \tag{5.10}$$

In this paper, we focus on instance-level differential privacy, which indicates that D_1 and D_2 differ on a single element. Typically, the randomized mechanism is applied to a query function of the dataset, $f : \mathcal{D} \rightarrow \mathcal{X}$. Without loss of generality, we assume that the output spaces $\mathcal{R}, \mathcal{X} \subseteq \mathbb{R}^m$. A key quantity in characterizing differential privacy for various mechanisms is the sensitivity of a query [DR14] $f : \mathcal{D} \rightarrow \mathbb{R}^m$ in a given norm ℓ_p. Formally this is defined as

$$\Delta_p \triangleq \max_{D_1, D_2} \|f(D_1) - f(D_2)\|_p. \tag{5.11}$$

Gaussian mechanism [DR14] is one simple and effective method to achieve (ϵ, δ)-differential privacy:

$$\mathcal{M}(D) \triangleq f(D) + Z, \quad \text{where} \quad Z \sim \mathcal{N}(0, \sigma^2 \Delta_p^2 \mathbf{I}). \tag{5.12}$$

It has been proved that under Gaussian mechanism, (ϵ, δ)-differential privacy is satisfied for the function f of sensitivity Δ_p if we choose $\sigma \geq \sqrt{2 \log \frac{1.25}{\delta}} / \epsilon$ [DR14]. Differentially private SGD (DP-SGD) [ACG16] then applies Gaussian mechanism to deep learning optimization with hundreds of steps and demonstrates the following theorem:

Theorem 1 (Differential Privacy of DP-SGD) *There exist constants c_1 and c_2 so that given the sampling probability q and the number of steps T, for any $\epsilon < c_1 q^2 T$, DP-SGD is (ϵ, δ)-differentially private for any $\delta > 0$ if*

$$\sigma \geq c_2 \frac{q\sqrt{T \log(1/\delta)}}{\epsilon}. \tag{5.13}$$

We then prove that by leveraging DP-SGD to update the synthetic dataset which is initialized from random Gaussian noise, FedDM can preserve differential privacy of the original dataset. We present this DP guarantee of FedDM in the theorem below:

Theorem 2 (Differential privacy of FedDM.) *Given the synthetic dataset $\mathcal{S}$ is initialized from random noise, FedDM trained with DP-SGD can guarantee (ϵ, δ)-differential privacy in a K-client federated learning system, with $\sigma \geq \sqrt{\frac{\log(\delta)}{Tq^2 - \epsilon}}$ or $\sigma \geq \sqrt{\frac{2\log(1/\delta)}{\epsilon}}$ if $Tq^2 \leq \epsilon/2$ in each communication round.*

A complete proof and an initial analysis of differential privacy for R communication rounds are included in Appendix D.We also present the whole procedure of FedDM integrated with DP-SGD in Appendix E.

Table 5.1: Summary of different FL methods.

Method	Client	Message	Server
FedAvg [MMR17]	min $f_k(w)$	Δ_w *	model averaging
FedProx [LSZ20]	min $f_k(w) + \mu\|w - w_r\|/2$	Δ_w	model averaging
FedNova [WLL20]	min $f_k(w)$	d and a *	normalized model averaging
SCAFFOLD [KKM20]	min $f_k(w, c)$	Δ_w and Δ_c *	model averaging for both w and c
FedDM(Ours)	min Equation (5.8)	$\mathcal{S}$	model updating on $\mathcal{S}$

* Δ_w denotes the model update, d is the aggregated gradient and a is the coefficient vector, Δ_c is the change of control variates. Refer to original papers for more details.

Table 5.2: Test accuracy of FL methods with different level of non-uniform partitioning.

Method	$\alpha = 0.1$			$\alpha = 0.01$		
	MNIST	CIFAR10	CIFAR100	MNIST	CIFAR10	CIFAR100
FedAvg	96.92 ± 0.09	57.32 ± 0.04	32.00 ± 0.50	91.04 ± 0.80	39.28 ± 0.25	27.05 ± 0.45
FedProx	96.72 ± 0.04	56.92 ± 0.30	30.77 ± 0.52	91.18 ± 0.16	40.30 ± 0.15	25.88 ± 0.39
FedNova	98.04 ± 0.03	60.76 ± 0.14	31.92 ± 0.42	90.27 ± 0.49	36.46 ± 0.42	27.52 ± 0.43
SCAFFOLD	98.32 ± 0.06	60.96 ± 1.20	34.39 ± 0.25	88.37 ± 0.25	32.42 ± 1.13	31.14 ± 0.20
FedDM	**98.67 ± 0.01**	**67.38 ± 0.32**	**37.58 ± 0.27**	**98.21 ± 0.23**	**63.82 ± 0.17**	**34.98 ± 0.17**

5.4 Experiments

5.4.1 Experimental setup

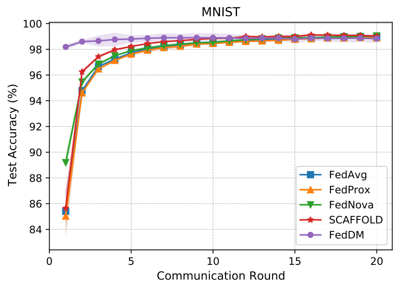
Datasets. In this paper, we focus on image classification tasks, and select three commonly-used datasets: MNIST [LeC98], CIFAR10 [KH09], and CIFAR100 [KH09]. We adopt the standard training and testing split. Following commonly-used scheme [WYS19], we simulate non-i.i.d. data partitioning with Dirichlet distribution $\text{Dir}_K(\alpha)$, where K is the number of clients and α determines the non-i.i.d. level, and allocate divided subsets to clients respectively. A smaller value of α leads to more unbalanced data distribution. The default data partitioning is based on $\text{Dir}_{10}(0.5)$ with 10 clients. Furthermore, we also take into account different scenarios of data distribution, including $\text{Dir}_{10}(0.1)$, $\text{Dir}_{10}(0.01)$.

Baseline methods. We compare FedDM with four representative iterative model averaging based methods: FedAvg [MMR17], FedProx [LSZ20], FedNova [WLL20], and SCAFFOLD [KKM20]. We summarize the action of the client and the server, and the transmitted message for all methods in Table 5.1. Two stronger methods, FedAvgM [HQB19] and FedAdam [RCZ], are compared in Appendix F.1.

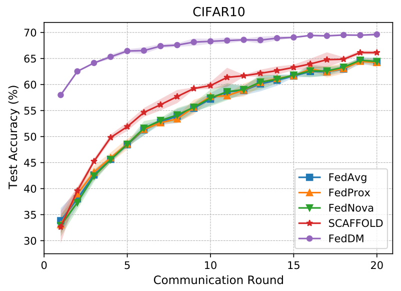
Hyperparameters. For FedDM, following [ZB21b], we select the batch size as 256 for real images, and update the synthetic set $\mathcal{S}_k$ for $T = 1,000$ iterations with $\eta_c = 1$ for each client in each communication round, and tune the number of images per class (ipc) within $[3, 5, 10]$. Synthetic images are initialized as randomly sampled real images with corresponding labels suggested by [ZMB20, ZB21b], and random noise initialization is leveraged when differential privacy is required. Considering the trade-off between communication efficiency and model performance, we choose ipc to be 10 for MNIST and CIFAR10, 5 for CIFAR100 when there are 10 clients. The choice of radius $\rho = 5$ is discussed in Section 5.4.5. On the server's side, the global model is trained with the batch size 256 for 500 epochs by SGD of $\eta_s = 0.01$. For baseline methods[1], we choose the same batch size of 256 for local training. We tune the learning rate at the client from $[0.001, 0.01, 0.1]$ and at the server from $[0.01, 0.1, 1]$, and local epoch from $[1, 2, 5, 10, 15, 20]$. In particular, we tune μ for FedProx in $[0.01, 0.1, 1]$. For a fair comparison, all methods share the fixed number of communication rounds as 20, and the same model structure ConvNet [ZMB20] by default. All experiments are run for three times with different random seeds with one NVIDIA 2080Ti GPU and the average performance is reported.

[1]We use implementations from `https://github.com/Xtra-Computing/NIID-Bench` in [LDC22].

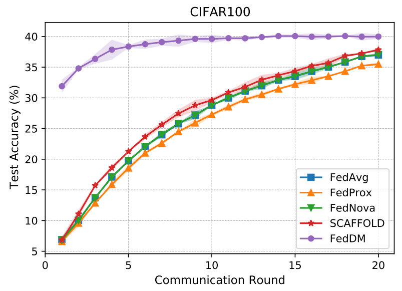
5.4.2 Communication efficiency & convergence rate



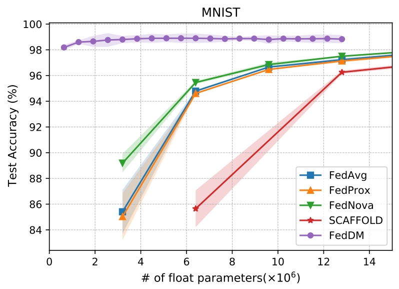
(a) MNIST; rounds



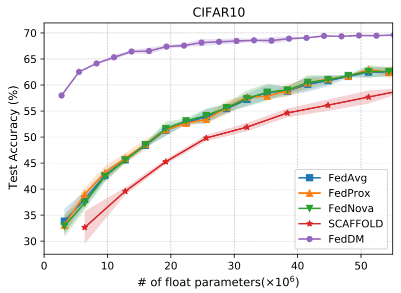
(b) CIFAR10; rounds



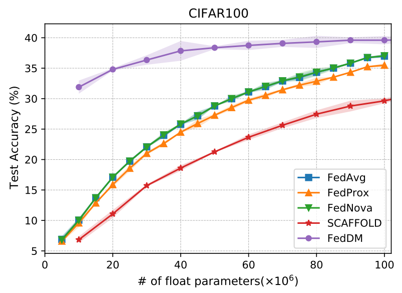
(c) CIFAR100; rounds



(d) MNIST; message size



(e) CIFAR10; message size



(f) CIFAR100; message size

Figure 5.2: Test accuracy along with the number of communication rounds and the message size. Within the limited communication budget, FedDM performs the best in all three datasets, in terms of efficiency and final test accuracy.

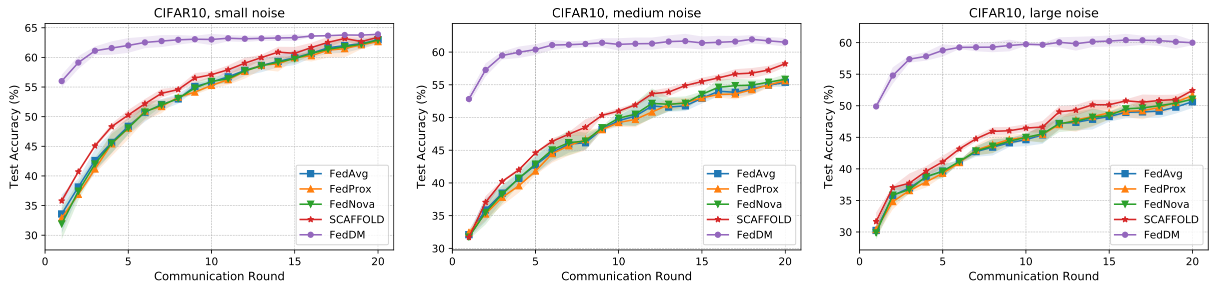
We first evaluate our method in terms of communication efficiency and convergence rate on all three datasets on the default data partitioning $\text{Dir}_{10}(0.5)$. As we can see in Figure 5.2(a)-(c), our method FedDM performs the best among all considered algorithms by a large margin on MNIST, CIFAR10, and CIFAR100. Specifically, for CIFAR10, FedDM achieves $69.66 \pm 0.13\%$ on test accuracy while the best baseline SCAFFOLD only has $66.12 \pm 0.17\%$ after 20 communication rounds. FedDM also has the best convergence rate and it significantly outperforms baseline methods within the initial few rounds. Advantages of FedDM are more evident when we evaluate convergence as a function of the message size. As mentioned in 5.3.2, FedDM requires less information per round. Therefore, we can observe in Figure 5.2(d)-(e) that FedDM converges the fastest along with the message size. Details of the message size of each method for different tasks are provided in Appendix C.

5.4.3 Evaluation on different data partitioning

In real-world applications, there are various extreme data distributions among clients. To synthesize such non-i.i.d. partitioning, we consider two more scenarios with $\mathrm{Dir}_{10}(0.1)$ and $\mathrm{Dir}_{10}(0.01)$. As mentioned, $\alpha \to 0$ implies each client holds examples from only one random class. It can be seen in Table 5.2 that previous methods based on iterative model averaging are insufficient to handle these two challenging scenarios and their performance degrades drastically compared with $\mathrm{Dir}_{10}(0.5)$. In contrast, FedDM performs consistently better and more robustly, since distribution matching enables it to approximate the global training objective more accurately.

5.4.4 Performance with DP guarantee

As discussed in Section 5.3.3, if the synthetic dataset is initialized from random noise, using DP-SGD in local training of FedDM can satisfy (ϵ, δ)-differential privacy, with $\sigma \geq \sqrt{\frac{2\log(1/\delta)}{\epsilon}}$ for any $Tq^2 \leq \epsilon/2$, a relatively loose bound independent of training steps T. To make a fair comparison, we use tensorflow privacy to compute ϵ with a tight bound given the number of examples, batch size, training steps, $\delta = 10^{-5}$ under $\sigma \in 1, 3, 5$ for FedDM, and obtain noise levels for baseline methods accordingly which are $[0.44, 0.76, 0.95]$ respectively. We tune clipping norm $C \in [1, 3, 5, 10]$. $\mathcal{S}$ is initialized from $\mathcal{N}(0, 1)$ to guarantee differential privacy. We notice in Figure 5.3 that under the same DP guarantee, FedDM outperforms other FL counterparts in terms of convergence rate and final performance. Moreover, the accuracy of FedDM does not drop significantly compared with all considered methods when the noise level increases, indicating that FedDM is most resistant to the perturbed optimization.



(a) Small noise ($\epsilon = 12.25$). (b) Medium noise ($\epsilon = 2.46$). (c) Large noise ($\epsilon = 1.35$).

Figure 5.3: Performance of FL methods with different noise levels. To preserve differential privacy, FedDM initializes the synthetic data from random Gaussian noise.

5.4.5 Analysis of FedDM

In this section, we analyze FedDM to investigate effects of hyperparameters including the initialization of the synthetic dataset, image-per-class (ipc), network structure and selection of ρ–radius ball. Besides, we compare our method with a strong baseline of sending real images with the same size. Extensive results are reported in Appendix F.1.

Initialization of the synthetic dataset. We conduct an ablation study on the initialization of the synthetic dataset $\mathcal{S}$ on CIFAR10 with the default partition $\text{Dir}_{10}(0.5)$. In detail, *random* initialize $\mathcal{S}$ based on the standard normal distribution $\mathcal{N}(0, 1)$ while *real* samples instances from the original dataset to be distilled. It can be observed in Table 5.3 that *real* performs consistently better than *random*, which concurs with the conclusion in [ZB21b] and justifies the choice of *real* in our experiments. Note that even *random* can outperform model averaging methods compared with results in Figure 5.2 and Table 5.2. In addition, *random* with DP-SGD helps preserve differential privacy of FedDM, and still improves the efficiency and accuracy significantly in Figure 5.3.

Effects of ipc. Experiments are conducted on CIFAR10 with the distribution $\text{Dir}_{10}(0.5)$ with three different ipc values from $[3, 5, 10]$. As the ipc increases, the performance gradually get better from $53.64 \pm 0.35\%$, $62.24 \pm 0.04\%$ to $69.62 \pm 0.14\%$. On the other hand, more

Partitioning	*random*	*real*
$\alpha = 0.5$	64.12 ± 0.15	$\mathbf{69.66 \pm 0.13}$
$\alpha = 0.1$	62.75 ± 0.24	$\mathbf{67.38 \pm 0.32}$
$\alpha = 0.01$	59.00 ± 0.26	$\mathbf{63.82 \pm 0.17}$

Table 5.3: Test accuracy of FedDM with *random* and *real* $\mathcal{S}$ initialization on CIFAR10. Three data partitionings are evaluated.

images per class indicates a heavier communication burden in the meanwhile. We need to trade off the model performance against the communication cost, and thus choose an appropriate ipc value based on the task.

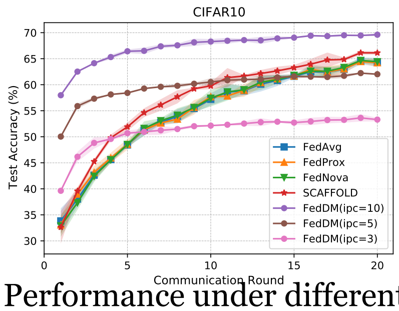


Figure 5.4: Performance under different ipc values.

Different network structures. Besides ConvNet, we evaluate FedDM under the default CIFAR10 setting on ResNet-18. It can be observed that our method works well even for this more complicated and larger model in Figure 5.5. It should also be emphasized that for FL baseline methods, they have to transmit a larger amount of message while FedDM maintains the original size. This makes FedDM more efficient in larger networks.

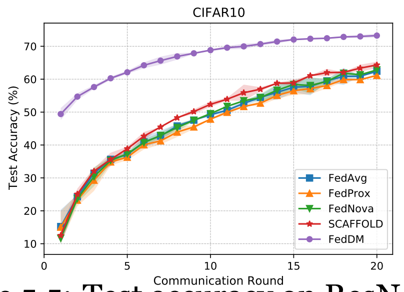


Figure 5.5: Test accuracy on ResNet-18.

Selection of ρ-radius ball. It has been discussed in Section 5.3.1 that $B_\rho(w_r)$ is a ρ-radius ball around w_r:

$$B_\rho(w_r) = \{w | \|w - w_r\|_2 \leq \rho\}. \tag{5.14}$$

In FedDM, we sample w based on a truncated Gaussian distribution below:

$$P_w(w_r) = \text{Clip}(\mathcal{N}(w_r, 1), \rho), \tag{5.15}$$

where we clip the sampled weight to guarantee that $\|w - w_r\|_2 \leq \rho$. At the server's side, when training the global model, we also clip the weight to the ρ-radius ball. We conduct experiments to choose ρ from $[3, 5, 10]$ and present the test accuracy after 20 communication rounds on CIFAR10 under the default $\text{Dir}_{10}(0.5)$ setting in Table 5.4. We find that performance is similar and FedDM is not very sensitive to the choice of ρ. $\rho = 5$ performs relatively the best and we hypothesize that a too small weight restricts the optimization to a limited range and a too big one adds to the difficulty of learning a surrogate function. Based on results in Table 5.4, we select $\rho = 5$ for all our experiments.

ρ	Test accuracy (%)
$\rho = 3$	69.15 ± 0.09
$\rho = 5$	**69.66 ± 0.13**
$\rho = 10$	69.32 ± 0.24

Table 5.4: Test accuracy of FedDM under different ρ.

Comparison with transmitting real images. Our method is compared with REAL, which sends real images of the same size as FedDM (ipc $=$ 10). In particular, REAL achieves test acccuracy of 68.66 ± 0.08% on CIFAR10 with the default setting, but cannot beat FedDM with 69.62 ± 0.14%. It indicates that our learned synthetic set can capture richer information of the whole dataset rather than just a few images.

5.5 Conclusions and Limitations

In this work we propose an iterative distribution matching based method, FedDM, to achieve more communication-efficient federated learning. By learning a synthetic dataset for each client to approximate the local objective function, the server can obtain a global view of the loss landscape better than aggregating local model updates. We also show that FedDM can preserve differential privacy with Gaussian mechanism. However, there is still a trade-off between the size of the synthetic set and the final performance, especially for classification tasks with hundreds of clients or classes. How to reduce the synthetic set to save communication costs can be a future direction.

Part II

Data Efficient Deep Learning

CHAPTER 6

Extreme Zero-Shot Learning for Extreme Text Classification

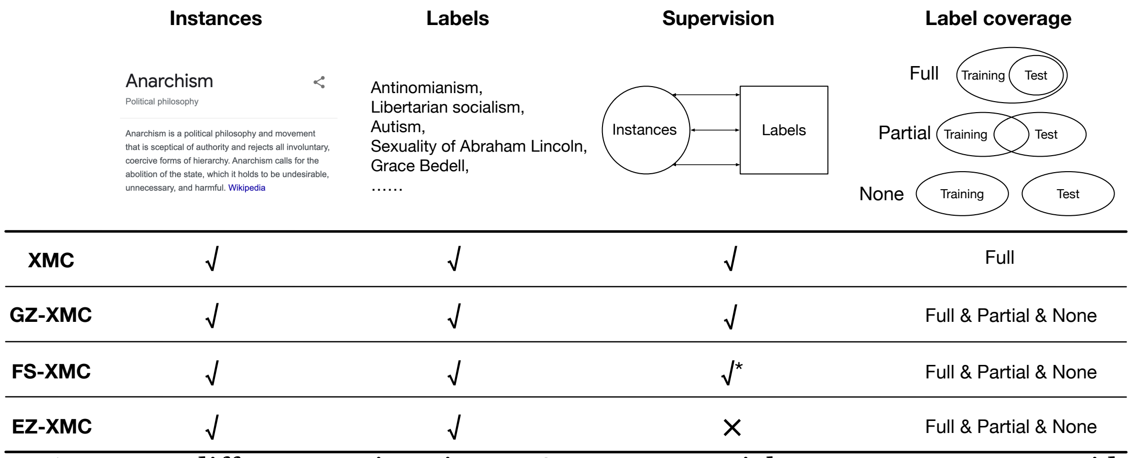
In parallel with algorithm efficiency, data efficiency is another essential factor that influences the development of deep learning. In this chapter, we present a pre-training method to tackle zero-shot extreme text classification, given limited or even no supervision.

6.1 Introduction

The e**X**treme **M**ulti-label text **C**lassification (XMC) problem aims at tagging a text input with most relevant subset of labels from an extremely large output space. Many web-related applications can be formulated as an XMC task with encouraging results, such as finding the best matching products from a large catalog in e-commerce systems [MHW19, CJY21], auto-completing queries given its prefix on search engines [YSH21], predicting search keywords for dynamic advertising [PKH18, CYZ20], tagging categories of Wikipedia articles from a large label taxonomy [DS10, CFM19], to name just a few.

The current XMC setup is built on full label coverage and full supervision, where full label coverage means labels to be predicted have appeared in the training set and full supervision indicates it requires a significant number of annotated (instance, label) pairs. In detail, it is assumed that an XMC algorithm has access to raw text of instances and labels, together with their corresponding relations during training, as shown in Figure 6.1.

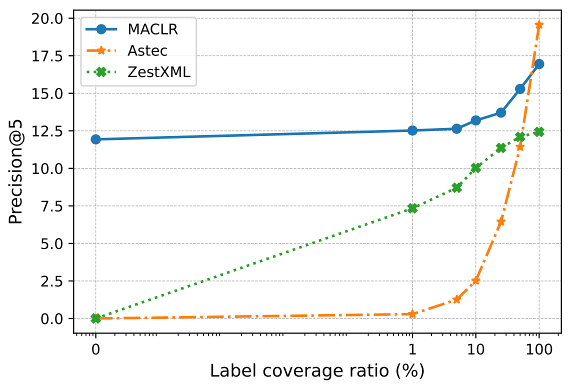
However, there are several limitations of this XMC setting. First of all, due to the as-


Anarchism
Political philosophy
Anarchism is a political philosophy and movement that is sceptical of authority and rejects all involuntary, coercive forms of hierarchy. Anarchism calls for the abolition of the state, which it holds to be undesirable, unnecessary, and harmful. Wikipedia
Antinomianism,
Libertarian socialism,
Autism,
Sexuality of Abraham Lincoln,
Grace Bedell,
……
Instances
Labels
Full
Partial
None
Training
Test

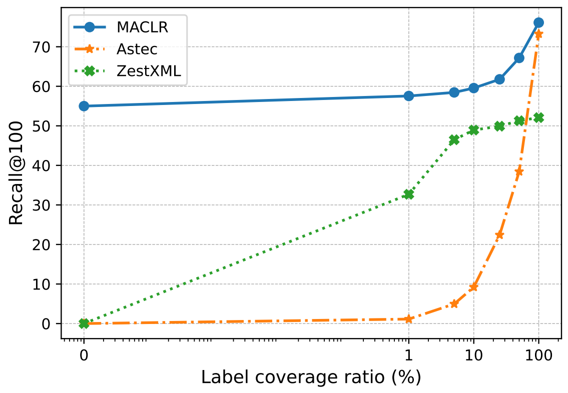
	Instances	Labels	Supervision	Label coverage
XMC	√	√	√	Full
GZ-XMC	√	√	√	Full & Partial & None
FS-XMC	√	√	√*	Full & Partial & None
EZ-XMC	√	√	×	Full & Partial & None

Figure 6.1: Four different settings in XMC. Four essential components are considered: instances (raw text), labels (raw text), supervision (positive pairs), and label coverage. We divide label coverage into 3 groups: full, partial, and none. * in FS-XMC emphasizes that only a limited amount of supervision is available. We can see that EZ-XMC is the most general and practical setting, where no supervision and label coverage are required.

sumption of full label coverage, it is typical in XMC approaches to simply treat labels as IDs for classification and thus they are restricted to making predictions within observed labels. This assumption is unrealistic since the label set usually keeps growing over time, e.g., newly added websites or products which are absent during training yet crucial for applications such as recommendation and advertising. Besides, collecting labeled pairs is time-consuming, expensive and sometimes infeasible, for example, launching an e-commerce system in the emerging locale, where no user behavioral signals are avaiable. In spite of these constraints, most existing methods [DSM21, YZW19, MSA21, DAS21] followed this XMC setup. It can be seen in Figure 6.2 that Astec [DSM21], one of the state-of-the-art extreme classifiers, is incapable of handling the scenario without supervision, which leads to zero performance in both Precision@5 and Recall@100. Moreover, the increasing trend in Astec's performance along with the label ratio suggests that it depends highly on the supervision level and is hard to generalize to unseen labels. This motivates us to investigate how to design an effective XMC model with zero supervision.



(a) Precision@5.



(b) Recall@100.

Figure 6.2: Performance of three representative XMC methods on LF-Amazon-131K at different ratios of label coverage. A subset covering $[0, 1, 5, 10, 25, 50, 100](\%)$ of the whole label set is sampled for fine-tuning.

In this work, we consider an essential yet under-explored XMC setting, called Extreme Zero-shot XMC (EZ-XMC). As depicted in Figure 6.1, we can access raw text of both instances and labels but do not know their corresponding relations in EZ-XMC. Moreover, we do not make any assumption on the label coverage, so the labels in the testing set may or may not appear in the training stage. An extension to EZ-XMC with a limited number of training pairs, Few-shot XMC (FS-XMC), is also taken into account in our work. Either EZ-XMC or FS-XMC occurs frequently in the real world since informative and abundant (instance, label) pairs are never easy to obtain. Also, it is more practical and worthwhile to reduce labor for manual annotation by solving problems under EZ-XMC. Note that generalized zero-shot XMC (GZ-XMC) proposed in a recent work [GBP21] can be regarded as a special case of EZ-XMC. GZ-XMC allows that the set of test labels is not completely overlapped with training labels but still requires supervision from positive pairs, as shown in Figure 6.1. From Figure 6.2, we can observe that ZestXML [GBP21] designed for GZ-XMC also suffers the issue of no supervision.

A natural question then arises: how should we deal with EZ-XMC problems? Despite the name, EZ-XMC is barely easy to tackle. Fortunately, although dedicated supervision signals are lacking, raw text of instances and labels, e.g., product descriptions and categories, are still accessible in EZ-XMC. Thus it is of vital importance to effectively leverage self-information of these data to train a model for classification. To overcome challenges encountered in EZ-XMC, we turn to solving the problem from a different perspective without learning classifiers explicitly. In particular, XMC can be cast into a problem which learns a sentence encoder $\mathcal{E}$ to map instances and labels into dense embeddings, and predictions are made through approximate nearest neighbor search algorithms in the latent space [SL14]. Motivated by recent progresses in self-supervised learning [GYC21, CKN20, HFW20, DCL18], we propose **MACLR** (**M**ulti-scale **A**daptive **C**lustering & **L**abel **R**egularization), a two-stage pre-training procedure with those unpaired raw data to obtain a sentence encoder $\mathcal{E}$ under EZ-XMC. As to FS-XMC, fine-tuning the encoder on a few paired data is sufficient for the performance boost. Figure 6.2 demonstrates that MACLR achieves superior performance when no supervision is available and achieves much higher recall than Astec and ZestXML by a large margin, even under the higher label coverage ratio.

Our main contributions are summarized below:

- We propose an essential Extreme Zero-Shot XMC (EZ-XMC) setting without any assumptions on supervision and label coverage, which has not been explored in previous work and is more practical in real applications.
- We leverage unlabeled data to pretrain the sentence encoder $\mathcal{E}$ with improved Inverse Cloze Task in Stage I of MACLR. In particular, multi-scale adaptive clustering and label regularization are proposed to utilize raw text thoroughly. In Stage II, we further self-train the encoder with pseudo positive pairs constructed from $\mathcal{E}$ in Stage I as well as TF-IDF model with complementary information.
- Extensive experiments are conducted on four public EZ-XMC benchmark datasets. Results demonstrate that our pre-trained encoder can outperform existing unsupervised baseline methods notably. As an example, MACLR achieves Recall@100 of

54.99%, nearly the same level as Astec [DSM21], one of the SOTA methods, trained with a supervised subset covering around 70% labels on LF-Amazon-131K.

- MACLR can also achieve comparable or even better performance under the few-shot setting than those heavily dependent on supervised information. For example, MACLR is better than the SOTA ZestXML [GBP21] in Recall@100 over 20% (57.55% v.s. 32.69%) when fine-tuned on the subset covering 1% labels of LF-Amazon-131K.

6.2 Related Work

Extreme multi-label classification Various extreme classifiers have been proposed to address the large output space challenge of XMC problems. We can broadly categorize them into two groups: partitioned-based models with linear classifiers [PKH18, PV14, YZZ20] that partition labels with hierarchical trees, leading to sub-linear inference time complexity, and embedding-based methods [BJK15, JBC19, GMW19] that learn a classifier for each label and leverage approximated nearest neighbor [MY18, GKC16] to index labels in the large output space. There are also deep learning models such as AttentionXML [YZW19], Astec [DSM21], SiameseXML [DAS21], and XR-Transformer [ZCY21] that further improve the accuracy of those linear counterparts with various advanced encoder architectures. Nevertheless, none of those XMC methods can handle the EZ-XMC setup: they not only suffer from the lack of supervised signals, but also fail to generalize to unseen cold-start labels in the test set. The only exception is ZestXML [GBP21], a recently proposed XMC method that was designed to address the generalized zero-shot XMC (GZ-XMC) problem where a number of labels for prediction are absent during training. While ZestXML partially resolves the generalization challenge of cold-start labels, just like those conventional XMC models, it still depends heavily on a large number of training data with positive (instance, label) pairs.

Self-supervised learning techniques Past few years have witnessed great promise in self-supervised learning, where a pre-training task is defined using only data's self-information [LCG20, CKN20, HFW20, DCL18, KTW20, GYC21]. Learned representations from the pre-training task can be then leveraged in a wide range of downstream tasks in various domains, such as image classification [CKN20, HFW20] and object detection [LHQ20] in computer vision, and open-domain question answering [LCT19, GLT20] in natural language processing. Specifically, we focus on contrastive approaches for SentenceBERT [RG19] models in this work, where the intuition is to pull semantically close neighbors together and push apart non-neighbors via noise contrastive estimation or N-pair losses. Various effective pre-training tasks such as Inverse Cloze Task (ICT) [LCT19] and SimCSE [GYC21] have been shown to further improve the performance.

6.3 Problem Formulation

In this section, we present the problem formulation of EZ-XMC. With $\mathcal{X}$ and $\mathcal{Y}$ denoting the set of instances and labels respectively, the general XMC problem can be viewed as learning a scoring function $f : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$. $f(\cdot, \cdot)$ maps an (instance, label) pair (x, y) to a similarity score, which is used to make a prediction through approximate nearest neighbor search algorithms. In previous settings such as XMC and GZ-XMC, a considerable amount of relevant (instance, label) pairs $\{(x_i, y_i)\}$ are available. On the contrary, in EZ-XMC, we have no knowledge about corresponding relations between instances and labels, but only their raw text, as shown in Figure 6.1. Thus, existing approaches dependent on relevant pairs fail to learn an effective scoring function, even with a few paired data under FS-XMC.

Recent progresses in self-supervised learning have shown that a generalized sentence encoder can be learned through elaborately designed pre-training tasks even without any supervision [LCT19, CYC20], and then adapted to different downstream tasks directly or via slight finetuning. On the other hand, the scoring function f can be modeled as

$f(x, y) = \langle \mathcal{E}(x), \mathcal{E}(y) \rangle$, where $\mathcal{E}$ is a sentence encoder producing semantical dense embeddings, and $\langle \cdot, \cdot \rangle$ is the similarity measurement such as inner product and cosine similarity. Without loss of generality, inner product is adopted in the paper as the similarity metric between embeddings of instances and labels. Thus, we formulate the problem as training an encoder $\mathcal{E}$ with raw text of $\mathcal{X}$ and $\mathcal{Y}$ through a pre-training task for EZ-XMC. As to the few-shot scenario FS-XMC, we can fine-tune $\mathcal{E}$ for improvement.

6.4 Method

In this section, we introduce a two-stage pre-training procedure, MACLR, to thoroughly leverage unpaired data with raw text for EZ-XMC. Specifically, we present the general framework in Section 6.4.1, and then dive into details of two stages, pre-training with the improved Inverse Cloze Task and self-training with pseudo positive pairs, in Sections 6.4.2 and 6.4.3 respectively.

6.4.1 Framework

The framework of our pre-training procedure is shown in Figure 6.3. MACLR consists of two stages:

- Stage I: title-context pairs are constructed for the Inverse Cloze Task, and the encoder $\mathcal{E}$ is then trained on these pairs together with two proposed techniques, multi-scale adaptive clustering and label regularization.
- Stage II: More pseudo positive pairs are crafted using different score functions modeled by the encoder from Stage I and TF-IDF respectively. $\mathcal{E}$ is further trained on additional pairs to improve the encoding performance.

Details of each component are discussed in the following sections.

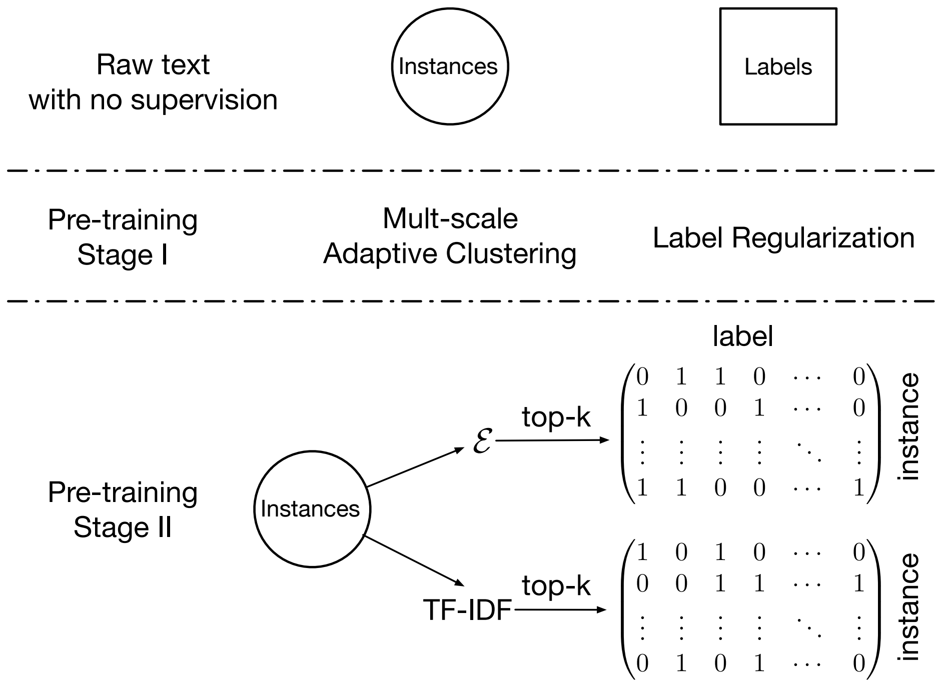


Figure 6.3: Framework of our pre-training procedure.

6.4.2 Stage I: Pre-training with improved ICT

Inverse Cloze Task [LCT19] is a frequently used pre-training task for the sentence encoder. Specifically, for an instance $x = \{s_1, \dots, s_n\}$ consisting of n sentences, ICT randomly samples a sentence to serve as the pseudo positive label $\hat{y} = s_i$ where $i \sim [1, n]$. Then the rest of x is the pseudo instance $\hat{x} = \{s_1, \dots, s_{i-1}, s_{i+1}, \dots, s_n\}$. In XMC, due to the property that the label usually summarizes the instance with one short sentence, which works similarly as the title s_1, we directly utilize (context, title) pairs in the form of ($\hat{x} = \{s_2, \dots, s_n\}, \hat{y} = s_1$). This construction works as the analog of the ground truth (instance, label) pairs and capture the semantics of a sentence. With these pseudo pairs, the contrastive training objective for a mini-batch of N pairs is as follows:

$$\mathcal{L}_{\text{contrastive}} = -\sum_{i=1}^{N} \log \frac{\exp(\mathcal{E}(\hat{x}_i) \cdot \mathcal{E}(\hat{y}_i))}{\sum_{j=1}^{N} \exp(\mathcal{E}(\hat{x}_i) \cdot \mathcal{E}(\hat{y}_j))} \tag{6.1}$$

Based on ICT, we also develop two techniques, multi-scale adaptive clustering and label reguarization, to fully leverage the information of unpaired instances and labels.

6.4.2.1 Multi-scale Adaptive Clustering

In the original ICT scheme, we can construct only one positive pair for a particular instance. It is relatively hard in contrastive learning without enough positive examples, especially for extreme multi-label classification where one instance might be associated with more than one label, and a label is also likely to point to several different instances at the same time. Thus a question arises naturally: is it possible to construct more positive pairs from purely unpaired raw data to intergrate richer information into the pre-training process? We solve it by the unsupervised K-means clustering. In detail, we divide pseudo (context, title) pairs from ICT into K clusters through K-means based on the embeddings of all instances. Then if $C(\hat{x}_i) = C(\hat{x}_j)$, i.e., $\hat{x}_i$ and $\hat{x}_j$ belong to the same cluster, $(\hat{x}_i, \hat{y}_j)$ and $(\hat{x}_j, \hat{y}_i)$ are regarded as positive pairs besides original ICT pairs. Furthermore, supervised contrastive loss is adopted for training the encoder with a mini-batch of N pairs based on the cluster assignment:

$$\mathcal{L}_{\text{cluster}} = \sum_{i=1}^{N} \frac{-1}{|P_{\mathcal{Y}}(i)|} \sum_{p \in P_{\mathcal{Y}}(i)} \log \frac{\exp(\mathcal{E}(\hat{x}_i) \cdot \mathcal{E}(\hat{y}_p))}{\sum_{j=1}^{N} \exp(\mathcal{E}(\hat{x}_i) \cdot \mathcal{E}(\hat{y}_j))} \tag{6.2}$$

Here, $P_{\mathcal{Y}}(i) = \{p \in \{1, \ldots, N\} : C(\hat{x}_i) = C(\hat{x}_p)\}$ is the set of indices of all positives for $\hat{x}_i$ in the batch, and $|P_{\mathcal{Y}}(i)|$ is its cardinality. Minimizing Equation (6.2) pulls close the representations of instances and their positive labels within the same cluster and pushes away the representations of those from different clusters.

Besides, since the ultimate goal is the minimization of Equation (6.1), we propose a multi-scale approach with adaptive training, which guides the encoder to learn the easier tasks with sufficient positive examples, and then master harder tasks gradually. This approach allows the encoder to learn from the coarse scale to the fine scale of clustering assignment, and is similar to the idea of curriculum learning [BLC09] to first focus on learning from a subset of simple examples, and expanding to include the remaining harder samples. Our adaptive training process can be conducted by modifying the cluster size to adjust the task difficulty accordingly. To be specific, we initialize the cluster assignment with the number of clusters K_0, and double the cluster size every T steps. The cluster assignment is also updated every T_{update} steps along with the training of $\mathcal{E}$. Such a process lasts for half of the total training steps T_{total} to take advantage of positive examples from constructed clusters. The obtained intermediate encoder from this adaptive procedure is expected to satisfactorily capture the semantics of a sentence and is ready to deal with the optimization of Equation (6.1). Then for the rest half of training steps, we turn to the hardest setting treating each instance as one independent cluster, which exactly falls into the contrastive training objective in Equation (6.1). Our multi-scale adaptive clustering is illustrated in Figure 6.4.

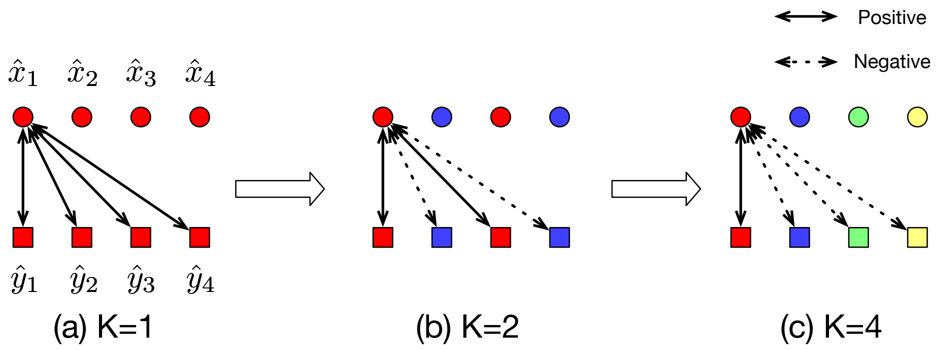

Positive
Negative
$\hat{x}_1$ $\hat{x}_2$ $\hat{x}_3$ $\hat{x}_4$
$\hat{y}_1$ $\hat{y}_2$ $\hat{y}_3$ $\hat{y}_4$
(a) K=1
(b) K=2
(c) K=4

Figure 6.4: An example of multi-scale adaptive clustering. Different colors represent different clusters. (a) In the beginning, there is only one cluster and $\{\hat{y}_j\}_{j=1}^4$ are all positive labels for $\hat{x}_1$. (b) K is doubled to 2 and $\hat{y}_1$ and $\hat{y}_3$ are positive to $\hat{x}_1$. (c) Finally, K is equal to 4 where each instance is a cluster, and hence $\hat{x}_1$ only has one positive label $\hat{y}_1$. The process is similar for the rest of the instances.

6.4.2.2 Label Regularization

In addition to leveraging information from the instance side, we also have access to the raw texts of the whole label set and can utilize them to boost the encoder's performance from the label side [MKR22, MPF22]. Intuitively, for a randomly sampled label, with a high probability it is an negative example to the instance of interest. We can take advantage of this intuition to make the embedding of the instance far from its irrelevant labels. Instead of increasing the distance directly, it is more stable and effective to adopt contrastive losses. To avoid overfitting, we choose a new positive example for each instance instead of its corresponding pseudo label from ICT which has been used in $\mathcal{L}_{\text{cluster}}$. More concretely, $\hat{x}_i^+$ is selected exactly the same as $\hat{x}_i$, since the dropout layer is placed in the standard training of Transformer-based models and can be viewed as a minimal form of data augmentation [GYC21]. By feeding the same sentence to the encoder $\mathcal{E}$, two embeddings with different dropout masks are obtained, i.e., $\hat{h}_i = \mathcal{E}(\hat{x}_i, z_i)$ and $\hat{h}_i^+ = \mathcal{E}(\hat{x}_i^+, z_i^+)$ where z represents a random mask for dropout. $\hat{h}_i \neq \hat{h}_i^+$ due to the dropout noise, but they hold similar semantics from the same sentence and thus can be used as a positive pair for contrastive learning. The procedure of label regularization is depicted in Figure 6.5. At each step, we sample M real labels from the label set $\mathcal{Y}$, and the reguarization term is computed as follows:

$$\mathcal{L}_{\text{label}} = \sum_{i=1}^{N} -\log \frac{\exp(\hat{h}_i \cdot \hat{h}_i^+)}{\sum_{j=1}^{M} \exp(\hat{h}_i \cdot \mathcal{E}(y_j^-)) + \exp(\hat{h}_i \cdot \hat{h}_i^+)} \tag{6.3}$$

Through minimizing $\mathcal{L}_{\text{label}}$, the encoder learns to pull the instance away from its irrelevant labels and incorporate the dropout augmentation at the same time. Together with $\mathcal{L}_{\text{cluster}}$, we have the final objective function for pre-training in the Stage I as $\mathcal{L} = \mathcal{L}_{\text{cluster}} + \mathcal{L}_{\text{label}}$.

6.4.3 Stage II: Self-training with multi-viewed pseudo pairs

After the pre-training procedure in Section 6.4.2, we can obtain an intermediate encoder $\mathcal{E}_I$. But are there any ways to further improve the encoder? Inspired by self-training in semi-supervised learning [YJC19, XLH20, HGS20, ZGL20], $\mathcal{E}_I$ can be leveraged to make predictions on those unpaired training instances themselves, to generate pseudo positive

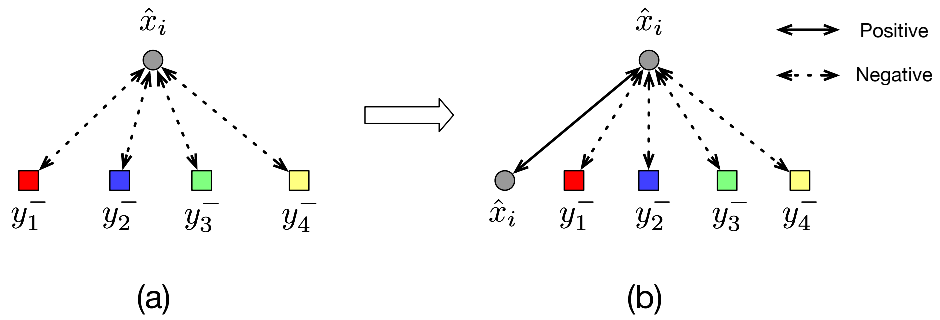


Figure 6.5: An illustration of label reguarization. (a) shows that $\hat{x}_i$ is expected to be far from sampled irrelevant labels $\{y_j^-\}_{j=1}^4$, while (b) indicates the identical $\hat{x}_i$ is added as a positive example for label regularization.

pairs. These pseudo pairs are much better than random guessing and can serve as a distinct view from ICT pairs. On the other hand, similar pseudo pairs can be constructed by other unsupervised methods such as TF-IDF, which provide different and complementary information about the instance.

With multi-viewed pseudo positive pairs, we can conduct further training on the encoder in State II from a new perspective and self-improve $\mathcal{E}_I$. The detailed process works as follows:

1) Compute the similarity score using $\mathcal{E}_I$ for each training instance x_i, and select labels with top-k maximum scores as its pseudo labels;
2) Generate labels similarly with TF-IDF, except that $\mathcal{E}(x)$ and $\mathcal{E}(y)$ are replaced with their TF-IDF vectors;
3) Mix pseudo positive pairs from 1) and 2), and train $\mathcal{E}_I$ on them with Equation (6.2).

6.4.4 MACLR Algorithm

The whole pre-training procedure of MACLR is shown in Algorithm 4. Note that for FS-XMC, we simply fine-tune the encoder $\mathcal{E}$ from MACLR on available positive pairs for several steps by minimizing the original contrastive loss in Equation (6.1).

Algorithm 4 Pre-training procedure of MACLR

```
Require: Raw text of instances and labels $(\mathcal{X}, \mathcal{Y})$, the sentence encoder $\mathcal{E}$, batch size $N$
    and $M$, training step parameters $T_K$, $T_{\text{update}}$ and $T_{\text{total}}$, initial cluster size $K_0$, # of top
    candidates $k$
Ensure: A pre-trained sentence encoder $\mathcal{E}$
    ▷ Stage I: Pre-training with the improved ICT
 1: Construct ICT (context, title) pairs from raw texts in $\mathcal{X}$
 2: Feed the context for each pair into the encoder $\mathcal{E}$ and cluster them into $K = K_0$ clusters
    via k-means
 3: for $t = 1, \ldots, T_{\text{total}}$ do
 4:     Sample a mini-batch of pseudo pairs of size $N$ and a mini-batch of real labels of
    size $M$
 5:     Compute the loss: $\mathcal{L} = \mathcal{L}_{\text{cluster}} + \mathcal{L}_{\text{label}}$
 6:     Train the encoder by minimizing $\mathcal{L}$
 7:     if $t$ mod $T_K = 0$ and $t < T_{\text{total}}/2$ then
 8:         $K = K * 2$
 9:     end if
10:     if $t$ mod $T_{\text{update}} = 0$ and $t < T_{\text{total}}/2$ then
11:         Feed raw texts of $\mathcal{X}$ again into $\mathcal{E}$, and update current cluster assignment via
    k-means with the cluster number $K$
12:     end if
13:     if $t \geq T_{\text{total}}/2$ then
14:         Treat each instance as an independent cluster
15:     end if
16: end for
    ▷ Stage II: Self-training with multi-viewed pseudo pairs
17: Construct pseudo pairs $(\mathcal{X}_{\text{pseu}}, \mathcal{Y}_{\text{pseu}})$ by selecting top-$k$ candidate labels with the sim-
    ilarity metric on the encoder $\mathcal{E}$ and TF-IDF respectively
18: Train the encoder $\mathcal{E}$ for $T_{\text{total}}$ steps by minimizing Equation (6.2)
```

6.5 Experimental Results

6.5.1 Experimental Settings

Datasets We evaluate MACLR on 4 public XMC benchmark datasets [BDJ16, GBP21] where raw text of instances and labels are available. These datasets are derived from real-world applications, ranging from item-to-item recommendation (LF-Amazon-131K, LF-Amazon-1M), to Wikipedia articles category/title tagging (LF-WikiSeeAlso-320K, LF-Wikipedia-500K).

Evaluation Protocol We consider two evaluation setups: Extreme Zero-shot Learning of XMC (EZ-XMC) and Few-shot Learning of XMC (FS-XMC). EZ-XMC is a fully unsupervised learning setup where no positive (instance, label) pairs are available. The only available information is the raw text of training instances and the whole label set. FS-XMC is a semi-supervised learning setup where only very few positive (instance, label) pairs in the training set are available. Regardless of the learning procedure, all models are evaluated on the same test set for fair comparison.

We evaluate the models' performance with precision@k (P@k, $k \in \{1, 3, 5\}$) and recall@k (R@k, $k \in \{1, 3, 5, 10, 100\}$), which are two commonly-used evaluation metrics in the XMC literature [RKY19, CJY21].

Baseline Methods For EZ-XMC, we compare our method with the following unsupervised learning algorithms: TF-IDF, XR-Linear, GloVe, SentBERT, MPNet, SimCSE and ICT. Note that SentBERT and MPNet are pre-trained on external multi-task learning datasets with extra supervision. In contrast, SimCSE and ICT are fully unsupervised pre-rained Siamese-Transformers on the specific XMC dataset only.

For FS-XMC, as few-shot (instance, label) pairs are available, we additionally compare fine-tuned MACLR with competitive XMC approaches, including Astec [DSM21], SiameseXML [DAS21], and ZestXML [GBP21]. ZestXML is the leading XMC method that improves performance on few-shot labels. We also take into account SentBERT [RG19] with further fine-tuning to demonstrate the effectiveness of our pre-training procedure.

Table 6.1: Extreme Zero-shot Learning (EZ-XMC) comparison of different methods.

Method	Precision @1	Precision @3	Precision @5	Recall @1	Recall @3	Recall @5	Recall @10	Recall @100
LF-Amazon-131K								
TF-IDF	12.38	11.50	9.14	6.91	18.14	23.21	29.32	45.04
XR-Linear	7.56	7.84	7.30	4.05	12.11	18.32	29.17	40.39
GloVe	3.67	2.78	2.15	2.05	4.33	5.44	7.23	14.17
SentBERT	1.86	1.44	1.14	1.01	2.22	2.88	4.01	10.18
MPNet	13.94	11.41	8.82	7.82	18.08	22.58	27.91	43.39
SimCSE	10.13	8.61	6.69	5.61	13.39	16.84	21.27	35.81
ICT	13.82	11.41	8.90	7.76	18.09	22.80	28.94	47.40
MACLR (ours)	**18.13**	**15.42**	**11.93**	**10.35**	**24.45**	**30.43**	**37.28**	**54.99**
LF-WikiSeeAlso-320K								
TF-IDF	10.71	8.90	7.15	5.92	13.03	16.48	21.60	42.55
XR-Linear	4.73	4.27	3.90	2.23	5.83	8.64	14.18	36.93
GloVe	3.86	2.76	2.21	2.12	4.11	5.22	6.95	15.33
SentBERT	1.71	1.27	1.06	1.08	2.16	2.90	4.17	10.76
MPNet	13.75	11.93	9.58	8.14	17.77	22.21	28.11	45.91
SimCSE	9.03	6.64	5.22	4.99	9.89	12.34	15.93	30.11
ICT	10.76	10.05	8.12	6.12	14.32	18.05	23.01	39.77
MACLR (ours)	**16.31**	**13.53**	**10.78**	**9.71**	**20.39**	**25.37**	**32.05**	**53.83**
LF-Wikipedia-500K								
TF-IDF	20.30	12.98	9.96	7.25	12.91	15.98	20.31	38.16
XR-Linear	10.67	8.77	7.61	3.69	8.58	12.11	19.80	31.02
GloVe	2.19	1.52	1.23	0.85	1.66	2.18	3.10	8.52
SentBERT	0.17	0.15	0.13	0.05	0.13	0.18	0.30	1.29
MPNet	22.46	12.87	9.49	8.74	14.07	16.76	20.64	34.72
SimCSE	14.32	6.84	4.55	4.24	8.03	11.26	14.35	27.68
ICT	17.74	9.67	7.06	7.35	11.60	13.84	17.19	31.08
MACLR (ours)	**28.44**	**17.75**	**13.53**	**10.40**	**18.16**	**22.38**	**28.52**	**50.09**
LF-Amazon-1M								
TF-IDF	7.68	9.20	7.23	5.61	19.30	24.92	31.76	51.79
XR-Linear	5.19	5.48	5.26	3.63	11.30	17.94	31.18	43.79
GloVe	4.05	4.07	3.07	2.91	8.42	10.44	12.90	21.18
SentBERT	2.82	2.87	2.13	2.03	5.91	7.21	8.80	14.22
MPNet	8.29	8.87	6.80	6.04	18.64	23.51	29.35	46.15
SimCSE	3.33	3.69	2.74	2.38	7.66	9.38	11.43	18.54
ICT	8.66	9.26	7.13	6.30	19.45	24.60	30.73	48.42
MACLR (ours)	**9.58**	**10.41**	**8.03**	**7.38**	**22.01**	**27.72**	**34.48**	**55.23**

6.5.2 Zero-Shot Learning

In this section, we focus on extreme zero-shot learning (EZ-XMC), where no real positive (instance, label) pairs are accessible. Table 6.1 presents detailed performance of precision and recall on all four datasets. Our proposed MACLR consistently outperforms all comparing baselines by a large margin on all four datasets. Compared to the leading sparse method TF-IDF, MACLR has an average of 5.3% and 9.1% improvement in Precision@1 and Recall@100, respectively. Compared to the leading model MPNet, MACLR has an average of 3.5% and 10.9% improvement in Precision@1 and Recall@100 respectively.

Speaking of sparse lexical matching approaches, TF-IDF remains a tough-to-beat unsupervised baseline. Specifically, TF-IDF performs better than many BERT variants (e.g., SentBERT, SimCSE, ICT), which is aligned with the finding in recent zero-shot dense retrieval literature [TRR21, ICH22]. It suggests the importance of designing proper self-supervised learning tasks for Transformer models in unsupervised EZ-XMC setup. Note that XR-Linear is based on TF-IDF vectors whereas the noise from pseudo pairs makes it even inferior to the original TF-IDF.

As for pre-trained SentBERT models, on the other hand, only MPNet shows comparable performance with TF-IDF. MPNet remains competitive because it was trained on a large supervised corpus (out-of-domain) to learn semantics between paraphrasing sentences. Thus, MPNet should be viewed as a multi-task learning baseline with extra supervision. However, MACLR is significantly better than MPNet with an average improvement of 3.5% in P@1 and over 10% in R@100. Furthermore, MACLR also outperforms its counterparts which are trained with effective pre-training tasks such as SimCSE and ICT on the target dataset, showing the effectiveness of pre-training strategies like multi-scale adaptive clustering in MACLR. Overall, results in Table 6.1 demonstrates that MACLR is capable to learn informative embeddings and to make useful predictions even with no supervision. We will investigate each component in MACLR in Section 6.5.4 thoroughly.

Table 6.2: Results of FS-XMC where the training subset covers 1% labels.

Method	Precision			Recall				
	@1	@3	@5	@1	@3	@5	@10	@100
LF-Amazon-131K								
XR-Linear	1.53	0.57	0.36	0.67	0.75	0.78	0.81	0.92
Astec	0.94	0.44	0.29	0.55	0.78	0.84	0.91	1.13
SiameseXML	1.45	0.56	0.35	0.84	0.96	1.00	1.03	1.16
ZestXML	10.10	9.19	7.34	5.63	14.46	18.61	23.73	32.69
SentBERT	12.64	9.82	7.80	6.97	15.34	19.74	25.33	43.53
MPNet	14.78	11.55	8.97	8.28	18.24	22.84	28.54	45.89
MACLR (ours)	**18.74**	**16.07**	**12.52**	**10.73**	**25.44**	**31.89**	**39.17**	**57.55**
LF-WikiSeeAlso-320K								
XR-Linear	1.24	0.57	0.37	0.42	0.58	0.63	0.68	0.76
Astec	1.25	0.60	0.41	0.69	0.98	1.11	1.27	1.56
SiameseXML	1.81	0.75	0.48	1.03	1.26	1.33	1.41	1.67
ZestXML	8.74	6.78	5.41	4.68	9.70	12.21	15.73	24.98
SentBERT	16.30	12.62	10.08	9.30	18.92	23.78	30.40	52.92
MPNet	17.14	12.64	9.96	9.97	18.98	23.45	29.67	50.75
MACLR (ours)	**19.09**	**14.57**	**11.53**	**11.39**	**22.34**	**27.63**	**34.81**	**57.92**
LF-Wikipedia-500K								
XR-Linear	2.95	1.19	0.75	0.62	0.74	0.76	0.79	0.84
Astec	2.85	1.16	0.73	1.46	1.75	1.84	1.92	2.08
SiameseXML	2.72	1.15	0.73	1.39	1.73	1.84	1.93	2.09
ZestXML	23.86	14.97	11.31	7.19	13.00	16.03	20.13	29.95
SentBERT	32.09	20.50	15.78	10.94	19.46	24.12	30.94	55.94
MPNet	34.58	22.02	16.86	11.96	21.32	26.30	33.53	57.78
MACLR (ours)	**44.27**	**28.46**	**21.83**	**15.14**	**27.04**	**33.33**	**42.03**	**67.95**
LF-Amazon-1M								
XR-Linear	0.51	0.20	0.12	0.36	0.42	0.43	0.45	0.49
Astec	0.49	0.59	0.12	0.34	0.40	0.42	0.44	0.49
SiameseXML	0.60	0.73	0.15	0.41	0.46	0.48	0.49	0.53
ZestXML	5.07	5.89	4.38	3.68	12.31	15.04	17.80	22.51
SentBERT	6.56	6.93	5.68	4.35	18.29	24.72	28.69	48.52
MPNet	8.87	10.34	7.56	6.78	20.11	26.14	31.98	50.48
MACLR (ours)	**10.37**	**11.23**	**8.58**	**7.57**	**23.55**	**29.60**	**36.71**	**56.44**

Table 6.3: Results of FS-XMC where the training subset covers 5% labels.

Method	Precision			Recall				
	@1	@3	@5	@1	@3	@5	@10	@100
				LF-Amazon-131K				
XR-Linear	5.09	2.09	1.32	2.36	2.86	3.02	3.18	3.74
Astec	3.94	1.92	1.26	2.31	3.34	3.66	4.00	4.96
SiameseXML	5.36	2.23	1.41	3.15	3.89	4.08	4.27	4.82
ZestXML	12.33	10.99	8.71	6.84	17.19	21.97	28.10	46.49
SentBERT	15.47	12.24	9.64	8.63	19.23	24.40	30.82	49.22
MPNet	15.03	11.88	9.28	8.47	18.74	23.69	29.93	48.84
MACLR (ours)	**19.56**	**16.19**	**12.64**	**11.15**	**25.65**	**32.18**	**39.63**	**58.45**
				LF-WikiSeeAlso-320K				
XR-Linear	4.69	2.20	1.46	1.82	2.41	2.63	2.82	3.42
Astec	5.90	2.80	1.86	3.26	4.49	4.95	5.49	6.83
SiameseXML	6.83	3.15	2.06	3.88	5.15	5.56	6.02	7.09
ZestXML	10.06	8.11	6.60	5.33	11.49	14.74	19.57	40.46
SentBERT	18.47	14.19	11.29	10.82	21.55	26.77	33.92	57.02
MPNet	18.59	13.99	11.08	10.89	21.12	26.10	32.82	54.70
MACLR (ours)	**20.99**	**15.57**	**12.26**	**12.59**	**23.94**	**29.41**	**36.78**	**59.81**
				LF-Wikipedia-500K				
XR-Linear	11.80	5.30	3.39	2.76	3.47	3.65	3.82	4.09
Astec	11.23	5.27	3.48	5.46	7.47	8.16	8.90	10.35
SiameseXML	12.44	5.69	3.79	6.05	7.98	8.62	9.22	10.40
ZestXML	27.31	17.31	13.09	8.28	15.13	18.64	23.30	36.50
SentBERT	41.06	26.35	20.25	14.17	25.34	31.32	39.77	66.24
MPNet	42.81	28.07	21.66	14.67	26.81	33.24	42.28	67.76
MACLR (ours)	**47.25**	**30.57**	**23.54**	**16.20**	**29.01**	**35.81**	**45.13**	**71.35**
				LF-Amazon-1M				
XR-Linear	2.11	0.84	0.53	1.45	1.74	1.81	1.88	2.04
Astec	2.22	2.56	0.71	1.54	1.91	2.03	2.16	2.41
SiameseXML	2.60	3.01	1.06	1.81	2.20	2.30	2.41	2.60
ZestXML	7.17	8.35	6.36	5.18	17.49	21.88	26.80	36.51
SentBERT	8.89	10.02	7.93	7.00	21.58	27.35	33.98	54.28
MPNet	9.25	10.41	8.00	7.11	21.87	27.64	34.61	54.72
MACLR (ours)	**10.60**	**11.47**	**8.80**	**7.89**	**24.14**	**30.44**	**37.95**	**58.45**

6.5.3 Few-Shot Learning

We further conduct few-shot learning (FS-XMC) experiments in which different learning algorithms can access a limited number of positive (instance, label) pairs. To simulate the scenario of few-shot learning, we first manually sample a small ratio of labels, then collect all their positive instances from the training set as the final subset of positive (instance, label) pairs for model training. Results of FS-XMC methods fine-tuned with 1% and 5% labels are shown in Tables 6.2 and 6.3 respectively.

Our proposed MACLR outperforms all other baselines significantly, including variants of Siamese-Transformer models (e.g., SentBERT, MPNet) and major competitive XMC methods (e.g., XR-Linear, Astec and SiameseXML), on all four datasets.

Note that SiameseXML is the state-of-the-art XMC method under the full supervision setup of XMC. Here, we again witness that existing XMC methods heavily rely on the supervision level as well as the full-coverage of label space for test set. MACLR, in contrast, still performs robustly under FS-XMC, which enjoy larger applicability to emerging domains with many cold-start labels.

Crucially, even ZestXML tailored to address the challenging scenario of unseen labels cannot match the performance of MACLR. In particular, when focusing on the few-shot scenario with only 1% sampled labels, MACLR achieves 18.74% in P@1, improving the performance of Astec with 0.94% and ZestXML with 10.10% significantly. Besides, MACLR outperforms all Sentence-BERT counterparts, validating the effectiveness of our pre-training procedure. As to fine-tuning on the subset with 5% labels, performance of all methods are improved as expected with more supervision. The relative rank among these methods remains the same, with MACLR still performing the best in terms of precision and recall on all four datasets.

6.5.4 Ablation Study

In this part, we conduct an ablation study to investigate each component in our pre-training procedure, including multi-scale adaptive clustering, label regularization, and self-training with pseudo positive pairs constructed from the encoder or TF-IDF. We add a component once a time on LF-Amazon-131K to observe its independent influence on the model performance. Table 4.4 presents detailed performance on seven different configurations.

Table 6.4: Ablation study on LF-Amazon-131K.

Index	Ablation Configuration				Precision			Recall				
	MAC [*]	LR [*]	$\mathcal{E}$ [†]	TFIDF [†]	@1	@3	@5	@1	@3	@5	@10	@100
1	No	No	No	No	13.82	11.41	8.90	7.76	18.09	22.80	28.94	47.40
2	Yes	No	No	No	15.79	13.16	10.22	8.85	20.90	26.27	32.61	49.83
3	No	Yes	No	No	16.02	13.29	10.28	9.04	21.27	26.51	32.97	50.34
4	Yes	Yes	No	No	16.37	13.71	10.65	9.29	21.63	27.03	33.93	51.45
5	Yes	Yes	Yes	No	17.01	14.75	11.41	9.72	23.33	29.04	35.20	53.55
6	Yes	Yes	No	Yes	16.51	14.12	10.92	9.52	22.43	28.02	34.64	52.78
7	Yes	Yes	Yes	Yes	18.13	15.42	11.93	10.35	24.45	30.43	37.28	54.99

[*] MAC represents adaptive clustering while LR stands for label regularization.
[†] Pseudo positive pairs are constructed from $\mathcal{E}$ or TFIDF.

For two techniques multi-scale adaptive clustering and label regularization during the Stage I, they can improve the performance of the encoder separately, as shown in the performance gain of the index 2 and 3 over the index 1. When combined, they can further improve the accuracy of the model, from 8.90% to 10.65% in $P@5$ and from 47.40% to 51.45% in $R@100$. As to the second stage, we explore the impact of self-training with pseudo positive pairs either from the encoder itself or TF-IDF. We can see from Table 6.4 that pairs from both $\mathcal{E}$ and TF-IDF contribute to the precision and recall gain over the index 5. It further validates that the encoder and TF-IDF provides complementary perspective when constructing pseudo positive pairs.

6.6 Conclusions

This work is the first to investigate the problem of Extreme zero-shot XMC without any supervision. We develop a two-stage pre-training procedure MACLR to train a Sentence-BERT style encoder on pseudo (context, title) pairs constructed from raw text. We demonstrate that techniques including multi-scale adaptive clustering, label regularization and self-training contribute to the performance gain of the pre-trained encoder. In particular, MACLR outperforms all unsupervised baselines significantly when there are no (instance, label) pairs provided. It also offers leading accuracy in both precision and recall after fine-tuning on a limited number of paired data. One limitation is relative low accuracy of top candidates and a future direction could be adding a ranker model after the encoder to improve performance on head labels.

CHAPTER 7

Structured Video-Language Modeling with Temporal Grouping and Spatial Grounding

The idea of pre-training followed by downstream adaptation can be extended to video-language learning, where web-craweled noisy data are easy to obtain. To improve the utilization of the pre-training dataset, we exploit fine-grained structures in this chapter to make latent representations more expressive and generalizable.

7.1 Introduction

Videos are composed of groups of pixels spanning spatially and temporally. Semantically related groups of pixels form the objects in the visual scenes and their changes through space and time vividly show the action and interactions of physical world. Scene switching further complicates the video story line and finally depicts attractive video stories. The similar structures also appear in the paragraphs when they come to describe the videos. Captions are built from the basic grammar components such as nouns and verbs, and sentences are concatenated to describe complex scenes.

Modern video-language models (VLMs), however, mostly neglect the fine-grained structures of such video-text pairs during the development. Video-language pre-training typically follows the pipeline: (1) encoding video and text pairs into latent representations, (2) modality fusion, and (3) pre-training on specific objectives. Existing methods typically optimize these three components in the pre-training pipeline by designing expressive encoders [BNV21, LLL22, GGL22a, NSS22, MJW23], fusing two modalities via a cross-encoder [GGL22a, LLL22, LLZ21, LCC20, LJS20, XGH21a, ZY20], or adopting a combination of various pre-training tasks such as contrastive learning and masked modeling [LLL22, GGL22a, FLG21, ZLL22, CYW22, GGL22b]. While these modifications benefit the pre-trained model, their lack of local discriminative modeling poses challenges for VLMs to further understand complex videos.

It has been shown that most video-language pre-training methods merely perform well on learning holistic representations to match a ⟨*video, caption*⟩ pair while neglect fine-grained information such as region-object correspondences, or scene/action changes along the time in a video [AYQ21, BNV21, LLZ21, LCC20, LJS20, MZA19, XGH21b, NSS22]. However, such regional or temporal fine-grained information has been demonstrated to play a vital role in localization and reasoning tasks [LLL22, GGL22a, ZYW22, MJW23, YQC22]. Motivated by aforementioned observations, we revive the strong connectivity between basic components of video clips and languages in self-supervised video-language pre-training. We approach the video-language pre-training task from a different perspective with a focus on exploiting spatiotemporally fine-grained structures.

In this work, we integrate structured video-language interactions into the pre-training stage and propose a novel framework, **S**tructured **Vi**deo-**L**anguage **M**odeling (S-ViLM), with temporal grouping and spatial grounding. S-ViLM encourages instance-level video-caption alignment, fine-grained region-object alignment, and learns temporal-aware video representations, simultaneously. As shown in Figure 3.1, S-ViLM consists of three training objectives: inter-clip spatial grounding, intra-clip temporal grouping, and global contrastive learning. Given a video-caption pair as the input, a classical dual-encoder model is leveraged to extract the representation for each modality, respectively. Videos are pre-processed with the cut-and-paste operation, inspired by [ZYW22, YHO19], i.e., pasting one clip in a video onto the other background video, to explicitly introduce temporal scene changes. We further adopt grouping blocks [XDL22, YWQ22] to aggregate semantically similar video patches to represent regions without off-the-shelf detectors via a set of group tokens shared among all videos. In *inter-clip spatial grounding*, we align grouped video tokens with objects represented by nouns in the caption by minimizing our grounding loss. In *intra-clip temporal grouping*, we improve features temporal granularity by distinguishing foreground and background representations within one clip. Finally, the model is trained by a global video-caption contrastive loss to match instance-level video-caption pairs. We evaluate our proposed method comprehensively on four representative tasks, including text-video retrieval, video question answering, video action recognition, and temporal action localization. Our strong experimental results demonstrate that exploiting fine-grained video-text structures during pre-training effectively improves VLM's video understanding and reasoning capabilities.

Our key contributions are summarized as follows:

- We propose S-ViLM, a dual-encoder video-language modeling framework, utilizing structured video-caption interactions for more expressive spatiotemporal features.
- We leverage a cut-and-paste operation to introduce scene changes into videos during pre-training, and propose an intra-clip grouping module to learn more temporal-aware features.
- We design an inter-clip spatial grounding module to capture fine-grained correspondences by aligning objects from the caption and regions from the video in a self-supervised manner.
- Experimental results have demonstrated the effectiveness of S-ViLM on four downstream tasks, including text-video retrieval, video question answering, video action recognition, and temporal action localization. For example, S-ViLM outperforms SOTA by 3% in R@1 in zero-shot video-text retrieval on MSR-VTT and 5% in accuracy in action recognition on UCF101, showing its advantages over both multi-modal and single-modal tasks.

7.2 Related Work

Video-language pre-training. Video-language pre-training is an emerging research area that aims to develop machine learning models capable of jointly understanding visual and textual content. Representations learned from large scale noisy datasets such as HowTo100M [MZA19], WebVid [BNV21], and VideoCC [NSS22] have demonstrated great potentials in adapting to downstream tasks, including text-video retrieval, video question answering, and video captioning. Elaborately designed pre-training objectives ranging from generative [CLY20, FLG21, LYY19, LXX22] to discriminative [BNV21, LLZ21, AYQ21, SXS22, LLL22, GGL22a, MXS22, WYC23, WGY23, WSC23] have been proposed, among which contrastive learning is prevalent and widely adopted to attract paired video-caption instances and repelling unpaired ones. However, their primary focus is still on learning holistic global representations to align instance-level $\langle$*video, caption*$\rangle$ pairs. Recently, some approaches have been proposed to leverage finer-grained information such as nouns/verb phrases from a caption. ALPRO [LLL22] extracts pseudo entity labels by feeding noun prompts into a frozen model and use contrastive objective to align cropped visual regions and the corresponding textual labels. In [GGL22a], MCQ recovers randomly masked noun/verb tokens via resorting to global video features, which implicitly improves text entity association in visual encoding. LAVILA [ZMK23] constructed temporally dense captions by automatic annotation from large language models to describe activities more comprehensively. In addition, TemPVL [MJW23] enables temporal and semantic alignment such that the trained model can accurately perceive temporal boundaries in videos given the text description. Despite these efforts, correspondences between visual regions and objects from noun concepts in captions and temporal scene shifts in a video are still neglected and not modeled explicitly in existing video-language pre-training methods. In this work, we propose two novel designs, spatial grounding and temporal grouping, to leverage fine-grained information in the pre-training stage.

Vision language grounding. The goal of visual grounding (VG) is to locate the most relevant object or region in a visual input based on a natural language query [FGI15, RRH16, FPY16, GGC22, GVC20]. Recently, visual grounding has been adapted to pre-training tasks for open-vocabulary image segmentation [GGC22, XDL22]. OpenSeg [GGC22], for example, semantically aligns a caption with extracted image regions via a grounding loss. Moreover, without the off-the-shelf object detectors, GroupViT [XDL22] learns to group together semantic regions from text supervision by contrastive learning. Note that visual grounding is mostly discussed in the image domain and its success motivates us to extend visual-semantic alignment to video-language pre-training. To achieve this, we integrate a novel spatial grounding module in our framework to promote visual and textual entity correspondences in a self-supervised manner.

Video temporal modeling. In contrast to images, videos contain a sequence of dynamic frames and how to model temporal information is critical in video understanding [FFM19, BWT21, TBF14, AGG21, ZYW22, QLY22]. Specifically, TSP [AGG21] learns temporal information via predicting clips inside or outside the action with substantial annotations. PAL [ZYW22] aligns features of pasted pseudo action regions from two synthetic videos. BSP [XPE21] introduces a novel boundary-sensitive pretext task via classifying the boundary types of synthetic videos. These techniques are elaborately designed for training models on long videos such as movies or TV dramas, which contains natural scene changes. However, few of them have been considered in video-language pre-training since the majority of video-language datasets contains short videos with repeated frames and are lacking in temporal differences. Instead, we develop a temporal grouping method to learn temporal-aware clip features in a self-supervised manner. We show that features extracted from explicitly temporal modeling achieve significant improvements in not only temporal action localization tasks, but also coarse-grained reasoning and understanding tasks such as video question answering and video action recognition.

7.3 Method

7.3.1 Overview

The framework of S-ViLM is presented in Figure 7.1. We adopt the dual encoder architecture for video-language pre-training, and there are three primary objectives used in the pre-training stage: (1) inter-clip spatial grounding, (2) intra-clip temporal grouping, and (3) global contrastive learning.

As shown in Figure 7.1, temporal changes are first artificially introduced into training examples through cut-and-paste. Then the pre-processed video together with learnable group tokens are fed into the video encoder. Specifically, group tokens aggregate semantically similar video tokens via grouping blocks and are then aligned with object concepts by spatial grounding. It promotes region-object groundingness, which indicates the alignment between a region in the video and an object in the caption, e.g., as illustrated in Inter-clip Spatial Grounding in Figure 1, the red region corresponds exactly to the word "pins" in red. In contrast to previous methods where regions are extracted with pre-trained object detectors [CGW22, LLL22, YSG21], these learnable group tokens can cluster and organize semantically similar regions in a self-supervised manner, which is more effective and reduces the artifacts of any detectors. For the language branch, the original captions are tokenized into a sequence of text tokens, which are then fed into a text encoder to extract the corresponding representation from the preceding `[CLS]` token. Noun tokens representing objects are extracted in the same way given a set of prompts.

To promote temporal awareness, we use masks derived from the cut-and-paste operations as the ground-truth for temporal grouping. Furthermore, we model the interaction between region features and noun tokens using inter-clip spatial grounding loss. Finally, a global contrastive loss is computed between the video and the caption representations to match the instance-level $\langle$*video, caption*$\rangle$ pair.

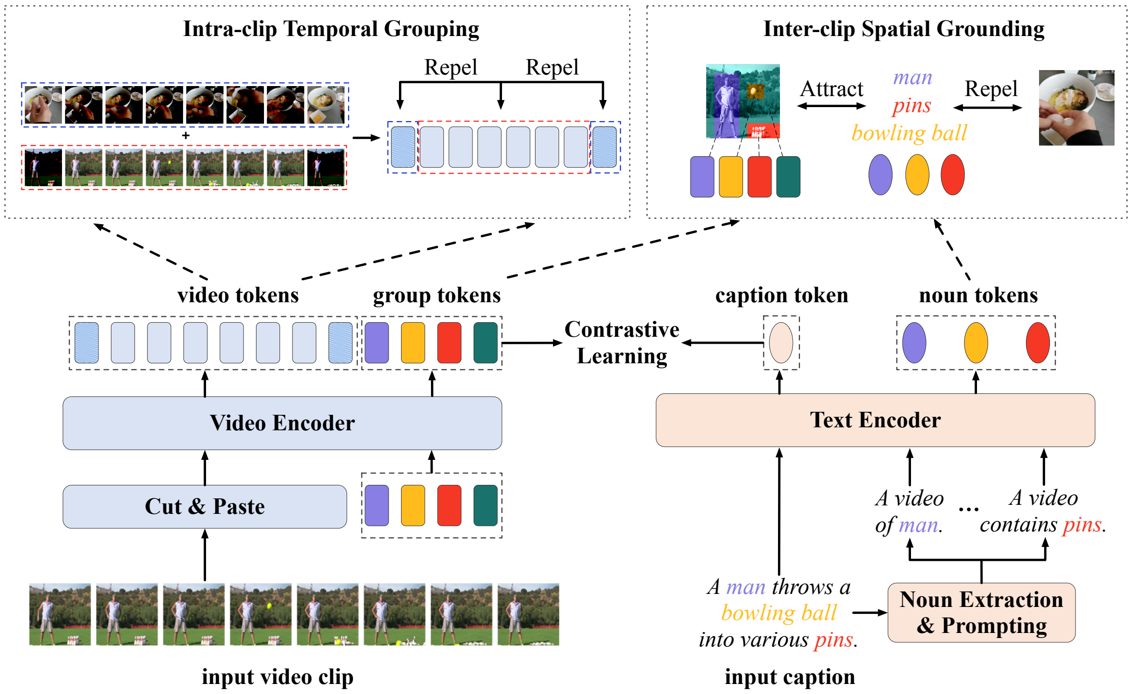


Figure 7.1: Illustration of S-ViLM pre-training. Three training objectives promote structured video-language interaction: (1) temporal grouping learns temporal-aware features by distinguishing clips from background or foreground; (2) spatial grounding focuses on local correspondences between regions and objects; (3) global contrastive learning matches instance-level $\langle$*video, caption*$\rangle$ pairs.

7.3.2 Intra-clip Temporal Grouping with Cut-and-Paste

Commonly-used video-language pre-training data usually consist of short video clips with repetitive scenes. To simulate scene shifts, we design a cut-and-paste operation inspired from image augmentations [YHO19, ZYW22] to introduce temporal changes manually to further improve video representations.

Given a target video v_i with T frames as the foreground and a randomly sampled video v_{p_i} with the index p_i as the background from the same batch of size B, we divide each video into $N_t = T/t$ clips with the temporal window size t. We then sample the start and end clip indices s and e from $(0, N_t)$, and paste the corresponding region from v_i into the background video v_{p_i} to form a blended video $\hat{v}_i$. For the clip sampling procedure, we first uniformly sample the duration of the foreground video d from $[N_t/2, N_t)$ to guarantee it is the majority of the blended video. Then we sample the start index s from $[0, N_t - d)$, and the end index e was computed naturally as $e = s + d$. We included this detail in our latest version. We define the foreground-background mask as $m_i \in \mathbb{R}^{N_t} = \{\mathbf{1}(j \in [s, e]) | j \in [0, N_t)\}$, where $\mathbf{1}(\cdot)$ is the indicator function. This operation is illustrated in Figure 7.1.

A video is first flattened into N non-overlapping voxels. After projected by a linear layer, these voxel tokens are fed into the transformer encoder to obtain transformed tokens $z_i^v \in \mathbb{R}^{N \times d}$, where d is the feature dimension. To obtain clip-level representations $z_i^{\text{clip}} \in R^{N_t \times d}$, we average-pool over z_i^v along the spatial dimension after recovering the feature map's 3D shape. Two cluster centers, z_i^b for the background and z_i^f for the foreground, are further computed by averaging features from z_i^v on the corresponding position based on the mask m_i. To assign each clip to either background or foreground, we compute a_i via cosine similarity with an element-wise softmax function applied on the last dimension, where $\langle \cdot, \cdot \rangle$ is cosine similarity and τ is the temperature to scale logits:

$$a_i = \text{Softmax}(\langle z_i^{\text{clip}}, [z_i^b; z_i^f]^T \rangle / \tau) \in \mathbb{R}^{N_t \times 2}. \tag{7.1}$$

Finally, the temporal grouping loss can be computed within a batch between a_i and the ground-truth one-hot masking m_i using mean squared error as

$$\mathcal{L}_{\text{t}} = \frac{1}{B} \sum_{i}^{B} \ell_{\text{BCE}}(a_i, \text{One-hot}(m_i)). \tag{7.2}$$

Note that we have also tried the binary cross entropy loss which performs comparably to MSE. Thus, we select a relatively simple MSE loss for temporal grouping.

7.3.3 Inter-clip Spatial Grounding with Group Tokens

Observing the correspondences between visual regions in a video and noun phrases (objects) in a caption, we model such fine-grained alignment for more expressive encoders.

In practice, it is infeasible to pool tokens of interest as cluster centers since we do not have ground-truth segmentation. Thus, we adopt M learnable group tokens to cluster semantically similar regions in a self-supervised manner. Note that group tokens are randomly initialized and shared among different videos. Multiple grouping blocks are placed at different layers of the video encoder to update group tokens progressively. The final group tokens denoted as $\mathcal{G} = \{g_i^m\}_{m=1}^M$ aggregate semantically similar voxels and represent different regions in the video v_i. Compared with using off-the-shelf region proposal networks, our design of token grouping is more computationally efficient, and can be adapted to the pre-training dataset without region annotations in a self-supervised manner dynamically and flexibly. For each caption c_i of a video, we extract K noun phrases using noun chunking in spaCy[1] and prompt each of them with a set of handcrafted sentence templates, e.g., "*A photo of a {noun}*". Such prompted noun phrases are fed into the text encoder to extract noun tokens $\{n_i^k\}_{k=1}^K$.

We denote softmax on a vector $\mathbf{x}$ at the i-th element as: $\sigma(\mathbf{x})_i = \frac{\exp(x_i)/\tau}{\sum_j \exp(x_j)/\tau}$, where τ is the temperature to scale logits. The similarity of all group tokens $\mathcal{G}$ with respect to a noun token n^k is defined as $s(\mathcal{G}, n^k) = [\langle g^1, n^k\rangle, \ldots, \langle g^M, n^k\rangle] \in \mathbb{R}^M$, where $\langle \cdot, \cdot \rangle$ is the cosine similarity. As the ground-truth correspondences between regions and nouns are inaccessible, we compute the grounding similarity between all group and noun tokens by:

$$G(v, c) = \frac{1}{K}\sum_{k=1}^{K} \left\langle n^k, \sum_{m=1}^{M} \sigma\left(s(\mathcal{G}, n^k)\right)_m \cdot g^m \right\rangle. \tag{7.3}$$

$G(v, c)$ encourages each noun to be grounded to one or a few regions and avoids penalizing regions that cannot find any relevant nouns.

Similarity scores are computed as: $G(\mathcal{V}, c_i) = [G(v_1, c_i), \ldots, G(v_B, c_i)] \in \mathbb{R}^B$ and $G(v_i, \mathcal{C}) = [G(v_i, c_1), \ldots, G(v_i, c_B)] \in \mathbb{R}^B$, where $\mathcal{V} = \{v_i\}_{i=1}^B$ and $\mathcal{C} = \{c_i\}_{i=1}^B$ denote the set of videos and captions in a batch respectively and B is the batch size. Inter-clip spatial ground-

[1]`https://spacy.io/`

ing loss $\mathcal{L}_{\mathrm{g}}$ is then defined to enable nouns to be matched with regions for each positive $\langle$*video, caption*$\rangle$ pair: $\mathcal{L}_{\mathrm{g}} = \mathcal{L}_{\mathrm{g}}^{v\to c} + \mathcal{L}_{\mathrm{g}}^{c\to v}$ consists of a video-to-caption grounding loss and a caption-to-video grounding loss

$$\mathcal{L}_{\mathrm{g}}^{v\to c} = -\frac{1}{B}\sum_{i=1}^{B}\log\sigma\left(G\left(v_i,\mathcal{C}\right)\right)_i, \quad \mathcal{L}_{\mathrm{g}}^{c\to v} = -\frac{1}{B}\sum_{i=1}^{B}\log\sigma\left(G\left(\mathcal{V},c_i\right)\right)_i. \tag{7.4}$$

Recall that the cut-and-paste operation indicates that $\hat{v}_i$ has another positive caption c_{p_i} besides its original c_i, and the loss weights of positive indices are $W^v \in \mathbb{R}^{B\times B} = \{w_{i,j}^v\}$ which satisfy

$$w_{i,j}^v = \begin{cases} \beta_i, & j = i \\ 1-\beta_i, & j = p_i \\ 0, & \text{otherwise} \end{cases}, \tag{7.5}$$

where $\beta_i = (e-s)/N_t$ is the ratio of the foreground in the cut-and-paste video $\hat{v}_i$. From the perspective of captions, we can obtain $W^c = (W^v)^\top$. We can derive the augmented grounding loss $\mathcal{L}_{\mathrm{g}}$ with the video-to-caption loss and and the caption-to-video loss:

$$\mathcal{L}_{\mathrm{g}}^{v\to c} = -\tfrac{1}{B}\textstyle\sum_{i=1}^{B}\sum_{j=1}^{B} w_{i,j}^v \log\sigma\left(G\left(\hat{v}_i,\mathcal{C}\right)\right)_j, \quad \mathcal{L}_{\mathrm{g}}^{c\to v} = -\tfrac{1}{B}\textstyle\sum_{i=1}^{B}\sum_{j=1}^{B} w_{i,j}^c \log\sigma\left(G\left(\hat{\mathcal{V}},c_i\right)\right)_j. \tag{7.6}$$

7.3.4 Overall Pre-training Objective

We include a global contrastive learning objective for instance-level alignment. f_i^v, the video representation of $\hat{v}_i$, is extracted from average-pooled group tokens and f_i^c, the caption representation c_i, is computed from the `[CLS]` token of the original caption. Instance similarity scores are defined as: $s(\mathcal{V},c_i) = [\langle f_1^v, f_i^c\rangle, \ldots, \langle f_B^v, f_i^c\rangle] \in \mathbb{R}^B$ and $s(\hat{v}_i,\mathcal{C}) = [\langle f_i^v, f_1^c\rangle, \ldots, \langle f_i^v, f_B^c\rangle] \in \mathbb{R}^B$. A global contrastive loss is defined as $\mathcal{L}_{\text{contrast}} = \mathcal{L}_{\text{contrast}}^{v\to c} +$

$\mathcal{L}^{c\to v}_{\text{contrast}}$, a combination of the video-to-caption and the caption-to-video views:

$$\mathcal{L}^{v\to c}_{\text{contrast}} = -\tfrac{1}{B}\sum_{i=1}^{B}\sum_{j=1}^{B} w^{v}_{i,j}\log\sigma(s(\hat{v}_i,\mathcal{C}))_j, \quad \mathcal{L}^{c\to v}_{\text{contrast}} = -\tfrac{1}{B}\sum_{i=1}^{B}\sum_{j=1}^{B} w^{c}_{i,j}\log\sigma(s(\mathcal{V},c_i))_j. \quad (7.7)$$

The overall pre-training objective is a weighted sum of grouping loss, grounding loss, and global contrastive loss: $\mathcal{L} = \omega_1\mathcal{L}_{\text{t}} + \omega_2\mathcal{L}_{\text{g}} + \omega_3\mathcal{L}_{\text{contrast}}$. We set three weights, ω_1, ω_2, and ω_3, to be equal to one in our experiments for simplicity.

7.4 Experiments

7.4.1 Downstream Tasks

Text-video retrieval. We adopt the widely used text-video retrieval benchmark MSR-VTT [XMY16a] for evaluation. It consists of 10K YouTube video clips with 200K captions. We conduct experiments in both zero-shot and fine-tuning settings. For fine-tuning setup, we follow [BNV21] and [GGL22a], and train and test the model on the split of 9K and 1K videos.

Video question answering (VQA). We consider open-ended VQA settings with two representative datasets: (1) MSRVTT-QA [XZX17] with 1,500 answer candidates and (2) MSVD-QA [XZX17] with 2,423 answer candidates.

Video action recognition. We select HMDB51 [KJG11] containing 6,766 videos with 51 categories and UCF101 [SZS12] containing 13,320 videos with 101 categories. Both linear probing and fine-tuning the whole model are explored.

Temporal action localization (TAL). TAL aims at predicting the temporal extent and the labels of action instances. We evaluate the performance on ActivityNet [HEG15], an action understanding dataset of 19,994 temporally annotated untrimmed videos with 200 action categories.

7.4.2 Implementation Details

Input. Following [NSS22], we sample 32 frames for each video and resize them into 224×224 with the same augmentations. Each caption is tokenized into 32 tokens including `[CLS]`. $K = 2$ noun phrases are extracted for each caption and then prompted with a set of prompt templates such as *"It is a video of {noun}"*.

Model architecture. We use a 12-layer ViT-base model with the patch size of $2 \times 16 \times 16$ as the video encoder and initialize it with weights pre-trained on Kinetics-400. We adopt 32 learnable group tokens and 3 grouping blocks featuring K-means attention [XDL22, YWQ22]. Grouping blocks are inserted at the 6th, 9th and last layers of the video encoder [XDL22, YWQ22]. The text encoder is initialized from the pre-trained BERT-base model. Representations are projected into the common space with the dimension of 256.

Pre-training datasets. We pre-train S-ViLM with the VideoCC [NSS22] dataset, which contains about 3.3M video-caption pairs. We also include ActivityNet-Caption [KHR17] with 20K well-aligned pairs into the pre-training corpus. We note the commonly-used WebVid [BNV21] is unavailable to us due to the restricted data access policy. To illustrate the effectiveness of our proposed method and how the pre-training datasets contribute to final results, we designed fair studies on dataset impacts. Details could be found in Section 7.4.3.5.

Pre-training and fine-tuning setups. We implement S-ViLM in JAX and train all models on TPU accelerators. During pre-training, SGD with momentum 0.9 and initial learning rate 0.1 is used for optimization. We train S-ViLM for 10 epochs with a batch size 1024 and adopt a cosine learning rate decay schedule with a warmup ratio 0.05. It takes about one day for the whole pre-training stage. In terms of fine-tuning, different tasks are trained independently with their own set of hyperparameters on the target dataset. For temporal action localization, we fix weights of the pre-trained video encoder and its grouping blocks to extract video features, which are then evaluated by G-TAD [XZR20], a

Table 7.1: Zero-shot (top) and fine-tuning evaluation (bottom) of text-video retrieval on MSR-VTT test set with 1K videos. **Higher** R@k and **lower** MedR (Median Rank) indicate better performance.

Method	Video Encoder Input	PT Dataset	#Pairs PT	R@1	R@5	R@10	MedR
MIL-NCE [MZA19]	Raw Videos	HowTo100M	120M	9.9	24.0	32.4	29.6
VATT [AYQ21]	Raw Videos	HowTo100M, AudioSet	138M	-	-	29.7	49.0
VideoCLIP [XGH21b]	S3D	HowTo100M	110M	10.4	22.2	30.0	-
SupportSet [PHA20]	R(2+1)D-34	HowTo100M	120M	12.7	27.5	36.2	24.0
Frozen [BNV21]	Raw Videos	CC3M, WebVid-2M	5.5M	18.7	39.5	51.6	10.0
AVLnet [RBH21]	ResNeXt-101	HowTo100M	120M	19.6	40.8	50.7	9.0
DemoVLP [CGW22]	Raw Videos	CC3M, WebVid-2M	5.5M	24.0	44.0	52.6	8.0
ALPRO [LLL22]	Raw Videos	CC3M, WebVid-2M	5.5M	24.1	44.7	55.4	8.0
MCQ [GGL22a]	Raw Videos	CC3M, WebVid-2M	5.5M	26.0	46.4	56.4	7.0
VCC [NSS22]	Raw Videos	VideoCC	3.3M	18.9	37.5	47.1	-
S-ViLM	Raw Videos	VideoCC, ActivityNet	3.3M	**28.6**	**53.6**	**65.1**	**5.0**
UniVL [LJS20]	S3D	HowTo100M	110M	21.2	49.6	63.1	6.0
MMT [GSA20]	S3D	HowTo100M	120M	26.6	57.1	69.6	4.0
ClipBERT [LLZ21]	Raw Videos	COCO, VisGenome	5.6M	22.0	46.8	59.9	6.0
AVLnet [RBH21]	ResNeXt-101	HowTo100M	120M	27.1	55.6	66.6	4.0
SupportSet [PHA20]	R(2+1)D-34	HowTo100M	120M	30.1	58.5	69.3	3.0
VideoCLIP [XGH21b]	S3D	HowTo100M	110M	30.9	55.4	66.8	-
Frozen [BNV21]	Raw Videos	CC3M, WebVid-2M	5.5M	31.0	59.5	70.5	3.0
DemoVLP [CGW22]	Raw Videos	CC3M, WebVid-2M	5.5M	36.0	61.0	71.8	3.0
ALPRO [LLL22]	Raw Videos	CC3M, WebVid-2M	5.5M	33.9	60.7	73.2	3.0
MCQ [GGL22a]	Raw Videos	CC3M, WebVid-2M	5.5M	37.6	64.8	75.1	3.0
VIOLETv2 [FLG23]	Raw Videos	CC3M, WebVid-2M	5.5M	37.2	64.8	75.8	-
All-in-One [WGY23]	Raw Videos	HowTo100M, WebVid-2M	112M	37.1	**66.7**	75.9	-
VCC [NSS22]	Raw Videos	VideoCC	3.3M	35.0	63.1	75.1	-
S-ViLM	Raw Videos	VideoCC, ActivityNet	3.3M	**38.4**	65.7	**76.3**	**2.0**

commonly used method for TAL.

7.4.3 Evaluation Results

7.4.3.1 Text-Video Retrieval

We evaluate S-ViLM for the text-video retrieval task on MSR-VTT under both zero-shot and fine-tuning settings, and compare it with existing prevalent methods in Table 7.1. S-ViLM outperforms other methods significantly for zero-shot evaluation with R@10 of 65.1, yielding approximately 9% improvement over the best-performing baseline MCQ. The superior results demonstrate that our pre-trained model builds up a good alignment between video and language and generalizes well to unseen datasets. S-ViLM also achieves performance gain when the model is fine-tuned on the target MSR-VTT dataset, which fur-

Table 7.2: Accuracy (%) of Video Question Answering on MSRVTT-QA and MSVD-QA.

Method	PT Dataset	MSRVTT-QA	MSVD-QA
HGA [JH20]	-	35.5	34.7
QUEST [JCL20]	-	34.6	36.1
HCRN [LLV20]	-	35.6	36.1
ClipBERT [LLZ21]	COCO, VG	37.4	-
SSML [ABR21]	HowTo100M	35.1	35.1
CoMVT [SNS21]	HowTo100M	39.5	42.6
DemoVLP [CGW22]	CC3M, WebVid-2M	38.3	39.5
ALPRO [LLL22]	CC3M, WebVid-2M	42.1	45.9
S-ViLM	VideoCC, ActivityNet	**43.5**	**46.4**

ther validates advantages of the pre-trained model. Note that S-ViLM performs favorably against existing methods despite the much smaller size of the pre-training data used in S-ViLM than those in baselines, such as HowTo100M and WebVid-2M.

7.4.3.2 Video Question Answering

VQA results on two open-ended datasets are shown in Table 7.2. To enable S-ViLM to deal with the VQA task, we add a fusion head adapted from BUTD [AHB18] by integrating video and text features with simple linear layers. Then a classifier is inserted after the fusion module to perform question answering as a classification problem. Compared with previous methods which leverage particular architectures for VQA or include a complicated fusion encoder, S-ViLM is the most efficient and flexible for various vision-language tasks. S-ViLM achieves better performance than competing methods with the accuracy of 43.5% (+1.4%) and 46.4% (+0.5%) on MSRVTT-QA and MSVD-QA, respectively.

7.4.3.3 Video Action Recognition

For video action recognition, we only keep the video encoder together with its grouping blocks to extract single-modality video representations for evaluation. Two evaluation settings are considered: (1) linear probing where the backbone encoder is frozen and only the last linear classifier is trained and (2) end-to-end fine-tuning where both the backbone and

Table 7.3: Experiments of action recognition on UCF101 and HMDB51 with linear evaluation (Lin) and fully fine-tuning evaluation (FT).

Method	**Modal**	**UCF101**		**HMDB51**	
		Lin	FT	Lin	FT
CoCLR [HXZ20]	OF	77.8	90.6	52.4	62.9
MVCGC [HDL21]	MV	78.0	90.8	53.0	63.4
XDC_R [AMK20]	A	80.7	88.8	49.9	61.2
XDC_K [AMK20]	A	85.3	91.5	56.0	63.1
MIL-NCE [MZA19]	T	83.4	89.1	54.8	59.2
Frozen [BNV21]	T	87.8	89.8	61.3	66.3
VATT [AYQ21]	A, T	89.2	-	63.3	-
ELO [PAR20]	A, OF	-	93.8	64.5	67.4
MMV [ARS20]	A	77.1	-	53.6	-
MMV [ARS20]	T	86.8	-	55.1	-
MMV [ARS20]	A, T	91.8	95.2	67.1	75.0
MCQ [GGL22a]	T	89.1	92.3	65.8	69.8
S-ViLM	T	**94.8**	**96.5**	**70.0**	**76.9**

Table 7.4: Comparison to SOTA methods on temporal action localization (**TAL**).

Method	**TAL Task (G-TAD)**			
	mAP@0.5	@0.75	@0.95	Avg
CoCLR [HXZ20]	47.9	32.3	7.3	31.9
XDC [AMK20]	48.4	32.6	7.6	32.3
MoCo-v2 [CFG20]	46.6	30.7	6.3	30.3
VideoMoCo [PSY21]	47.8	32.1	7.0	31.7
RSPNet [CHH21]	47.1	31.2	7.1	30.9
AoT [WLZ18]	44.1	28.9	5.9	28.8
SpeedNet [BEL20]	44.5	29.5	6.1	29.4
PAL [ZYW22]	50.7	35.5	8.7	34.6
TAC [XZR20]	48.5	32.9	7.2	32.5
BSP [XPE21]	50.9	35.6	8.0	34.8
LoFi [XPZ21]	50.4	35.4	8.9	34.4
TSP [AGG21]	51.3	**37.1**	9.3	**35.8**
S-ViLM	**51.7**	36.4	**9.7**	35.6

the classifier are trained. Top-1 accuracy on UCF101 and HMDB51 is reported in Table 7.3. We observe that in linear probing, S-ViLM outperforms other baselines, with 3.0% and 2.9% higher than current SOTA, MMV that leverages audio and text modalities in addition on UCF101 and HMDB51. S-ViLM also achieves consistently superior performance under the fine-tuning evaluation. Outstanding performance of S-ViLM demonstrates that leveraging fine-grained video language structures during pre-training contributes to meaningful video representations. This aligns with our intuition because finer-grained video-text alignment improves video understanding.

7.4.3.4 Temporal Action Localization

We report the mean average precision (mAP) under different temporal Intersection over Union (tIoU) thresholds on ActivityNet in Table 7.4. For temporal action localization, the model is pre-trained on HowTo100M only, which is observed to be beneficial to TAL compared with VideoCC + ActivityNet (see the ablation study below). We directly use pre-trained models to extract video features as the input to G-TAD and do not further train the encoder. S-ViLM consistently exceeds other self-supervised competitors and even fully supervised approaches such as LoFi and BSP. This observation again consolidates the con-

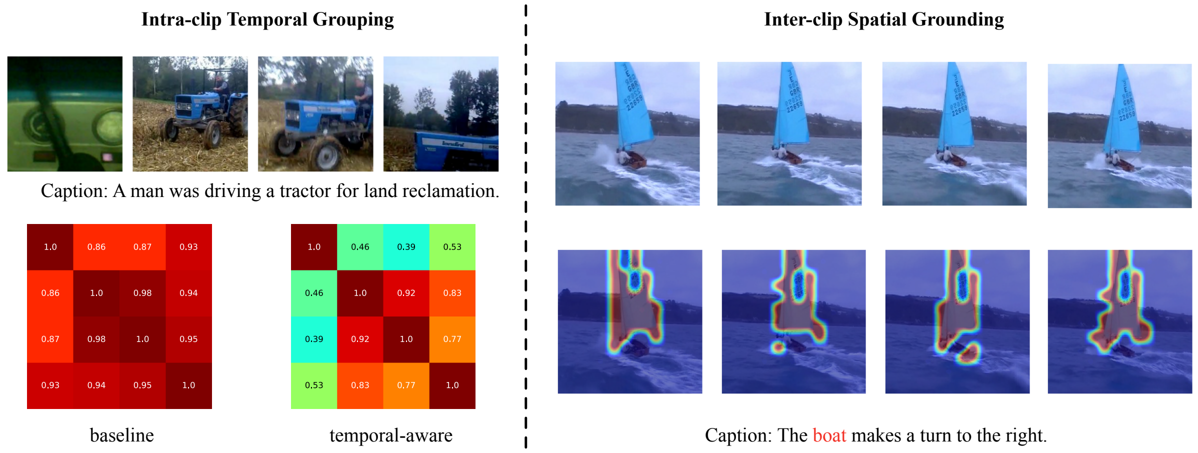


Figure 7.2: Visualization of S-ViLM. Left: Similarity scores derived from the baseline and our method. Right: Attention maps between region and object with spatial grounding.

clusion that vision-language pre-training can not only be applied to specific VL problems like text-video retrieval, but also benefit single-modal downstream tasks.

7.4.3.5 Ablation Studies

Pre-training datasets. To analyze of effects of pre-training datasets, we report the model performances on selected downstream tasks in Table 7.5. In particular, the same model pre-trained on VideoCC achieves the best performance in zero-shot retrieval on MSR-VTT, compared with HowTo100M and WebVid-2M. These results coincide with findings in [NSS22], where HowTo100M has been pointed out not appropriate for vision-language tasks requiring strong alignment. S-ViLM trained on VideoCC alone significantly outperforms VCC on both tasks, showing the effectiveness of our proposed techniques. In particular, when pre-trained on the same VideoCC dataset, S-ViLM leads to better performance than MCQ. The significant improvement over MCQ shows that our techniques do help to learn better features for downstream tasks. It is also worth noting that pre-training on VideoCC and ActivityNet performs consistently better than using only one dataset, and thus we choose this setup in the main experiments.

Training objectives. Without loss of generality, the model in this ablation is pre-trained

Table 7.5: Effects on different choices of pre-training datasets.

Method	PT Dataset	MSRVTT-ZS			TAL			
		R@1	R@5	R@10	mAP@0.5	0.75	0.95	Avg
VCC	HowTo100M	10.4	22.2	30.0		-		
	WebVid	15.4	33.6	44.1		-		
	VideoCC	18.9	37.5	47.1	49.9	34.3	8.7	33.7
MCQ	VideoCC	22.5	43.8	54.8		-		
S-ViLM	HowTo100M	9.4	22.9	31.3	51.7	36.4	9.7	35.6
	ActivityNet	14.4	33.5	44.0	50.5	35.3	8.7	34.5
	VideoCC	24.7	47.4	59.0	50.5	35.0	9.2	34.2
	VideoCC, ActivityNet	28.6	53.6	65.1	50.8	35.6	9.3	34.7

on VideoCC only. For better understanding S-ViLM, we start with the contrastive baseline represented in Scenario 1 in Table 7.6. Then we add our proposed spatial grouping module during the pre-training phase. This module is driven by the grouping loss $\mathcal{L}_g$ in Scenario 2, and we observe consistent improvements on all tasks across the board comparing to Scenario 1. Similarly, we introduce the temporal grouping module in $\mathcal{L}_t$ to encourage more temporal discriminative video representation. After comparing Scenario 3 to Scenario 1 in Table 7.6, we also observe noticeable improvements on different downstream tasks. These phenomenons suggest both spatially and temporally fine-grained features improve video understanding tasks. After combining everything together in Scenario 4, we show significant performance improvements on all tasks, which demonstrates the effectiveness of S-ViLM pre-training. Moreover, we visualize effects of temporal grouping and spatial grounding in Figure 7.2. It can be observed from similarity scores among frames that with temporal grouping, features from different scenes are much easier to distinguish. Besides, attention maps from spatial grounding indicates the alignment between the region and the object has been learned in the pre-training stage without any fine-grained annotations.

7.5 Conclusion

In this work, we present a novel video-language pre-training framework, named S-ViLM, that aims to utilize fine-grained structures in video and languages to learn region-object

Table 7.6: Ablation study on training objectives. We validate that our proposed spatial grounding loss and temporal grouping loss both benefit downstream tasks.

Scenario	$\mathcal{L}_{\text{contrast}}$	$\mathcal{L}_{\text{g}}$	$\mathcal{L}_{\text{t}}$	**MSRVTT-ZS**			**MSVD-QA**	**UCF101**	**TAL**			
				R@1	R@5	R@10	Acc	Acc	mAP@0.5	0.75	0.95	Avg
1	✓			22.7	45.9	57.0	43.6	90.5	49.9	34.3	8.7	33.7
2	✓	✓		23.3	46.6	58.6	44.1	90.6	50.2	34.7	8.7	34.0
3	✓		✓	24.2	46.7	58.2	43.9	90.9	50.1	34.6	8.8	34.0
4	✓	✓	✓	24.7	47.4	59.0	44.9	91.0	50.5	35.0	9.2	34.2

correspondences and temporal-aware features simultaneously. Spatial grounding and temporal grouping are introduced to achieve the goal of local region-object alignment and temporal distinction in a self-supervised manner. The proposed framework outperforms existing methods significantly on downstream tasks, including text-video retrieval, video question answering, video action recognition, and temporal action localization. The superior performance validates our design and our method could be easily scaled up, as it is self-contained and does not rely on other artifacts.

CHAPTER 8

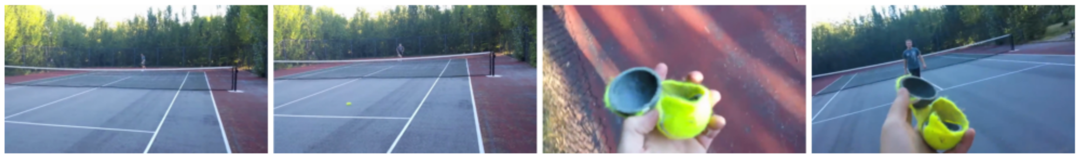
UNICORN: A Unified Causal Video-Oriented Language-Modeling Framework for Temporal Video-Language Tasks

Even though video-language representation learning has achieved great success in a variety of downstream tasks, problems such as mismatching objectives between pre-training and fine-tuning stages impedes its development. Therefore, we resort to multimodal language modeling for a unified framework to tackle temporal video-language tasks in this chapter. The learned model outperforms its counterparts on downstream tasks and is leveraged in an automatic annotation process for improved data efficiency.

8.1 Introduction

Recent breakthroughs in large language models (LLMs) [OWJ22, cha23, Ope23, vic23, TLI23, TMS23] have reignited the enthusiasm about the achievement of artificial general intelligence where a single foundation model can accomplish a large variety of downstream tasks based on human instructions. Towards this ultimate goal, the community has witnessed promising advances in powerful large multimodal models (LMMs) for vision and language [LLW23, LLL23b, WCC23, DLL23, BBY23, LZC23a, ZCS23], the two essential modalities to understand the world. Most of these LMMs follow the pipeline of visual instruction tuning [LLW23] and demonstrate strong capabilities in various tasks, including

Table 8.1: Example of instruction-following data. In particular, the response of moment retrieval is computed by time tokenization for the window [7.7s, 22.1s] with 75 bins.

Visual input example, Playing Tennis (34s in total):	
	
Task 1: Moment Retrieval	
Instruction	Please predict start and end time of the following moment: ***He hits the ball over the net several times***. The output format should be <start><end>.
Response	<16><48>
Task 2: Video Paragraph Captioning	
Instruction	Provide a detailed description of the video, capturing its key moments.
Response	A man is bouncing a tennis ball on an outdoor court. He hits the ball over the net several times. The balls roll over to the opposing fence, broken in half from the impact.

vision-centric tasks like image classification and object detection [WCC23], and vision-language tasks like image captioning and visual question answering [DLL23, LLW23].

Despite impressive results in the image domain, videos, another important data format in the vision modality, are under-explored. In contrast to images, videos have an extra temporal dimension and are much more difficult to process due to increased complexity. Existing approaches either directly apply LMMs trained on image-text pairs [DLL23] to the video domain without fine-tuning or develop video-oriented LMMs [ZLB23, MK23, LHW23] on short trimmed videos. However, such models are limited to handle problems which are less dependent on temporal information like action recognition and video question answering. It still remains unclear how to solve video-language tasks that requires explicit temporal modeling, including moment retrieval [HWS17, LBB21], video paragraph captioning [PRD19], and dense video captioning [KHR17] in one single LMM.

In fact, the inherent disparities among these task formats pose a challenge to the development of such models: moment retrieval requires predicting the temporal location of a moment described by language, paragraph captioning entails to write a coherent story from an untrimmed video, while the goal of dense video captioning is to generate captions and temporal locations for a series of moments simultaneously. These tasks are typically solved individually by specifically-designed models [LWS20, YNS23, LBB21, LZC23b]. While attempts have been made to unify these temporal video-language tasks [WZZ23, YXN23], separate modules and training objectives tailored for each task are involved in these methods, making them complicated in both training and inference.

To address these challenges, we propose a **UNI**fied **C**ausal vide**O**-o**R**iented la**N**guage modeling framework (**UNICORN**) that unifies the tasks as a simple yet generic language modeling problem. For moment retrieval and video paragraph captioning, we convert original training datasets into corresponding instruction-following formats, as shown in Table 8.1. In particular, inspired by previous efforts in discretizing bounding box coordinates [CSL22, PWD23, ZSC23], our approach represents the continuous event boundaries as a sequence of discrete tokens via a novel time-tokenization procedure to adapt this task into a unified language interface. The model then generates the target output in the form of natural language descriptions conditioned on the input video and instruction. On a range of datasets and tasks, we show that our unified approach achieve comparable or better performance over previous methods.

On the other hand, the development of large video-language models is hindered by the lack of semantically- and temporally-aligned video-text pairs, an issue unique to the video domain. As pointed out in [HXZ22], the models pre-trained on commonly-used noisy datasets such as HowTo100M [MZA19] and YT-Temporal-1B [ZLL22] suffer from the misalignment between videos and captions severely. Thanks to the generalization ability of LMMs, our UNICORN can be leveraged to automatically annotate public internet videos lacking in human annotations to collect a high-quality pre-training dataset with better aligned captions. We demonstrate that **qualitatively** the corresponding dataset contains more semantically- and temporally-aligned captions compared with original automated transcribed speeches, and **quantitatively** incorporating our generated captions in either instruction-tuning for moment retrieval or end-to-end representation learning leads to significant performance gains.

Our primary contributions are summarized as follows: **(1)** We propose UNICORN, a simple and generic framework that unifies various temporal video-language tasks via language modeling; **(2)** Our approach achieves comparable or better performance to state-of-the-art methods on multiple downstream tasks, including moment retrieval, video paragraph captioning, and dense video captioning; **(3)** Compared to existing captions, those automatically generated by our method have shown to be better aligned with the videos, both semantically and temporally. Empirically, the generated captions have demonstrated to improve performance of models trained on them. Our automatic annotation pipeline can be useful for empowering the development of future large multimodal models.

8.2 Related Work

Large Multimodal Models. Large language models are taking the world by storm with their incredible capabilities to answer questions in a coherent and informative way aligned with human instructions [cha23, OWJ22, vic23, Ope23, TLI23, TMS23]. The universality and generalization of LLMs make it potential to unlock the door to a foundation general-purpose model. Towards this goal, a variety of large multimodal models are emerging to bridge different modalities, in particular vision and language [LLW23, LLL23b, WCC23, DLL23, BBY23, LZC23a, ZCS23]. Such LMMs adopt the pipeline of visual instruction tuning [LLW23] by converting original datasets into the instruction-following format and casting traditional computer vision problems as a language modeling task. For instance, LLaVa [LLW23] generates multimodal language-image instructional data using GPT-4 [Ope23] and develops an LMM connecting a pre-trained image encoder and a pre-trained large language model to deal with vision-language tasks. InstructBLIP [DLL23] enlarges the task coverage by gathering 26 publicly available datasets and proposes an instruction-aware visual feature extraction process. These models achieve the state-of-the-art performance on numerous downstream tasks, ranging from vision-centric ones such as image classification and object detection to vision-language ones such as image captioning and visual reasoning. Despite efforts in understanding images, few attempts have been made for video-language tasks due to additional complexity. Thus, in this paper we study how to model the interaction between long untrimmed videos and captions from the perspective of language modeling.

Video-Language Modeling. While large multimodal models tailored for videos are under-explored, video-language modeling has been a heated topic for past few years. Elaborate designs and complicated structures have been proposed to solve particular video-language tasks respectively such as moment retrieval [LBB21, LZC23b, MCH20, ZXH20], video paragraph captioning [LWS20, PRD19, YNS23, WZL21], and dense video captioning [KHR17, YNS23, WZL21]. Some methods [LZC23b, YXN23, WZZ23, LLL22] pretrain a model on large-scale corpus to generate latent video and language representations, which can be then adapted to different downstream tasks. This line of work typically requires elaborate designs and multiple training objectives tailored for each target task. In contrast, we propose a more elegant unified framework to integrate various temporal video-language tasks into a simple yet generic language modeling problem. Compared with existing video-oriented LMMs targeting at short video clips [LHW23, ZLB23, MK23], UNICORN attaches more attention to long untrimmed videos. The most relevant method to UNICORN is Vid2Seq [YNS23], which also formulates dense video captioning as language modeling. However, it should be emphasized that Vid2Seq depends heavily on pre-training and is unable to handle tasks other than captioning. On the contrary, by visual instruction tuning on high quality datasets, our efficient and effective UNICORN demonstrates superior performance on a series of video-language tasks without intensive pre-training. Moreover, our method can be applied towards noisy video datasets to generate better-aligned captions.

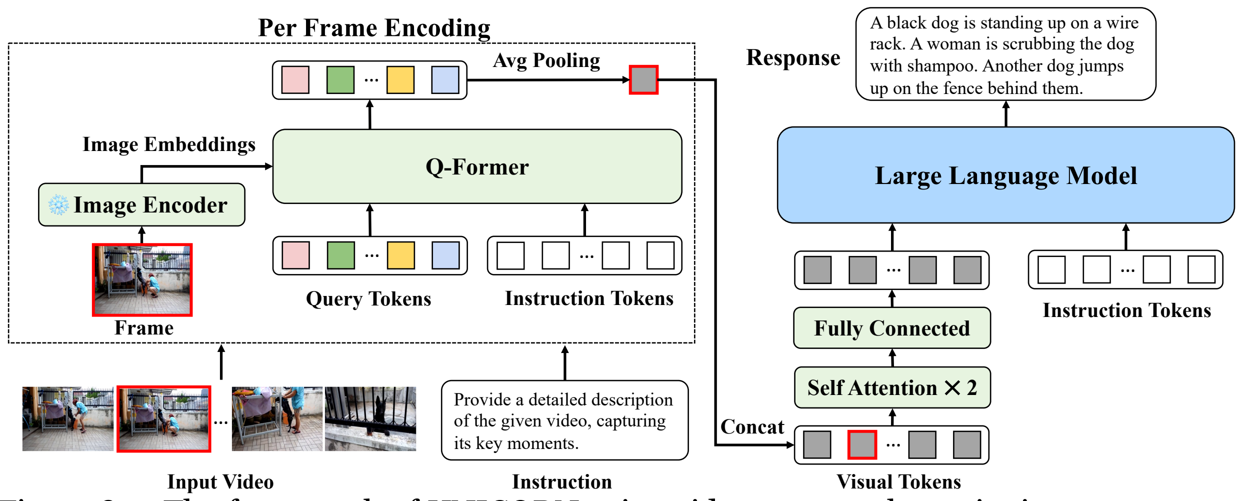


Figure 8.1: The framework of UNICORN using video paragraph captioning as an example. Each frame of the input video is encoded individually to obtain a visual token and we concatenate all visual tokens to represent the video. We highlight the encoding process of one frame in red. All modules are instruction-tuned with the generic language modeling loss except for the frozen image encoder.

8.3 Method

In this section, we introduce our unified framework UNICORN in detail. We start by discussing how to transform the original datasets for different downstream tasks into the general instruction-following format in Section 8.3.1. Then in Section 8.3.2, we describe the model architecture designed for video-language interaction. In Section 8.3.3, we present the training pipeline of UNICORN including datasets and training objective. Finally in Section 8.3.4, we demonstrate how to conduct inference with the obtained model on downstream tasks together with the process to generate captions for noisy datasets.

8.3.1 Instruction-Following Data Generation

As the ultimate goal of our method is to unify various temporal video-language tasks in one framework, we cast moment retrieval and video paragraph captioning into a sequence-to-sequence problem with instruction following formats. For dense video captioning, it can be regarded as a two-stage procedure of paragraphing captioning and moment retrieval and thus no specific training data are required. We provide details in following sections.

Moment Retrieval In moment retrieval (MR) [HWS17, GSY17, KHR17, LYB20, LBB21], a continuous time window with start and end timestamps is predicted given an untrimmed video and a language moment query. With the task definition, an example instruction can be: "**Please predict start and end time of the following moment: {target}**", where **{target}** is replaced by the specific query. We curate a template instruction list to explicitly teach the underlying model the concepts of the task and the objective.

A key challenge here is how to generate output sequences to represent moment locations. To reduce the exploration space for more controllable predictions, we follow previous sequence generation strategies for such continuous values [CSL22, PWD23, YNS23, WCC23, CZZ23], and discretize the timestamp t in a d-s long video into an integer in $\{0, 1, \ldots, N_{\text{bin}} - 1\}$ with N_{bin} equally-spaced bins by $\lfloor t \times N_{\text{bin}} \rfloor / d$. Moreover, since recent large language models exhibit surprising performance in mathematical reasoning, we use the original vocabulary without adding extra special time tokens, which in turn reduces the number of trainable parameters and avoids large-scale pre-training to re-acquire the ability to reason about numbers. Meanwhile, to distinguish our discrete relative timestamps from other numerical expressions, such as "5 apples", we enclose the timestamp values into angle brackets: "<start><end>" where start and end are replaced by corresponding converted timestamps. For instance, as illustrated in Table 8.1, the moment starting at 7.7s and ending at 22.1s within a 34s-long video is transformed into the desired output sequence "<16><48>" after our proposed time tokenization with 75 bins. To make output predictions consistent in format, we append a language constraint to our instruction: "**The output format should be <start><end>.**" For a moment query associated with multiple time windows, we regard each query-location pair as an individual data sample and reform it into the above instruction-following format.

Video Paragraph Captioning In contrast to single-sentence captioning [XMY16b], video paragraph captioning (VPC) [PRD19, LWS20] aims at generating a set of coherent sentences to describe an untrimmed video that contains several events. While previous pipelines [PRD19, LWS20] decompose the problem by segmenting the video into multiple clips based on ground-truth event boundary proposals, our method takes as input frames sampled from the whole untrimmed video and the instruction "**Provide a detailed description of the given video, capturing its key moments.**". For the output, it is intuitive to leverage the paragraph caption of the target video as the prediction.

Dense Video Captioning The goal of dense video captioning (DVC) [KHR17] is to generate multiple corresponding captions for a series of events together with their temporal locations from the untrimmed video. It is much harder than moment retrieval and paragraph captioning since it requires predicting events and their timestamps simultaneously. The most straightforward way to convert the task into the instruction-following format is to construct a sequence containing the information of both events and locations given a specific input prompt instruction. However, design choices such as event serialization (e.g., chronological or random) and where to insert associated time windows might affect the performance significantly [CSL22, YNS23]. Furthermore, the training of such models is challenged by the longer input sequence with both timestamps and event descriptions. It also takes extra computational costs to learn redundant information from moment retrieval and video paragraph captioning again. Considering the inherent property of dense video captioning, we find that it can be naturally decomposed into a two-stage procedure of video paragraph captioning followed by moment retrieval. Thus, no additional instruction-following data are required on dense video captioning and this task can be addressed at inference-time by the model instruction-tuned on two tasks above, with more details in Section 8.3.4.

8.3.2 Model Architecture

After the construction of instruction-following data, we need to design a model architecture that can be leveraged in a sequence-to-sequence framework by bridging together video frames and natural language instructions as the ultimate input sequence. To tackle this problem, we propose a large multimodal language model, demonstrated in Figure 8.1. Specifically, a sequence of visual tokens are obtained by feeding frames and the corresponding instruction from an untrimmed minutes-long video into our per-frame encoding module. Visual tokens are processed by a projection layer to the same latent space as the large language model (LLM). The LLM takes a concatenation of visual tokens and instruction tokens as input and generates the desired output given different task instructions.

How to connect vision and language is an essential problem in LMMs. Compared with image-language interaction, few attempts have been made in the video domain due to increased complexity. However, using image encoders to conduct per-frame encoding for videos by brute force will lead to an extremely long sequence of visual tokens proportional to the number of frames, which are then processed by the subsequent language model. On the other hand, a completely new encoder might require a considerable amount of training to align modalities of vision and language again. To strike a balance between two aforementioned issues, we resort to the recently proposed InstructBLIP [DLL23] and make some adaptions on the Q-Former module to handle the video input. In detail, our method first extracts n_q visual tokens from each frame using the frame-based encoding of the original Q-Former. For efficiency, we then apply average pooling in a frame-wise manner, which results in one token for each frame. Given a video of N frames, these N tokens are further processed by a module with two self-attention layers to integrate temporal information. Our design maintains a reasonable length of visual tokens for instruction tuning and takes advantage of pre-trained LLMs for feature alignment between the two modalities. Note that UNICORN is a generic framework to which we can flexibly utilize various LMMs as the base model. We analyze the effects of different LMMs like LLaVA [LLW23] in Section 4.5.6 to justify our framework design.

8.3.3 Training

With all data converted into the instruction-following format, we can now present a unified framework of instruction tuning on various downstream tasks.

Objective. The instruction-following format makes it feasible to train the model to predict next tokens with an auto-regressive language modeling loss. Given input video frames $X = \{x_i\}_{i=1}^{N}$ and the input task instruction $Y = \{y_j\}_{j=1}^{M}$, we maximize the log likelihood of the output sequence $Z = \{z_k\}_{k=1}^{L}$:

$$\max \sum_{k=1}^{L} \log p_\theta(z_k | X, Y, z_{1:k-1}), \tag{8.1}$$

where L is the length of the output target sequence. We use θ to represent all the trainable parameters in the model and p_θ denotes the output probability distribution over the vocabulary of the large language model.

Trainable parameters. As shown in Figure 8.1, we only keep the weights of the CLIP image encoder frozen during instruction tuning, and tune the rest modules including Q-Former, self-attention layers, the fully connected projection layer and the LLM. For efficient training, instead of fine-tuning the whole large language model, we use Low-Rank Adaptation (LoRA) [HSW22].

8.3.4 Inference

At inference time, since the model has already been instructed-tuned on the tasks of moment retrieval and paragraph captioning, we directly prompt the model using corresponding task instructions to generate responses via beam search. Note that for moment retrieval, we also need to de-quantize the predicted discrete time bins and transform them back to continuous windows by reversing time tokenization in Section 8.3.1. As to dense video captioning, recall that in Section 8.3.1 we point out that we can divide this task into a two-stage procedure. In detail, given the target video, the model first generates a paragraph caption based on the instruction, then the paragraph is split into a number of sentences and each sentence serves as a query for the subsequent moment retrieval task to obtain timestamps. In this way, we are able to obtain both captions and their associated timestamps during inference.

8.4 Experiments

In this section, we evaluate UNICORN comprehensively against state-of-the-art methods to show the effectiveness of our unified framework. We first introduce setups of all the experiments in Section 8.4.1. Then we present results on downstream tasks including moment retrieval, video paragraph captioning and dense video captioning in Section 8.4.2. Ablation studies are conducted in Section 8.4.3 for better understanding of our designs. Finally, in Section 8.4.4 we investigate the quality of the automatic annotation generated by UNICORN on HowTo100M.

8.4.1 Experimental Setups

Architecture. As discussed in Section 8.3.2, the backbone of our video encoding module is adapted from InstructBLIP [DLL23]. Specifically, we implement the video encoder with the same image encoder (ViT-G/14) [FWX23], Q-Former with 32 learnable query embeddings and a fully-connected projection layer as the original InstructBLIP structure, plus a temporal modeling module with 2 self-attention layers. For the language side, we select Vicuna-7B [vic23], a publicly available large language model fine-tuned with instructions from LLaMa [TLI23]. The video encoder is initialized from InstructBLIP [DLL23] except for the temporal modeling module.

Datasets. Rather than intensive pre-training on a large scale noisy dataset without an-

Table 8.2: Comparison with the state-of-the-art for moment retrieval. Results are reported on QVHighlights (*test*), Charades-STA (*test*), and ActivitityNet Captions (*val_2*) benchmarks. We **bold** the best results, and <u>underline</u> the second-best.

Method	QVHighlights			Charades-STA			ActivityNet Captions		
	R@0.5	R@0.7	mAP avg	R@0.5	R@0.7	mIoU	R@0.5	R@0.7	mIoU
LGI [MCH20]	—	—	—	59.5	35.5	51.4	41.5	23.1	41.1
2D TAN [ZPF20]	—	—	—	46.0	27.5	41.2	44.5	26.5	—
VSLNet [ZSJ20]	—	—	—	42.7	24.1	41.6	43.2	26.2	43.2
MDETR [LBB21]	59.8	40.3	36.1	52.1	30.6	45.5	—	—	—
GVL [WZZ23]	—	—	—	—	—	—	**48.9**	27.2	<u>46.4</u>
UnLoc [YXN23]	64.5	48.8	—	58.1	35.4	—	48.0	<u>29.7</u>	—
UniVTG [LZC23b]	58.9	40.9	35.5	58.0	35.6	50.1	—	—	—
UniVTG w/ PT [LZC23b]	<u>65.4</u>	<u>50.1</u>	<u>43.6</u>	<u>60.2</u>	<u>38.5</u>	<u>52.2</u>	—	—	—
UNICORN	**68.4**	**51.9**	**45.0**	**69.0**	**45.6**	**58.9**	<u>48.4</u>	**29.8**	**47.1**

notations, we directly fine-tune our model on a comprehensive set of publicly available video-language datasets, including QVHighlights [LBB21], Charades-STA [GSY17], ActivityNet Captions [KHR17], and YouCook2 [ZXC18]. The collection covers moment retrieval and video paragraph captioning (no training data for dense video captioning as discussed in Section 8.3.1), and is from various domains with different length distributions.

Instruction tuning details. We adopt the LAVIS library [LLL23a] to implement our model and run all the experiments. The model is instruction tuned for 5 epochs with a batch size of 32. We randomly sample one task at a time based on data size. We use AdamW [LH19] with $\beta_1 = 0.9, \beta_2 = 0.999$, and a weight decay of 0.05 to optimize the model. The learning rate is warmuped from 10^{-6} to 10^{-4} in the first epoch, followed by a cosine decay with a minimum of 10^{-5}. We freeze the image encoder and fine-tune the rest of the model, with LoRA applied on the LLM. The number of trainable parameters is around 243M. UNICORN is trained utilizing 8 NVIDIA A100 (80G) GPUs in 12 hours.,

Evaluation. For moment retrieval, we evaluate on QVHighlights, Charades-STA, and ActivityNet Captions. We report the standard metrics Recall at 1 under temporal Intersection over Union (IoU) thresholds of 0.5 and 0.7, abbreviated as R@0.5 and R@0.7. Besides, we use the average mAP over IoU thresholds [0.5:0.05:0.95] on QVHighlights with multiple ground-truth segments for one moment, and mean IoU (mIoU) for the other two datasets.

For video paragraph captioning, we use commonly-adopted metrics CIDEr [VLP15] (C) and METEOR [BL05] (M) and report results on YouCook2 and ActivityNet Captions. As to dense video captioning, we follow the existing protocol [KHR17] which first matches pairs between generated sentences and the ground truth across IoU thresholds [0.3, 0.5, 0.7, 0.9] and compute captioning metrics over the matched pairs. SODA_c [FHK20] (S) is also used to measure the coherence for a set of captions by considering their temporal structure. This task is evaluated on YouCook2 and ActivityNet Captions as well.

8.4.2 Results

We evaluate our instruction-tuned model on three video-language tasks: moment retrieval, video paragraph captioning, and dense video captioning. Note that all results are obtained from one shared model and different tasks are addressed by changing the prompting instructions *at inference time* only.

Moment retrieval. In Table 8.2, our method is compared with state-of-the-art algorithms for this task on three representative datasets, QVHighlights [LBB21], Charades-STA [GSY17], and ActivityNet Captions [KHR17]. It can be observed that our method achieves comparable (mostly better) performance on all three datasets. In particular, on QVHighlights we achieve 68.4, 51.9, and 45.0 for R@0.5, R@0.7 and average mAP respectively, improving the best-performing baseline UniVTG with pre-training substantially by +3.0, +1.8 and +1.4. Note that all the baseline methods adopt the conventional localization loss for a regression problem to predict moment boundaries while we only use a generic language modeling loss. In contrast to complicated designs in previous approaches, we remove most of the specification: UNICORN is based mainly on the intuition that if a model knows about where the moment is within the video, we just need to teach it how to read the location out. In summary, UNICORN makes minimal assumptions on the task yet accomplishes it with superior performance.

Table 8.3: Comparison with the SoTA for video paragraph captioning on YouCook2 (*val*) and ActivityNet Captions (*ae-test*). V/F/O refers to visual/flow/object features.

Method	**Backbone**	**YouCook2** C	**YouCook2** M	**ActivityNet** C	**ActivityNet** M
With GT Proposals					
VTransformer [ZZC18]	V (ResNet-200) + F	32.3	15.7	22.2	15.6
Transformer-XL [DYY19]	V (ResNet-200) + F	26.4	14.8	21.7	15.1
MART [LWS20]	V (ResNet-200) + F	<u>35.7</u>	<u>15.9</u>	23.4	15.7
GVDSup [ZKC19]	V (ResNet-101) + F + O	—	—	22.9	16.4
AdvInf [PRD19]	V (ResNet-101) + F + O	—	—	21.0	16.6
PDVC [WZL21]	V + F (TSN)	—	—	27.3	15.9
With Learned Proposals					
MFT [XDL18]	V + F (TSN)	—	—	19.1	14.7
PDVC [WZL21]	V + F (TSN)	—	—	20.5	15.8
PDVC [WZL21]	V (CLIP)	—	—	23.6	15.9
TDPC [SCJ21]	V (ResNet-200) + F	—	—	26.5	15.6
Vid2Seq [YNS23]	V (CLIP)	—	—	<u>28.0</u>	<u>17.0</u>
GVL [WZZ23]	V (TSN)	—	—	26.0	16.3
UNICORN	V (CLIP)	**37.8**	**18.3**	**34.8**	**17.3**

Video paragraph captioning. Table 8.3 shows the video paragraph captioning results. In UNICORN, we consider this task as a general captioning problem, and the only difference is that the target output sequence now becomes a paragraph with multiple sentences instead of only one single caption. Without any customized training objectives or prior knowledge on the input such as ground-truth event proposals as in previous methods [PRD19, LWS20], our method demonstrates outstanding performance on captioning metrics over other baselines under both settings of ground truth or learned proposals. It further showcases the strong adaptation of LMMs to downstream tasks through instruction tuning with high-quality instruction-following data.

Dense video captioning. We generate dense video captions following the procedure introduced in Section 8.3.1 and evaluate the performance in Table 8.4. It can be observed that our method takes the lead among the compared approaches, including Vid2Seq which leverages language modeling to predict captions and their associated timestamps simultaneously. These promising results also demonstrate the effectiveness of our divide-and-conquer strategy for dense video captioning. Such an inference design makes the training more efficient without learning on redundant and lengthy DVC data again while still achieving competitive results.

Table 8.4: Comparison with SoTA for dense video captioning on YouCook2 (*val*) and ActivityNet Captions (*val_1* and *val_2*).

Method	**Backbone**	**YouCook2**			**ActivityNet**		
		S	C	M	S	C	M
MT [ZZC18]	TSN	—	6.1	3.2	—	9.3	5.0
ECHR [WZY20]	C3D	—	—	3.8	3.2	14.7	7.2
PDVC [WZL21]	TSN	4.4	22.7	4.7	5.4	29.0	8.0
PDVC [WZL21]	CLIP	4.9	<u>28.9</u>	<u>5.7</u>	<u>6.0</u>	29.3	7.6
UEDVC [ZSJ22]	TSN	—	—	—	5.5	26.9	7.3
E2ESG [ZPT22]	C3D	—	25.0	3.5	—	—	—-
Vid2Seq [YNS23]	CLIP	**5.7**	25.3	—	5.9	30.2	<u>8.5</u>
GVL [WZZ23]	TSN	4.9	26.5	5.0	<u>6.2</u>	<u>32.8</u>	<u>8.5</u>
UNICORN	CLIP	**5.7**	**37.0**	**7.7**	**6.3**	**35.4**	**9.2**

8.4.3 Ablation Studies

We conduct ablation studies to further analyze effects of the key components in UNICORN, including training strategies, base model selection, the choice of time tokens, and various model designs. We evaluate the performance on QVHighlights (*val*) for moment retrieval and on ActivityNet Captions (*ae-test*) for video paragraph captioning.

Training strategies. In this section, we study the effect of the multi-task and multi-dataset training strategy used in UNICORN. Specifically, we consider three strategies: single-task & single dataset, single task & multi-dataset, and multi-task & multi-dataset. For the single-task version, we fine-tune two separate models with corresponding instructions tailored for moment retrieval and video paragraph captioning respectively, and select one representative dataset for each task for evaluation. For the single-dataset version, we train only on the training split of the evaluation dataset (i.e., QVHighlights for moment retrieval and ActivityNet Captions for video paragraph captioning)

We report detailed results in Table 8.5a. By introducing datasets from different domains for the same task, we can improve the model's capability on the single dataset. Be-

Table 8.5: Ablation studies on training strategies and base model selection.

Training Setup	QVHighlights R@0.5	QVHighlights R@0.7	QVHighlights mAP	ActivityNet C	ActivityNet M
Single-task, single-dataset	66.3	51.5	42.8	33.6	16.4
Single-task, multi-dataset	68.2	52.3	44.8	34.6	16.9
Multi-task, multi-dataset	**69.5**	**54.4**	**45.3**	**34.8**	**17.3**

(a) Comparison of different training strategies.

Base model	QVHighlights R@0.5	QVHighlights R@0.7	QVHighlights mAP	ActivityNet C	ActivityNet M
LLaVA-7B [LLW23]	66.3	51.5	42.8	33.6	16.4
InstructBLIP-7B [DLL23]	68.2	52.3	44.8	34.6	16.9
InstructBLIP-13B [DLL23]	**69.5**	**54.4**	**45.3**	**34.8**	**17.3**

(b) Comparison of different base models.

Table 8.6: Ablation studies on model designs including the number of frames, LoRA, and temporal modeling.

#frames	QVHighlights R@0.5	QVHighlights R@0.7	QVHighlights mAP	ActivityNet C	ActivityNet M
25	61.5	37.4	35.0	33.4	16.9
50	65.4	47.9	41.4	34.5	17.0
75	**69.5**	**54.4**	**45.3**	**34.8**	**17.3**
100	67.8	52.8	44.7	34.6	**17.3**

(a) Effects of the number of frames.

LoRA	Temporal modeling	QVHighlights R@0.5	QVHighlights R@0.7	QVHighlights mAP	ActivityNet C	ActivityNet M
✗	✗	60.6	36.4	33.2	23.0	16.0
✓	✗	66.7	49.2	39.8	34.4	17.2
✗	✓	65.5	47.0	40.4	27.6	16.8
✓	✓	**69.5**	**54.4**	**45.3**	**34.8**	**17.3**

(b) Effects of LoRA and temporal modeling.

sides, in contrast to traditional multi-task training strategies, instruction tuning on various descriptions is more like a unified approach to integrate different tasks and can even boost the performance from understanding a video from multiple perspectives. Meanwhile, it is more convenient to store only one model to accomplish distinct tasks, which narrows the gap from constructing a general-purpose foundation model.

Base model. As discussed in Section 8.3.2, our framework is generic to different model structures and the base model can be replaced by other LMMs like LLaVA [LLW23] with a simple re-design to take video inputs. Thus, we study the effects of UNICORN adopting different base models. Specifically, we instruction-tuned LLaVA for moment retrieval and video paragraph captioning. Results are shown in Table 8.5b and it is expected that the performance of LLaVA variant drops compared with our InstructBLIP variant, due to the information loss from 256 frame-level tokens pooled to one token. Besides, InstructBLIP has a QFormer while LLaVA only uses a simple projection layer, which may be insufficient to align video and language. We also added an experiment with InstrutBLIP-13B and observed performance gains with a larger model size.

Time tokens. To represent time in our tasks, we can either introduce new dedicated time tokens or directly use the digits in the original vocabulary. We investigate the impacts of these two strategies in the single-task, single-dataset setup on QVHighlights in Table 8.7.

Table 8.7: Comparison between two choices of time tokens.

Time tokens	R@0.5	R@0.7	mAP
Dedicated	64.1	48.2	40.7
Original vocab	**66.3**	**51.5**	**42.8**

We can observe that representing time via the original vocabulary performs better than new dedicated tokens, which indicates the knowledge of digits in LLM can be transferred to our tasks easily. Meanwhile, new tokens would introduce extra overheads and increase the number of trainable parameters by 262M, more than double of the original value.

Number of frames. By default, we evenly sample 75 frames from a video as model inputs. In Table 8.6a, we study the impact with #frames of 25, 50, 75, and 100. The performance generally improves when we adopt more frames while it saturates or even gets worse around 100 frames. Since the videos in the datasets we studied are usually not very long (*e.g.*, videos in QVHighlights are on average 150 seconds long), we hypothesize that 75 frames are enough to cover all the semantic information needed for the tasks.

LoRA. LoRA is a parameter-efficient fine-tuning method, and we use it to fine-tune the LLM of UNICORN. We analyze the effects of using LoRA in Table 8.6b. LoRA has been proven effective in boosting performance for downstream tasks (row 1 $\rightarrow$ 2 and row 3 $\rightarrow$ 4). It is expected that frozen LLM would not work properly because we have assigned new meanings to original digit tokens to represent discrete time bins, and efficient LoRA training mitigates the issue without tuning the whole LLM intensively.

Temporal modeling. Since our model is adapted from image-based InstructBLIP, we include an additional module with self-attention layers on top of Q-Former to incorporate temporal information for video inputs in Figure 8.1. As shown in Table 8.6b, when temporal modeling is enabled from average pooling only to self-attention interaction (row 1 $\rightarrow$ 3 and row 2 $\rightarrow$ 4), there is substantial improvement in both moment retrieval and paragraph captioning, indicating the necessity of this module for temporal video-language tasks.

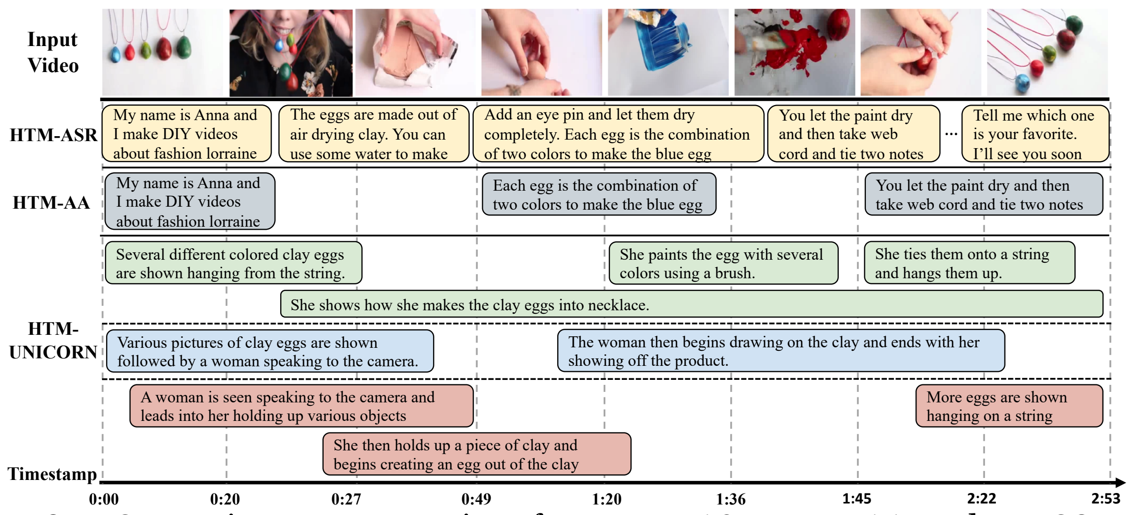

Input Video
HTM-ASR
My name is Anna and I make DIY videos about fashion lorraine
The eggs are made out of air drying clay. You can use some water to make
Add an eye pin and let them dry completely. Each egg is the combination of two colors to make the blue egg
You let the paint dry and then take web cord and tie two notes
…
Tell me which one is your favorite. I'll see you soon
HTM-AA
My name is Anna and I make DIY videos about fashion lorraine
Each egg is the combination of two colors to make the blue egg
You let the paint dry and then take web cord and tie two notes
HTM-UNICORN
Several different colored clay eggs are shown hanging from the string.
She paints the egg with several colors using a brush.
She ties them onto a string and hangs them up.
She shows how she makes the clay eggs into necklace.
Various pictures of clay eggs are shown followed by a woman speaking to the camera.
The woman then begins drawing on the clay and ends with her showing off the product.
A woman is seen speaking to the camera and leads into her holding up various objects
More eggs are shown hanging on a string
She then holds up a piece of clay and begins creating an egg out of the clay
Timestamp
0:00
0:20
0:27
0:49
1:20
1:36
1:45
2:22
2:53

Figure 8.2: Comparison among captions from HTM-ASR, HTM-AA, and UNICORN respectively. For HTM-UNICORN, we show three sets of generated captions via beam search, coded with different colors.

8.4.4 Auto Annotation of HowTo100M

Thanks to the generalization of LMMs, the instruction-tuned model to deal with temporal video-language tasks can be further deployed on large-scale public video datasets crawled from the web such as HowTo100M [MZA19] and YT-Temporal-1B [ZLL22]. Videos in these datasets are previously paired with captions from auto speech recognition (ASR), a majority of which are not visually and temporally aligned [MAS20, TLB21, HXZ22]. Since our model is capable of generating dense captions for the target video, it is a promising direction to leverage UNICORN for annotating the dataset automatically.

To evaluate the quality of our generated captions, we use our trained model to densely caption a subset of 240K videos from the HowTo100M dataset in [HXZ22]. We denote our annotated dataset as HTM-UNICORN, and compare it with two variants with the same set of videos, HTM-ASR [MZA19] with original ASR transcripts, and HTM-AA [HXZ22] which is a version of HTM-ASR that has been aligned temporally via an automated process. Note that UNICORN can output multiple diverse captions using beam search [VCS16], which can increase the training data size and as a result improve model performance.

Table 8.8: Zero-shot and fine-tuning moment retrieval evaluation on QVHighlights (*val*) among models pre-trained on different datasets. HTM-UNICORN $\times n$ indicates that we generated n sets of captions for each video.

Dataset	**#queries**	**Zero-shot**			**Fine-tuning**		
		R@0.5	R@0.7	mAP	R@0.5	R@0.7	mAP
InstructBLIP	—	—	—	—	66.3	51.5	42.8
HTM-ASR [MZA19]	5.0M	7.7	2.8	1.9	63.5	48.3	40.3
HTM-AA [HXZ22]	3.3M	13.0	4.8	3.5	65.8	50.6	42.1
HTM-UNICORN	690K	44.2	26.4	26.0	68.9	53.6	45.0
HTM-UNICORN $\times 2$	1.4M	47.5	30.8	29.9	69.5	54.0	45.2
HTM-UNICORN $\times 3$	2.1M	**50.0**	**32.7**	**30.4**	**70.2**	**54.6**	**45.5**

In Figure 8.2, we present an qualitative comparison of the 3 types of captions. Although the auto-aligned HTM-AA has removed irrelevant captions and reorganized the rest compared with HTM-ASR, it can be observed that our captions from HTM-UNICORN are the best aligned with the input video both visually and temporally, compared with HTM-ASR and the auto-aligned HTM-AA. In addition, captions from different sets can complement each other, leading to more comprehensive descriptions of the video.

For quantitative evaluation, we use three HowTo100M variants to pre-train the model with instructions for moment retrieval only, and compute metrics on QVHighlights under both zero-shot and fine-tuning settings. We convert these datasets into the instruction-following format as described in Section 8.3.1, and train the model from the same initialization. In Table 8.8, we can observe that our automatically annotated HTM can achieve superior zero-shot performance, which shows the better alignment of moments and their timestamps. Besides, we fine-tune the pre-trained models on QVHighlights to further analyze data quality in Table 8.8. We notice that performance even degrades on models pre-trained on HTM-ASR and HTM-AA, while the one pre-trained on HTM-UNICORN outperforms other variants, reflecting the high quality of the generated dataset.

Apart from temporal video-language tasks, we follow [HXZ22] to conduct end-to-end representation learning with an Info-NCE loss [MAS20] on the automatically annotated dataset. After contrastive pre-training, we evaluate video representations by linear probing on three action recognition datasets, UCF101 [SZS12], HMDB51 [KJG11], and Kinetics-400 (K400) [KCS17] in Table 8.9. UNICORN achieves the highest classification accuracy on all datasets, which again demonstrates the better quality of our generated captions.

Both qualitatively and quantitatively, data generated from our automated annotation pipeline utilizing UNICORN has shown to be better than noisy web data. As data quality and quantity are crucial for the performance of large models [ZLX24, JDG23, LLL23b], we hope such a pipeline could be useful for empowering the development of future large multimodal models.

Table 8.9: Linear probing accuracy for action recognition.

PT Dataset	**Backbone**	**UCF101**	**HMDB51**	**K400**
HTM-ASR [MAS20]	S3D	82.1	55.2	55.7
HTM-AA [HXZ22]	S3D	83.2	56.7	56.2
HTM-UNICORN	S3D	**84.1**	**57.7**	**56.6**

8.5 Conclusion

In this work, we propose a unified causal video-oriented language modeling framework UNICORN to address temporal video-language tasks. By finetuning on instruction-following data constructed from existing datasets, our model achieves outstanding performance on various downstream tasks including moment retrieval, video paragraph captioning and dense video captioning. We further show that UNICORN can be leveraged for automatic annotation on internet videos such as HowTo100M to provide semantically- and temporally-aligned captions. These captions can be used to improve video-language model performance against ASR ones. In conclusion, UNICORN paves the way towards a general-purpose foundation model that explicitly considers temporal information.

CHAPTER 9

Conclusions and Future Directions

Although prevailing research topics, approaches and methodologies in artificial intelligence have been evolving constantly, the pursuit of algorithm and data efficiency in deep learning will never pause. In this chapter, we provide a concise summary of what has been achieved for optimizing the computational and data resources required to train and deploy deep learning models, and delve into the future research tendencies in the field.

9.1 Summary and Conclusions

In summary, this thesis explores our contributions towards democratizing and realizing the full potential of deep learning models, categorized into two topics discussed below.

- **Algorithm efficiency.** To make deep learning approaches more algorithm efficient, we have focused on improving optimization process during the training stage. In particular, we have leveraged neural optimizers to replace orignial hand-designed ones for adversarial training with faster convergence rate and better robustness performance. We have also investigated techniques including smooth regularization and automatic learning rate scheduling to stabilize such meta-learned optimizers, reducing efforts in hyperparameter tuning significantly. Apart from optimization, we have utilize dataset distillation in federated learning to make existing methods more communication-efficient with few interactions between the server and the clients.
- **Data efficiency.** When the access to large scale annotated datasets is infeasible,

it is crucial to think of data efficient approaches. Pre-training followed by downstream fine-tuning is one of the most effective solutions to address the challenge of data scarcity. Specifically, we have successfully adopted contrastive learning to deal with zero-shot extreme text classification under limited supervision. We have extended such an idea to the video-language domain, making use of fine-grained structures concealed behind pre-training data pairs to learn more expressive spatiotemporal features. Moreover, we have exploited potentials of pre-trained large language models and fine-tuned a multimodal version for temporal video-language tasks efficiently. The obtained model has further enabled a novel annotation pipeline, improving data efficiency by generating high quality labeled data automatically.

9.2 Future Directions

While significant progress has been made in enhancing algorithm and data efficiency in deep learning, there remain numerous opportunities for further advancements in this critical area. The following are some promising future plans and directions:

- **Parameter-efficient fine-tuning.** Large models [cha23, Ope23, LLW23, DLL23] trained on vast amounts of data have demonstrated remarkable capabilities in reasoning and understanding tasks. However, the sheer size and complexity of these models, often comprising billions of parameters, pose significant challenges to computational resources and memory requirements. Various techniques, such as adapter modules [PSP23], prompt tuning [LAC21], prefix tuning [LL21], and LoRA [HSW22], have been proposed to realize parameter-efficient fine-tuning, A potential future direction can be selecting the optimal subset of parameters to fine-tune for a given task. This could involve leveraging meta-learning, reinforcement learning, or other techniques to dynamically identify the most relevant parameters based on task characteristics and resource constraints.

- **Model and data co-development.** Model and data co-development is an emerging paradigm that recognizes the interdependence between machine learning models and their training data. By closely integrating model design and data engineering, researchers and practitioners can maximize the value extracted from limited data resources, enabling the development of powerful AI systems in data-constrained settings. Such a pipeline has been successfully deployed in recent large models like Segment Anything [KMR23], where models can be designed to efficiently learn from partially labeled or weakly supervised data, reducing extensive manual annotation.
- **Building foundation models.** Foundation models [BHA21] which are trained on a considerable number data from diverse sources and modalities, have attracted great attention in the AI community. By pre-training a single, unified model on a broad range of data and tasks, we can acquire a rich, general-purpose model that can be adapted and fine-tuned for a wide variety of downstream applications, improving data efficiency significantly. This approach is contrary to the conventional training of specialized models for individual tasks, offering a more efficient and scalable path towards achieving artificial general intelligence (AGI).

Efficiency in deep learning is evolving at a rapid speed, with new directions/methods emerging frequently. There are many important research topics that are not discussed in this thesis, mostly due to the daily-updated research innovation. As we reflect on the accomplishments and challenges of efficient deep learning, it is clear that the journey is far from over. Future endeavors will continue to explore new frontiers in algorithmic efficiency and data utilization, making intelligent technologies accessible to all.

Bibliography

[ABR21] Elad Amrani, Rami Ben-Ari, Daniel Rotman, and Alex Bronstein. "Noise estimation using density estimation for self-supervised multimodal learning." In *AAAI Conference on Artificial Intelligence*, volume 35, pp. 6644–6652, 2021.

[ACG16] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. "Deep learning with differential privacy." In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pp. 308–318, 2016.

[Ach18] Joshua Achiam. "Spinning Up in Deep Reinforcement Learning." 2018.

[ACW18] Anish Athalye, Nicholas Carlini, and David Wagner. "Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples." In *International Conference on Machine Learning*, pp. 274–283, 2018.

[ADG16] Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando De Freitas. "Learning to learn by gradient descent by gradient descent." *arXiv preprint arXiv:1606.04474*, 2016.

[AGG21] Humam Alwassel, Silvio Giancola, and Bernard Ghanem. "Tsp: Temporally-sensitive pretraining of video encoders for localization tasks." In *IEEE International Conference on Computer Vision*, pp. 3173–3183, 2021.

[AHB18] Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. "Bottom-up and top-down attention for image captioning and visual question answering." In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6077–6086, 2018.

[ALA98] Luís B Almeida, Thibault Langlois, José D Amaral, and Alexander Plakhov. "Parameter adaptation in stochastic optimization." *On-Line Learning in Neural Networks, Publications of the Newton Institute*, pp. 111–134, 1998.

[AMK20] Humam Alwassel, Dhruv Mahajan, Bruno Korbar, Lorenzo Torresani, Bernard Ghanem, and Du Tran. "Self-supervised learning by cross-modal audio-video clustering." In *Advances in Neural Information Processing Systems*, volume 33, pp. 9758–9770, 2020.

[ARS20] Jean-Baptiste Alayrac, Adria Recasens, Rosalia Schneider, Relja Arandjelović, Jason Ramapuram, Jeffrey De Fauw, Lucas Smaira, Sander Dieleman, and Andrew Zisserman. "Self-supervised multimodal versatile networks." In *Advances in Neural Information Processing Systems*, volume 33, pp. 25–37, 2020.

[ASY18] Naman Agarwal, Ananda Theertha Suresh, Felix Xinnan X Yu, Sanjiv Kumar, and Brendan McMahan. "cpSGD: Communication-efficient and differentially-private distributed SGD." *Advances in Neural Information Processing Systems*, **31**, 2018.

[AYQ21] Hassan Akbari, Liangzhe Yuan, Rui Qian, Wei-Hong Chuang, Shih-Fu Chang, Yin Cui, and Boqing Gong. "Vatt: Transformers for multimodal self-supervised learning from raw video, audio and text." In *Advances in Neural Information Processing Systems*, volume 34, pp. 24206–24221, 2021.

[BB12] James Bergstra and Yoshua Bengio. "Random search for hyper-parameter optimization." *The Journal of Machine Learning Research*, **13**(1):281–305, 2012.

[BBC19] Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemyslaw Debiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris

Hesse, et al. “Dota 2 with large scale deep reinforcement learning.” *arXiv preprint arXiv:1912.06680*, 2019.

[BBY23] Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. “Qwen-vl: A frontier large vision-language model with versatile abilities.” *arXiv preprint arXiv:2308.12966*, 2023.

[BCR17] Atilim Gunes Baydin, Robert Cornish, David Martinez Rubio, Mark Schmidt, and Frank Wood. “Online learning rate adaptation with hypergradient descent.” *arXiv preprint arXiv:1703.04782*, 2017.

[BCW18] Nitin Bansal, Xiaohan Chen, and Zhangyang Wang. “Can we gain more from orthogonality regularizations in training deep CNNs?” *arXiv preprint arXiv:1810.09102*, 2018.

[BDJ16] K. Bhatia, K. Dahiya, H. Jain, P. Kar, A. Mittal, Y. Prabhu, and M. Varma. “The extreme classification repository: Multi-label datasets and code.”, 2016.

[BEL20] Sagie Benaim, Ariel Ephrat, Oran Lang, Inbar Mosseri, William T Freeman, Michael Rubinstein, Michal Irani, and Tali Dekel. “Speednet: Learning the speediness in videos.” In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9922–9931, 2020.

[BHA21] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. “On the opportunities and risks of foundation models.” *arXiv preprint arXiv:2108.07258*, 2021.

[BHB18] Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. "Relational inductive biases, deep learning, and graph networks." *arXiv preprint arXiv:1806.01261*, 2018.

[BJK15] Kush Bhatia, Himanshu Jain, Purushottam Kar, Manik Varma, and Prateek Jain. "Sparse Local Embeddings for Extreme Multi-label Classification." In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pp. 730–738, 2015.

[BL05] Satanjeev Banerjee and Alon Lavie. "METEOR: An automatic metric for MT evaluation with improved correlation with human judgments." In *ACL workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, 2005.

[BLC09] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. "Curriculum learning." In *Proceedings of the 26th Annual International Conference on Machine Learning, ICML 2009, Montreal, Quebec, Canada, June 14-18, 2009*, volume 382 of *ACM International Conference Proceeding Series*, pp. 41–48. ACM, 2009.

[BMK20] Zalán Borsos, Mojmir Mutny, and Andreas Krause. "Coresets via bilevel optimization for continual learning and streaming." *Advances in Neural Information Processing Systems*, **33**:14879–14890, 2020.

[BNV21] Max Bain, Arsha Nagrani, Gül Varol, and Andrew Zisserman. "Frozen in time: A joint video and image encoder for end-to-end retrieval." In *IEEE International Conference on Computer Vision*, pp. 1728–1738, 2021.

[BPL16] Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, and Koray kavukcuoglu. "Interaction networks for learning about objects, relations and physics." In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pp. 4509–4517, 2016.

[BRB17] Wieland Brendel, Jonas Rauber, and Matthias Bethge. "Decision-based adversarial attacks: Reliable attacks against black-box machine learning models." *arXiv preprint arXiv:1712.04248*, 2017.

[BWT21] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. "Is space-time attention all you need for video understanding?" In *International Conference on Machine Learning*, volume 2, p. 4, 2021.

[CC90] Neil E Cotter and Peter R Conwell. "Fixed-weight networks can learn." In *1990 IJCNN International Joint Conference on Neural Networks*, pp. 553–559. IEEE, 1990.

[CCC22] Tianlong Chen, Xiaohan Chen, Wuyang Chen, Howard Heaton, Jialin Liu, Zhangyang Wang, and Wotao Yin. "Learning to optimize: A primer and a benchmark." *Journal of Machine Learning Research*, **23**(189):1–59, 2022.

[CFG20] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. "Improved baselines with momentum contrastive learning." *arXiv preprint arXiv:2003.04297*, 2020.

[CFM19] Ilias Chalkidis, Emmanouil Fergadiotis, Prodromos Malakasiotis, and Ion Androutsopoulos. "Large-Scale Multi-Label Text Classification on EU Legislation." In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 6314–6322, Florence, Italy, 2019. Association for Computational Linguistics.

[CGW22] Guanyu Cai, Yixiao Ge, Alex Jinpeng Wang, Rui Yan, Xudong Lin, Ying Shan, Lianghua He, Xiaohu Qie, Jianping Wu, and Mike Zheng Shou. "Revitalize region feature for democratizing video-language pre-training." *arXiv preprint arXiv:2203.07720*, 2022.

[cha23] "ChatGPT." `https://openai.com/blog/chatgpt`, 2023.

[CHH21] Peihao Chen, Deng Huang, Dongliang He, Xiang Long, Runhao Zeng, Shilei Wen, Mingkui Tan, and Chuang Gan. "Rspnet: Relative speed perception for unsupervised video representation learning." In *AAAI Conference on Artificial Intelligence*, volume 35, pp. 1045–1053, 2021.

[CJY21] Wei-Cheng Chang, Daniel Jiang, Hsiang-Fu Yu, Choon Hui Teo, Jiong Zhang, Kai Zhong, Kedarnath Kolluri, Qie Hu, Nikhil Shandilya, Vyacheslav Ievgrafov, et al. "Extreme multi-label learning for semantic matching in product search." In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 2643–2651, 2021.

[CKN20] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. "A Simple Framework for Contrastive Learning of Visual Representations." In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 1597–1607. PMLR, 2020.

[CLC18] Minhao Cheng, Thong Le, Pin-Yu Chen, Jinfeng Yi, Huan Zhang, and Cho-Jui Hsieh. "Query-efficient hard-label black-box attack: An optimization-based approach." *arXiv preprint arXiv:1807.04457*, 2018.

[CLD21] A Feder Cooper, Yucheng Lu, and Christopher De Sa. "Hyperparameter Optimization Is Deceiving Us, and How to Stop It." *arXiv preprint arXiv:2102.03034*, 2021.

[CLY20] Yen-Chun Chen, Linjie Li, Licheng Yu, Ahmed El Kholy, Faisal Ahmed, Zhe

Gan, Yu Cheng, and Jingjing Liu. “Uniter: Universal image-text representation learning.” In *European Conference on Computer Vision*, pp. 104–120, 2020.

[CRK20] Patrick H. Chen, Sashank Reddi, Sanjiv Kumar, and Cho-Jui Hsieh. “Learning to learn with better convergence.”, 2020.

[CSA14] Trishul Chilimbi, Yutaka Suzue, Johnson Apacible, and Karthik Kalyanaraman. “Project adam: Building an efficient and scalable deep learning training system.” In *11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14)*, pp. 571–582, 2014.

[CSB16] Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun. “Very deep convolutional networks for text classification.” *arXiv preprint arXiv:1606.01781*, 2016.

[CSC20] Minhao Cheng, Simranjit Singh, Patrick H. Chen, Pin-Yu Chen, Sijia Liu, and Cho-Jui Hsieh. “Sign-OPT: A Query-Efficient Hard-label Adversarial Attack.” In *ICLR*, 2020.

[CSJ19] Yang Chen, Xiaoyan Sun, and Yaochu Jin. “Communication-efficient federated deep learning with layerwise asynchronous model update and temporally weighted aggregation.” *IEEE transactions on neural networks and learning systems*, **31**(10):4229–4238, 2019.

[CSL22] Ting Chen, Saurabh Saxena, Lala Li, David J Fleet, and Geoffrey Hinton. “Pix2seq: A language modeling framework for object detection.” In *ICLR*, 2022.

[CSN19] Dami Choi, Christopher J Shallue, Zachary Nado, Jaehoon Lee, Chris J Maddison, and George E Dahl. “On empirical comparisons of optimizers for deep learning.” *arXiv preprint arXiv:1910.05446*, 2019.

[CSP21] Mingzhe Chen, Nir Shlezinger, H Vincent Poor, Yonina C Eldar, and Shuguang Cui. "Communication-efficient federated learning." *Proceedings of the National Academy of Sciences*, **118**(17), 2021.

[CW17] Nicholas Carlini and David Wagner. "Towards evaluating the robustness of neural networks." In *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 39–57. IEEE, 2017.

[CYC20] Wei-Cheng Chang, Felix X. Yu, Yin-Wen Chang, Yiming Yang, and Sanjiv Kumar. "Pre-training Tasks for Embedding-based Large-scale Retrieval." In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[CYW22] Meng Cao, Tianyu Yang, Junwu Weng, Can Zhang, Jue Wang, and Yuexian Zou. "Locvtp: Video-text pre-training for temporal localization." In *European Conference on Computer Vision*, pp. 38–56, 2022.

[CYZ20] Wei-Cheng Chang, Hsiang-Fu Yu, Kai Zhong, Yiming Yang, and Inderjit S. Dhillon. "Taming Pretrained Transformers for Extreme Multi-label Text Classification." In *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*, pp. 3163–3171. ACM, 2020.

[CZJ20] Tianlong Chen, Weiyi Zhang, Zhou Jingyang, Shiyu Chang, Sijia Liu, Lisa Amini, and Zhangyang Wang. "Training Stronger Baselines for Learning to Optimize." *Advances in Neural Information Processing Systems*, **33**, 2020.

[CZM18] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. "Autoaugment: Learning augmentation policies from data." *arXiv preprint arXiv:1805.09501*, 2018.

[CZZ23] Keqin Chen, Zhao Zhang, Weili Zeng, Richong Zhang, Feng Zhu, and Rui Zhao. "Shikra: Unleashing Multimodal LLM's Referential Dialogue Magic." *arXiv preprint arXiv:2306.15195*, 2023.

[DAS21] Kunal Dahiya, Ananye Agarwal, Deepak Saini, K Gururaj, Jian Jiao, Amit Singh, Sumeet Agarwal, Purushottam Kar, and Manik Varma. "SiameseXML: Siamese Networks meet Extreme Classifiers with 100M Labels." In *International Conference on Machine Learning*, pp. 2330–2340. PMLR, 2021.

[DBK20] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale." In *International Conference on Learning Representations*, 2020.

[DCL18] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "Bert: Pre-training of deep bidirectional transformers for language understanding." *arXiv preprint arXiv:1810.04805*, 2018.

[DCM12] Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Marc'aurelio Ranzato, Andrew Senior, Paul Tucker, Ke Yang, et al. "Large scale distributed deep networks." *Advances in neural information processing systems*, **25**, 2012.

[DHS11] John Duchi, Elad Hazan, and Yoram Singer. "Adaptive subgradient methods for online learning and stochastic optimization." *Journal of machine learning research*, **12**(Jul):2121–2159, 2011.

[DKM06] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. "Our data, ourselves: Privacy via distributed noise generation."

In *Annual international conference on the theory and applications of cryptographic techniques*, pp. 486–503. Springer, 2006.

[DL09] Cynthia Dwork and Jing Lei. "Differential privacy and robust statistics." In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pp. 371–380, 2009.

[DLL23] Wenliang Dai, Junnan Li, Dongxu Li, Anthony Meng Huat Tiong, Junqi Zhao, Weisheng Wang, Boyang Li, Pascale Fung, and Steven Hoi. "InstructBLIP: Towards General-purpose Vision-Language Models with Instruction Tuning." In *NeurIPS*, 2023.

[DMN06] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. "Calibrating noise to sensitivity in private data analysis." In *Theory of cryptography conference*, pp. 265–284. Springer, 2006.

[Don19] Mark Dong. "Pytorch implementation of Optimization as a Model for Few-shot Learning." `https://github.com/markdtw/meta-learning-lstm-pytorch`, 2019.

[DR14] Cynthia Dwork, Aaron Roth, et al. "The algorithmic foundations of differential privacy." *Found. Trends Theor. Comput. Sci.*, **9**(3-4):211–407, 2014.

[DS10] Ofer Dekel and Ohad Shamir. "Multiclass-multilabel classification with more classes than examples." In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pp. 137–144. JMLR Workshop and Conference Proceedings, 2010.

[DSM21] Kunal Dahiya, Deepak Saini, Anshul Mittal, Ankush Shaw, Kushal Dave, Akshay Soni, Himanshu Jain, Sumeet Agarwal, and Manik Varma. "DeepXML: A Deep Extreme Multi-Label Learning Framework Applied to Short Text Documents." In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, pp. 31–39, 2021.

[DTN16] Christian Daniel, Jonathan Taylor, and Sebastian Nowozin. "Learning step size controllers for robust neural network training." In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.

[Dwo11] Cynthia Dwork. "A firm foundation for private data analysis." *Communications of the ACM*, **54**(1):86–95, 2011.

[DYY19] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. "Transformer-xl: Attentive language models beyond a fixed-length context." In *ACL*, 2019.

[DZL22] Tian Dong, Bo Zhao, and Lingjuan Lyu. "Privacy for Free: How does Dataset Condensation Help Privacy?" *arXiv preprint arXiv:2206.00240*, 2022.

[EIA18] Logan Engstrom, Andrew Ilyas, and Anish Athalye. "Evaluating and understanding the robustness of adversarial logit pairing." *arXiv preprint arXiv:1807.10272*, 2018.

[EMH18] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. "Neural architecture search: A survey." *arXiv preprint arXiv:1808.05377*, 2018.

[FAL17] Chelsea Finn, Pieter Abbeel, and Sergey Levine. "Model-agnostic meta-learning for fast adaptation of deep networks." In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1126–1135. JMLR. org, 2017.

[FFM19] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. "Slow-fast networks for video recognition." In *IEEE International Conference on Computer Vision*, pp. 6202–6211, 2019.

[FGI15] Hao Fang, Saurabh Gupta, Forrest Iandola, Rupesh K Srivastava, Li Deng, Piotr Dollár, Jianfeng Gao, Xiaodong He, Margaret Mitchell, John C Platt, et al. "From captions to visual concepts and back." In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1473–1482, 2015.

[FHK20] Soichiro Fujita, Tsutomu Hirao, Hidetaka Kamigaito, Manabu Okumura, and Masaaki Nagata. "SODA: Story oriented dense video captioning evaluation framework." In *ECCV*, 2020.

[FLG21] Tsu-Jui Fu, Linjie Li, Zhe Gan, Kevin Lin, William Yang Wang, Lijuan Wang, and Zicheng Liu. "Violet: End-to-end video-language transformers with masked visual-token modeling." *arXiv preprint arXiv:2111.12681*, 2021.

[FLG23] Tsu-Jui Fu, Linjie Li, Zhe Gan, Kevin Lin, William Yang Wang, Lijuan Wang, and Zicheng Liu. "An empirical study of end-to-end video-language transformers with masked visual modeling." In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 22898–22909, 2023.

[FPY16] Akira Fukui, Dong Huk Park, Daylen Yang, Anna Rohrbach, Trevor Darrell, and Marcus Rohrbach. "Multimodal compact bilinear pooling for visual question answering and visual grounding." *arXiv preprint arXiv:1606.01847*, 2016.

[FWX23] Yuxin Fang, Wen Wang, Binhui Xie, Quan Sun, Ledell Wu, Xinggang Wang, Tiejun Huang, Xinlong Wang, and Yue Cao. "Eva: Exploring the limits of masked visual representation learning at scale." In *CVPR*, 2023.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[GBP21] Nilesh Gupta, Sakina Bohra, Yashoteja Prabhu, Saurabh Purohit, and Manik Varma. "Generalized Zero-Shot Extreme Multi-label Learning." In *Proceed-*

ings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, pp. 527–535, 2021.

[GBR12] Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. "A kernel two-sample test." *The Journal of Machine Learning Research*, **13**(1):723–773, 2012.

[GGC22] Golnaz Ghiasi, Xiuye Gu, Yin Cui, and Tsung-Yi Lin. "Scaling open-vocabulary image segmentation with image-level labels." In *European Conference on Computer Vision*, pp. 540–557, 2022.

[GGL22a] Yuying Ge, Yixiao Ge, Xihui Liu, Dian Li, Ying Shan, Xiaohu Qie, and Ping Luo. "Bridging video-text retrieval with multiple choice questions." In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 16167–16176, 2022.

[GGL22b] Yuying Ge, Yixiao Ge, Xihui Liu, Jinpeng Wang, Jianping Wu, Ying Shan, Xiaohu Qie, and Ping Luo. "Miles: visual bert pre-training with injected language semantics for video-text retrieval." In *European Conference on Computer Vision*, pp. 691–708, 2022.

[GKC16] Ruiqi Guo, Sanjiv Kumar, Krzysztof Choromanski, and David Simcha. "Quantization based Fast Inner Product Search." In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics, AISTATS 2016, Cadiz, Spain, May 9-11, 2016*, volume 51 of *JMLR Workshop and Conference Proceedings*, pp. 482–490. JMLR.org, 2016.

[GKK19] Rong Ge, Sham M Kakade, Rahul Kidambi, and Praneeth Netrapalli. "The step decay schedule: A near optimal, geometrically decaying learning rate procedure for least squares." *arXiv preprint arXiv:1904.12838*, 2019.

[GKX18] Akhilesh Gotmare, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. “A closer look at deep learning heuristics: Learning rate restarts, warmup and distillation.” *arXiv preprint arXiv:1810.13243*, 2018.

[GLS16] Shixiang Gu, Timothy Lillicrap, Ilya Sutskever, and Sergey Levine. “Continuous deep q-learning with model-based acceleration.” In *International Conference on Machine Learning*, pp. 2829–2838. PMLR, 2016.

[GLT20] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. “Realm: Retrieval-augmented language model pre-training.” *arXiv preprint arXiv:2002.08909*, 2020.

[GMW19] Chuan Guo, Ali Mousavi, Xiang Wu, Daniel Niels Holtmann-Rice, Satyen Kale, Sashank J. Reddi, and Sanjiv Kumar. “Breaking the Glass Ceiling for Embedding-Based Classifiers for Large Output Spaces.” In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 4944–4954, 2019.

[GSA20] Valentin Gabeur, Chen Sun, Karteek Alahari, and Cordelia Schmid. “Multimodal transformer for video retrieval.” In *European Conference on Computer Vision*, pp. 214–229, 2020.

[GSR17] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. “Neural message passing for quantum chemistry.” In *International Conference on Machine Learning*, pp. 1263–1272. PMLR, 2017.

[GSS14] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. “Explaining and harnessing adversarial examples.” *arXiv preprint arXiv:1412.6572*, 2014.

[GSY17] Jiyang Gao, Chen Sun, Zhenheng Yang, and Ram Nevatia. “Tall: Temporal activity localization via language query.” In *ICCV*, 2017.

[GTS19] Neel Guha, Ameet Talwalkar, and Virginia Smith. "One-shot federated learning." *arXiv preprint arXiv:1902.11175*, 2019.

[GVC20] Tanmay Gupta, Arash Vahdat, Gal Chechik, Xiaodong Yang, Jan Kautz, and Derek Hoiem. "Contrastive learning for weakly supervised phrase grounding." In *European Conference on Computer Vision*, pp. 752–768, 2020.

[GYC21] Tianyu Gao, Xingcheng Yao, and Danqi Chen. "SimCSE: Simple Contrastive Learning of Sentence Embeddings." In *Empirical Methods in Natural Language Processing (EMNLP)*, 2021.

[Ham74] Frank R Hampel. "The influence curve and its role in robust estimation." *Journal of the american statistical association*, **69**(346):383–393, 1974.

[HBM22] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. "Training compute-optimal large language models." *arXiv preprint arXiv:2203.15556*, 2022.

[HDL21] Yuqi Huo, Mingyu Ding, Haoyu Lu, Nanyi Fei, Zhiwu Lu, Ji-Rong Wen, and Ping Luo. "Compressed video contrastive learning." In *Advances in Neural Information Processing Systems*, volume 34, pp. 14176–14187, 2021.

[HEG15] Fabian Caba Heilbron, Victor Escorcia, Bernard Ghanem, and Juan Carlos Niebles. "Activitynet: A large-scale video benchmark for human activity understanding." In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 961–970, 2015.

[HFW20] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross B. Girshick. "Momentum Contrast for Unsupervised Visual Representation Learning." In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pp. 9726–9735. IEEE, 2020.

[HGS20] Junxian He, Jiatao Gu, Jiajun Shen, and Marc'Aurelio Ranzato. "Revisiting Self-Training for Neural Sequence Generation." In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[HMX20] Shell Xu Hu, Pablo G Moreno, Yang Xiao, Xi Shen, Guillaume Obozinski, Neil D Lawrence, and Andreas Damianou. "Empirical Bayes Transductive Meta-Learning with Synthetic Gradients." *arXiv preprint arXiv:2004.12696*, 2020.

[HQB19] Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. "Measuring the effects of non-identical data distribution for federated visual classification." *arXiv preprint arXiv:1909.06335*, 2019.

[HSS12] Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. "Neural networks for machine learning lecture 6a overview of mini-batch gradient descent." 2012.

[HSW22] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. "LoRA: Low-Rank Adaptation of Large Language Models." In *ICLR*, 2022.

[HVP18] Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Ian Osband, et al. "Deep q-learning from demonstrations." In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[HWS17] Lisa Anne Hendricks, Oliver Wang, Eli Shechtman, Josef Sivic, Trevor Darrell, and Bryan Russell. "Localizing moments in video with natural language." In *ICCV*, 2017.

[HXZ20] Tengda Han, Weidi Xie, and Andrew Zisserman. "Self-supervised co-training for video representation learning." In *Advances in Neural Information Processing Systems*, volume 33, pp. 5679–5690, 2020.

[HXZ22] Tengda Han, Weidi Xie, and Andrew Zisserman. "Temporal alignment networks for long-term video." In *CVPR*, 2022.

[HZB19] Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. "Natural adversarial examples." *arXiv preprint arXiv:1907.07174*, 2019.

[HZR16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[ICH22] Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. "Contrastive Pre-training for Zero-Shot Information Retrieval.", 2022.

[JBC19] Himanshu Jain, Venkatesh Balasubramanian, Bhanu Chunduri, and Manik Varma. "Slice: Scalable Linear Extreme Classifiers Trained on 100 Million Labels for Related Searches." In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, WSDM 2019, Melbourne, VIC, Australia, February 11-15, 2019*, pp. 528–536. ACM, 2019.

[JCL20] Jianwen Jiang, Ziqiang Chen, Haojie Lin, Xibin Zhao, and Yue Gao. "Divide and conquer: Question-guided spatio-temporal contextual attention for video question answering." In *AAAI Conference on Artificial Intelligence*, volume 34, pp. 11101–11108, 2020.

[JCS18] Haoming Jiang, Zhehui Chen, Yuyang Shi, Bo Dai, and Tuo Zhao. "Learning to Defense by Learning to Attack." *arXiv preprint arXiv:1811.01213*, 2018.

[JDG23] Yunjie Ji, Yong Deng, Yan Gong, Yiping Peng, Qiang Niu, Lei Zhang, Baochang Ma, and Xiangang Li. “Exploring the impact of instruction data scaling on large language models: An empirical study on real-world use cases.” *arXiv preprint arXiv:2303.14742*, 2023.

[JH20] Pin Jiang and Yahong Han. “Reasoning with heterogeneous graph alignment for video question answering.” In *AAAI Conference on Artificial Intelligence*, volume 34, pp. 11109–11116, 2020.

[JZH19] Yunseok Jang, Tianchen Zhao, Seunghoon Hong, and Honglak Lee. “Adversarial Defense via Learning to Generate Diverse Attacks.” In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2740–2749, 2019.

[KB14] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization.” *arXiv preprint arXiv:1412.6980*, 2014.

[KCS17] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. “The kinetics human action video dataset.” *arXiv preprint arXiv:1705.06950*, 2017.

[KGB16a] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. “Adversarial examples in the physical world.” *arXiv preprint arXiv:1607.02533*, 2016.

[KGB16b] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. “Adversarial machine learning at scale.” *arXiv preprint arXiv:1611.01236*, 2016.

[KH09] Alex Krizhevsky, Geoffrey Hinton, et al. “Learning multiple layers of features from tiny images.” 2009.

[KHR17] Ranjay Krishna, Kenji Hata, Frederic Ren, Li Fei-Fei, and Juan Carlos Niebles.

"Dense-captioning events in videos." In *IEEE International Conference on Computer Vision*, pp. 706–715, 2017.

[KJG11] Hildegard Kuehne, Hueihan Jhuang, Estíbaliz Garrote, Tomaso Poggio, and Thomas Serre. "HMDB: a large video database for human motion recognition." In *IEEE International Conference on Computer Vision*, pp. 2556–2563, 2011.

[KKM20] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. "Scaffold: Stochastic controlled averaging for federated learning." In *International Conference on Machine Learning*, pp. 5132–5143. PMLR, 2020.

[KMA21] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. "Advances and open problems in federated learning." *Foundations and Trends® in Machine Learning*, **14**(1–2):1–210, 2021.

[KMH20] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. "Scaling laws for neural language models." *arXiv preprint arXiv:2001.08361*, 2020.

[KMR23] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. "Segment anything." In *ICCV*, 2023.

[KMY16] Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. "Federated learning: Strategies for improving communication efficiency." *arXiv preprint arXiv:1610.05492*, 2016.

[KNH10] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. “Cifar-10 (canadian institute for advanced research).” *URL http://www. cs. toronto. edu/kriz/cifar. html*, **8**, 2010.

[KNH14] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. “The cifar-10 dataset.” *online: http://www. cs. toronto. edu/kriz/cifar. html*, **55**, 2014.

[KOV15] Peter Kairouz, Sewoong Oh, and Pramod Viswanath. “The composition theorem for differential privacy.” In *International conference on machine learning*, pp. 1376–1385. PMLR, 2015.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks.” *Advances in neural information processing systems*, **25**, 2012.

[KT19] Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.” In *Proceedings of NAACL-HLT*, pp. 4171–4186, 2019.

[KTW20] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. “Supervised Contrastive Learning.” In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[KW16] Thomas N Kipf and Max Welling. “Semi-supervised classification with graph convolutional networks.” *arXiv preprint arXiv:1609.02907*, 2016.

[LAC21] Brian Lester, Rami Al-Rfou, and Noah Constant. “The Power of Scale for Parameter-Efficient Prompt Tuning.” In *EMNLP*, 2021.

[LBB98] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. "Gradient-based learning applied to document recognition." *Proceedings of the IEEE*, **86**(11):2278–2324, 1998.

[LBB21] Jie Lei, Tamara L Berg, and Mohit Bansal. "Detecting moments and highlights in videos via natural language queries." In *NeurIPS*, 2021.

[LCC20] Linjie Li, Yen-Chun Chen, Yu Cheng, Zhe Gan, Licheng Yu, and Jingjing Liu. "HERO: Hierarchical Encoder for Video+ Language Omni-representation Pre-training." In *Empirical Methods in Natural Language Processing*, pp. 2046–2065, 2020.

[LCG20] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. "ALBERT: A Lite BERT for Self-supervised Learning of Language Representations." In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[LCT19] Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. "Latent Retrieval for Weakly Supervised Open Domain Question Answering." In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 6086–6096, Florence, Italy, 2019. Association for Computational Linguistics.

[LDC22] Qinbin Li, Yiqun Diao, Quan Chen, and Bingsheng He. "Federated Learning on Non-IID Data Silos: An Experimental Study." In *IEEE International Conference on Data Engineering*, 2022.

[LeC98] Yann LeCun. "The MNIST database of handwritten digits." *http://yann. lecun. com/exdb/mnist/*, 1998.

[LH16] Ilya Loshchilov and Frank Hutter. "Sgdr: Stochastic gradient descent with warm restarts." *arXiv preprint arXiv:1608.03983*, 2016.

[LH17] Ilya Loshchilov and Frank Hutter. "Decoupled weight decay regularization." *arXiv preprint arXiv:1711.05101*, 2017.

[LH19] Ilya Loshchilov and Frank Hutter. "Decoupled Weight Decay Regularization." In *ICLR*, 2019.

[LHL17] Hyeungill Lee, Sungyeob Han, and Jungwoo Lee. "Generative adversarial trainer: Defense to adversarial perturbations with gan." *arXiv preprint arXiv:1705.03387*, 2017.

[LHQ20] Yandong Li, Di Huang, Danfeng Qin, Liqiang Wang, and Boqing Gong. "Improving object detection with selective self-supervised self-training." In *European Conference on Computer Vision*, pp. 589–607. Springer, 2020.

[LHW23] KunChang Li, Yinan He, Yi Wang, Yizhuo Li, Wenhai Wang, Ping Luo, Yali Wang, Limin Wang, and Yu Qiao. "Videochat: Chat-centric video understanding." *arXiv preprint arXiv:2305.06355*, 2023.

[LJD17] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. "Hyperband: A novel bandit-based approach to hyperparameter optimization." *The Journal of Machine Learning Research*, **18**(1):6765–6816, 2017.

[LJL17] Kaifeng Lv, Shunhua Jiang, and Jian Li. "Learning gradient descent: Better generalization and longer horizons." In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 2247–2255. JMLR. org, 2017.

[LJS20] Huaishao Luo, Lei Ji, Botian Shi, Haoyang Huang, Nan Duan, Tianrui Li, Jason Li, Taroon Bharti, and Ming Zhou. "Univl: A unified video and language pre-training model for multimodal understanding and generation." *arXiv preprint arXiv:2002.06353*, 2020.

[LL21] Xiang Lisa Li and Percy Liang. "Prefix-Tuning: Optimizing Continuous Prompts for Generation." In *ACL*, 2021.

[LLL22] Dongxu Li, Junnan Li, Hongdong Li, Juan Carlos Niebles, and Steven CH Hoi. "Align and prompt: Video-and-language pre-training with entity prompts." In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4953–4963, 2022.

[LLL23a] Dongxu Li, Junnan Li, Hung Le, Guangsen Wang, Silvio Savarese, and Steven C.H. Hoi. "LAVIS: A One-stop Library for Language-Vision Intelligence." In *ACL*, 2023.

[LLL23b] Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. "Improved Baselines with Visual Instruction Tuning." *arXiv preprint arXiv:2310.03744*, 2023.

[LLV20] Thao Minh Le, Vuong Le, Svetha Venkatesh, and Truyen Tran. "Hierarchical conditional relation networks for video question answering." In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9972–9981, 2020.

[LLW23] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. "Visual instruction tuning." In *NeurIPS*, 2023.

[LLZ21] Jie Lei, Linjie Li, Luowei Zhou, Zhe Gan, Tamara L Berg, Mohit Bansal, and Jingjing Liu. "Less is more: Clipbert for video-and-language learning via sparse sampling." In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7331–7341, 2021.

[LM16] Ke Li and Jitendra Malik. "Learning to optimize." *arXiv preprint arXiv:1606.01885*, 2016.

[LOG19] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov.

"Roberta: A robustly optimized bert pretraining approach." *arXiv preprint arXiv:1907.11692*, 2019.

[LST20] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. "Federated learning: Challenges, methods, and future directions." *IEEE Signal Processing Magazine*, **37**(3):50–60, 2020.

[LSZ20] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. "Federated optimization in heterogeneous networks." *Proceedings of Machine Learning and Systems*, **2**:429–450, 2020.

[LWS20] Jie Lei, Liwei Wang, Yelong Shen, Dong Yu, Tamara Berg, and Mohit Bansal. "MART: Memory-Augmented Recurrent Transformer for Coherent Video Paragraph Captioning." In *ACL*, 2020.

[LWW21] Qinbin Li, Zeyi Wen, Zhaomin Wu, Sixu Hu, Naibo Wang, Yuan Li, Xu Liu, and Bingsheng He. "A survey on federated learning systems: vision, hype and reality for data privacy and protection." *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[LXL19] Liangchen Luo, Yuanhao Xiong, Yan Liu, and Xu Sun. "Adaptive gradient methods with dynamic bound of learning rate." *arXiv preprint arXiv:1902.09843*, 2019.

[LXX22] Yuqi Liu, Pengfei Xiong, Luhui Xu, Shengming Cao, and Qin Jin. "Ts2-net: Token shift and selection transformer for text-video retrieval." In *European Conference on Computer Vision*, pp. 319–335, 2022.

[LYB20] Jie Lei, Licheng Yu, Tamara L Berg, and Mohit Bansal. "Tvr: A large-scale dataset for video-subtitle moment retrieval." In *ECCV*, 2020.

[LYY19] Liunian Harold Li, Mark Yatskar, Da Yin, Cho-Jui Hsieh, and Kai-Wei Chang. "Visualbert: A simple and performant baseline for vision and language." *arXiv preprint arXiv:1908.03557*, 2019.

[LZC23a] Bo Li, Yuanhan Zhang, Liangyu Chen, Jinghao Wang, Fanyi Pu, Jingkang Yang, Chunyuan Li, and Ziwei Liu. "Mimic-it: Multi-modal in-context instruction tuning." *arXiv preprint arXiv:2306.05425*, 2023.

[LZC23b] Kevin Qinghong Lin, Pengchuan Zhang, Joya Chen, Shraman Pramanick, Difei Gao, Alex Jinpeng Wang, Rui Yan, and Mike Zheng Shou. "UniVTG: Towards Unified Video-Language Temporal Grounding." In *ICCV*, 2023.

[MAS20] Antoine Miech, Jean-Baptiste Alayrac, Lucas Smaira, Ivan Laptev, Josef Sivic, and Andrew Zisserman. "End-to-end learning of visual representations from uncurated instructional videos." In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9879–9889, 2020.

[MCH20] Jonghwan Mun, Minsu Cho, and Bohyung Han. "Local-global video-text interactions for temporal grounding." In *CVPR*, 2020.

[McS09] Frank D McSherry. "Privacy integrated queries: an extensible platform for privacy-preserving data analysis." In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, pp. 19–30, 2009.

[Men23] Gaurav Menghani. "Efficient deep learning: A survey on making deep learning models smaller, faster, and better." *ACM Computing Surveys*, **55**(12):1–37, 2023.

[MFF16] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. "Deepfool: a simple and accurate method to fool deep neural networks." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2574–2582, 2016.

[MHW19] Tharun Medini, Qixuan Huang, Yiqiu Wang, Vijai Mohan, and Anshumali Shrivastava. “Extreme Classification in Log Memory using Count-Min Sketch: A Case Study of Amazon Search with 50M Products.” In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 13244–13254, 2019.

[MJW23] Fan Ma, Xiaojie Jin, Heng Wang, Jingjia Huang, Linchao Zhu, Jiashi Feng, and Yi Yang. “Temporal Perceiving Video-Language Pre-training.” *arXiv preprint arXiv:2301.07463*, 2023.

[MK23] Salman Khan Muhammad Maaz, Hanoona Rasheed and Fahad Khan. “Video-ChatGPT: Towards Detailed Video Understanding via Large Vision and Language Models.” *ArXiv 2306.05424*, 2023.

[MKK18] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. “Spectral Normalization for Generative Adversarial Networks.” In *International Conference on Learning Representations*, 2018.

[MKR22] Aaron Mueller, Jason Krone, Salvatore Romeo, Saab Mansour, Elman Mansimov, Yi Zhang, and Dan Roth. “Label Semantic Aware Pre-training for Few-shot Text Classification.” *arXiv preprint arXiv:2204.07128*, 2022.

[MMN18] Luke Metz, Niru Maheswaranathan, Jeremy Nixon, C Daniel Freeman, and Jascha Sohl-Dickstein. “Understanding and correcting pathologies in the training of learned optimizers.” *arXiv preprint arXiv:1810.10180*, 2018.

[MMN19] Luke Metz, Niru Maheswaranathan, Jeremy Nixon, Daniel Freeman, and Jascha Sohl-Dickstein. “Understanding and correcting pathologies in the training of learned optimizers.” In *International Conference on Machine Learning*, pp. 4556–4565, 2019.

[MMR17] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. "Communication-efficient learning of deep networks from decentralized data." In *Artificial intelligence and statistics*, pp. 1273–1282. PMLR, 2017.

[MMS17] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. "Towards deep learning models resistant to adversarial attacks." *arXiv preprint arXiv:1706.06083*, 2017.

[MPF22] Thomas Müller, Guillermo Pérez-Torró, and Marc Franco-Salvador. "Few-Shot Learning with Siamese Networks and Label Tuning." *arXiv preprint arXiv:2203.14655*, 2022.

[MRT17] H Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. "Learning differentially private recurrent language models." *arXiv preprint arXiv:1710.06963*, 2017.

[MS19] Purnendu Mishra and Kishor Sarawadekar. "Polynomial learning rate policy with warm restart for deep neural network." In *TENCON 2019-2019 IEEE Region 10 Conference (TENCON)*, pp. 2087–2092. IEEE, 2019.

[MSA21] Anshul Mittal, Noveen Sachdeva, Sheshansh Agrawal, Sumeet Agarwal, Purushottam Kar, and Manik Varma. "ECLARE: Extreme Classification with Label Graph Correlations." In *Proceedings of the Web Conference 2021*, pp. 3721–3732, 2021.

[MXS22] Yiwei Ma, Guohai Xu, Xiaoshuai Sun, Ming Yan, Ji Zhang, and Rongrong Ji. "X-clip: End-to-end multi-grained contrastive learning for video-text retrieval." In *ACM International Conference on Multimedia*, pp. 638–647, 2022.

[MY18] Yu A Malkov and Dmitry A Yashunin. “Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs.” *IEEE transactions on pattern analysis and machine intelligence*, **42**(4):824–836, 2018.

[MZA19] Antoine Miech, Dimitri Zhukov, Jean-Baptiste Alayrac, Makarand Tapaswi, Ivan Laptev, and Josef Sivic. “Howto100m: Learning a text-video embedding by watching hundred million narrated video clips.” In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2630–2640, 2019.

[NCL20] Timothy Nguyen, Zhourong Chen, and Jaehoon Lee. “Dataset Meta-Learning from Kernel Ridge-Regression.” In *International Conference on Learning Representations*, 2020.

[NNX21] Timothy Nguyen, Roman Novak, Lechao Xiao, and Jaehoon Lee. “Dataset distillation with infinitely wide convolutional networks.” *Advances in Neural Information Processing Systems*, **34**, 2021.

[NSS22] Arsha Nagrani, Paul Hongsuck Seo, Bryan Seybold, Anja Hauth, Santiago Manen, Chen Sun, and Cordelia Schmid. “Learning audio-video modalities from image captions.” In *European Conference on Computer Vision*, pp. 407–426, 2022.

[NW06] Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

[Ope23] OpenAI. “GPT-4 Technical Report.” *ArXiv*, **abs/2303.08774**, 2023.

[OWJ22] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F Christiano, Jan Leike, and Ryan Lowe. In *NeurIPS*, 2022.

[PAE16] Nicolas Papernot, Martín Abadi, Ulfar Erlingsson, Ian Goodfellow, and Kunal Talwar. "Semi-supervised knowledge transfer for deep learning from private training data." *arXiv preprint arXiv:1610.05755*, 2016.

[PAR20] AJ Piergiovanni, Anelia Angelova, and Michael S Ryoo. "Evolving losses for unsupervised video representation learning." In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 133–142, 2020.

[PHA20] Mandela Patrick, Po-Yao Huang, Yuki Asano, Florian Metze, Alexander Hauptmann, Joao Henriques, and Andrea Vedaldi. "Support-set bottlenecks for video-text representation learning." *arXiv preprint arXiv:2010.02824*, 2020.

[PKH18] Yashoteja Prabhu, Anil Kag, Shrutendra Harsola, Rahul Agrawal, and Manik Varma. "Parabel: Partitioned Label Trees for Extreme Classification with Application to Dynamic Search Advertising." In *Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018, Lyon, France, April 23-27, 2018*, pp. 993–1002. ACM, 2018.

[PRD19] Jae Sung Park, Marcus Rohrbach, Trevor Darrell, and Anna Rohrbach. "Adversarial inference for multi-sentence video description." In *CVPR*, 2019.

[PS21] Nicolas Papernot and Thomas Steinke. "Hyperparameter Tuning with Renyi Differential Privacy." *arXiv preprint arXiv:2110.03620*, 2021.

[PSP23] Clifton Poth, Hannah Sterz, Indraneil Paul, Sukannya Purkayastha, Leon Engländer, Timo Imhof, Ivan Vulić, Sebastian Ruder, Iryna Gurevych, and Jonas Pfeiffer. "Adapters: A unified library for parameter-efficient and modular transfer learning." *arXiv preprint arXiv:2311.11077*, 2023.

[PSY21] Tian Pan, Yibing Song, Tianyu Yang, Wenhao Jiang, and Wei Liu. “Videomoco: Contrastive video representation learning with temporally adversarial examples.” In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 11205–11214, 2021.

[PV14] Yashoteja Prabhu and Manik Varma. “FastXML: a fast, accurate and stable tree-classifier for extreme multi-label learning.” In *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD ’14, New York, NY, USA - August 24 - 27, 2014*, pp. 263–272. ACM, 2014.

[PWD23] Zhiliang Peng, Wenhui Wang, Li Dong, Yaru Hao, Shaohan Huang, Shuming Ma, and Furu Wei. “Kosmos-2: Grounding Multimodal Large Language Models to the World.” *arXiv preprint arXiv:2306.14824*, 2023.

[QLY22] Rui Qian, Yeqing Li, Liangzhe Yuan, Boqing Gong, Ting Liu, Matthew Brown, Serge Belongie, Ming-Hsuan Yang, Hartwig Adam, and Yin Cui. “Exploring temporal granularity in self-supervised video representation learning.” In *British Machine Vision Conference*, 2022.

[RBH21] Andrew Rouditchenko, Angie Boggust, David Harwath, Brian Chen, Dhiraj Joshi, Samuel Thomas, Kartik Audhkhasi, Hilde Kuehne, Rameswar Panda, Rogerio Feris, et al. “Avlnet: Learning audio-visual language representations from instructional videos.” In *Interspeech*, 2021.

[RCZ] Sashank J Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečnỳ, Sanjiv Kumar, and Hugh Brendan McMahan. “Adaptive Federated Optimization.” In *International Conference on Learning Representations*.

[RG19] Nils Reimers and Iryna Gurevych. “Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks.” In *Proceedings of the 2019 Conference on*

Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pp. 3982–3992, Hong Kong, China, 2019. Association for Computational Linguistics.

[RKY19] Sashank J. Reddi, Satyen Kale, Felix X. Yu, Daniel Niels Holtmann-Rice, Jiecao Chen, and Sanjiv Kumar. "Stochastic Negative Mining for Learning with Large Output Spaces." In *The 22nd International Conference on Artificial Intelligence and Statistics, AISTATS 2019, 16-18 April 2019, Naha, Okinawa, Japan*, volume 89 of *Proceedings of Machine Learning Research*, pp. 1940–1949. PMLR, 2019.

[RL17] Sachin Ravi and Hugo Larochelle. "Optimization as a model for few-shot learning." In *ICLR*, 2017.

[RM51] Herbert Robbins and Sutton Monro. "A stochastic approximation method." *The annals of mathematical statistics*, pp. 400–407, 1951.

[RMH20] Amirhossein Reisizadeh, Aryan Mokhtari, Hamed Hassani, Ali Jadbabaie, and Ramtin Pedarsani. "Fedpaq: A communication-efficient federated learning method with periodic averaging and quantization." In *International Conference on Artificial Intelligence and Statistics*, pp. 2021–2031. PMLR, 2020.

[ROG18] Konda Reddy Mopuri, Utkarsh Ojha, Utsav Garg, and R Venkatesh Babu. "NAG: Network for adversary generation." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 742–751, 2018.

[RRH16] Anna Rohrbach, Marcus Rohrbach, Ronghang Hu, Trevor Darrell, and Bernt Schiele. "Grounding of textual phrases in images by reconstruction." In *European Conference on Computer Vision*, pp. 817–834, 2016.

[RTR18] Mengye Ren, Eleni Triantafillou, Sachin Ravi, Jake Snell, Kevin Swersky, Joshua B Tenenbaum, Hugo Larochelle, and Richard S Zemel. "Meta-learning for semi-supervised few-shot classification." *arXiv preprint arXiv:1803.00676*, 2018.

[RXR19] Yangjun Ruan, Yuanhao Xiong, Sashank Reddi, Sanjiv Kumar, and Cho-Jui Hsieh. "Learning to Learn by Zeroth-Order Oracle." *arXiv preprint arXiv:1910.09464*, 2019.

[SB18] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[Sch15] Jürgen Schmidhuber. "Deep learning in neural networks: An overview." *Neural networks*, **61**:85–117, 2015.

[SCJ21] Yuqing Song, Shizhe Chen, and Qin Jin. "Towards diverse paragraph captioning for untrimmed videos." In *CVPR*, 2021.

[SGP20] Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter Battaglia. "Learning to simulate complex physics with graph networks." In *International Conference on Machine Learning*, pp. 8459–8468. PMLR, 2020.

[SHM16] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. "Mastering the game of Go with deep neural networks and tree search." *nature*, **529**(7587):484–489, 2016.

[SHR13] Andrew Senior, Georg Heigold, Marc'aurelio Ranzato, and Ke Yang. "An empirical study of learning rates in deep neural networks for speech recognition." In *2013 IEEE international conference on acoustics, speech and signal processing*, pp. 6724–6728. IEEE, 2013.

[SKC18] Pouya Samangouei, Maya Kabkab, and Rama Chellappa. “Defense-gan: Protecting classifiers against adversarial attacks using generative models.” *arXiv preprint arXiv:1805.06605*, 2018.

[SL14] Anshumali Shrivastava and Ping Li. “Asymmetric LSH (ALSH) for Sublinear Time Maximum Inner Product Search (MIPS).” In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pp. 2321–2329, 2014.

[SLA12] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. “Practical bayesian optimization of machine learning algorithms.” In *Advances in neural information processing systems*, pp. 2951–2959, 2012.

[SLJ20] Qianli Shen, Yan Li, Haoming Jiang, Zhaoran Wang, and Tuo Zhao. “Deep Reinforcement Learning with Smooth Policy.” *arXiv preprint arXiv:2003.09534*, 2020.

[SMV20] Prabhu Teja Sivaprasad, Florian Mai, Thijs Vogels, Martin Jaggi, and François Fleuret. “Optimizer benchmarking needs to account for hyperparameter tuning.” In *International Conference on Machine Learning*, pp. 9036–9045. PMLR, 2020.

[SNS21] Paul Hongsuck Seo, Arsha Nagrani, and Cordelia Schmid. “Look before you speak: Visually contextualized utterances.” In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 16877–16887, 2021.

[SRL20] Felipe Petroski Such, Aditya Rawal, Joel Lehman, Kenneth Stanley, and Jeffrey Clune. “Generative teaching networks: Accelerating neural architecture search by learning to generate synthetic training data.” In *International Conference on Machine Learning*, pp. 9206–9216. PMLR, 2020.

[SS21] Ilia Sucholutsky and Matthias Schonlau. “Soft-label dataset distillation and text dataset distillation.” In *2021 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE, 2021.

[SSG19] Arsalan Sharifnassab, Saber Salehkaleybar, and S Jamaloddin Golestani. “Order optimal one-shot distributed learning.” *Advances in Neural Information Processing Systems*, **32**, 2019.

[SSG21] Saber Salehkaleybar, Arsalan Sharifnassab, and S Jamaloddin Golestani. “One-shot federated learning: theoretical limits and algorithms to achieve them.” *Journal of Machine Learning Research*, **22**(189):1–47, 2021.

[SSS17] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. “Mastering the game of go without human knowledge.” *nature*, **550**(7676):354–359, 2017.

[SSZ14] Ohad Shamir, Nati Srebro, and Tong Zhang. “Communication-efficient distributed optimization using an approximate newton-type method.” In *International conference on machine learning*, pp. 1000–1008. PMLR, 2014.

[Sut92] Richard S Sutton. “Gain adaptation beats least squares.” In *Proceedings of the 7th Yale workshop on adaptive and learning systems*, volume 161168, 1992.

[SWD17] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. “Proximal policy optimization algorithms.” *arXiv preprint arXiv:1707.06347*, 2017.

[SWM19] Felix Sattler, Simon Wiedemann, Klaus-Robert Müller, and Wojciech Samek. “Robust and communication-efficient federated learning from non-iid data.” *IEEE transactions on neural networks and learning systems*, **31**(9):3400–3413, 2019.

[SXS22] Yuchong Sun, Hongwei Xue, Ruihua Song, Bei Liu, Huan Yang, and Jianlong Fu. "Long-form video-language pre-training with multimodal temporal contrastive learning." In *NIPS*, volume 35, pp. 38032–38045, 2022.

[SYA06] Nicol N Schraudolph, Jin Yu, Douglas Aberdeen, et al. "Fast online policy gradient learning with SMD gain vector adaptation." *Advances in neural information processing systems*, **18**:1185, 2006.

[SZ14] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." *arXiv preprint arXiv:1409.1556*, 2014.

[SZL13] Tom Schaul, Sixin Zhang, and Yann LeCun. "No more pesky learning rates." In *International Conference on Machine Learning*, pp. 343–351. PMLR, 2013.

[SZS12] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. "UCF101: A dataset of 101 human actions classes from videos in the wild." *arXiv preprint arXiv:1212.0402*, 2012.

[TBF14] Du Tran, Lubomir D Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. "C3D: generic features for video analysis." *CoRR, abs/1412.0767*, **2**:8, 2014.

[TH12] T. Tieleman and G. Hinton. "Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude." COURSERA: Neural Networks for Machine Learning, 2012.

[TLB21] Zineng Tang, Jie Lei, and Mohit Bansal. "DeCEMBERT: Learning from Noisy Instructional Videos via Dense Captions and Entropy Minimization." In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, June 2021.

[TLI23] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. "Llama: Open and efficient foundation language models." *arXiv preprint arXiv:2302.13971*, 2023.

[TMS23] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. "Llama 2: Open foundation and fine-tuned chat models." *arXiv preprint arXiv:2307.09288*, 2023.

[TRR21] Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. "BEIR: A Heterogeneous Benchmark for Zero-shot Evaluation of Information Retrieval Models." In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

[VBL16] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. "Matching networks for one shot learning." In *Advances in neural information processing systems*, pp. 3630–3638, 2016.

[VCS16] Ashwin K Vijayakumar, Michael Cogswell, Ramprasath R Selvaraju, Qing Sun, Stefan Lee, David Crandall, and Dhruv Batra. "Diverse beam search: Decoding diverse solutions from neural sequence models." *arXiv preprint arXiv:1610.02424*, 2016.

[vic23] "Vicuna." `https://github.com/lm-sys/FastChat`, 2023.

[VLP15] Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. "Cider: Consensus-based image description evaluation." In *CVPR*, 2015.

[VSP17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. “Attention is all you need.” *arXiv preprint arXiv:1706.03762*, 2017.

[VTD21] Arthur EW Venter, Marthinus W Theunissen, and Marelie H Davel. “Pre-interpolation loss behavior in neural networks.” In *Southern African Conference for Artificial Intelligence Research*, pp. 296–309. Springer, 2021.

[WCC23] Wenhai Wang, Zhe Chen, Xiaokang Chen, Jiannan Wu, Xizhou Zhu, Gang Zeng, Ping Luo, Tong Lu, Jie Zhou, Yu Qiao, et al. “Visionllm: Large language model is also an open-ended decoder for vision-centric tasks.” *arXiv preprint arXiv:2305.11175*, 2023.

[WGY23] Jinpeng Wang, Yixiao Ge, Rui Yan, Yuying Ge, Kevin Qinghong Lin, Satoshi Tsutsui, Xudong Lin, Guanyu Cai, Jianping Wu, Ying Shan, et al. “All in one: Exploring unified video-language pre-training.” In *CVPR*, pp. 6598–6608, 2023.

[WLL20] Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, and H Vincent Poor. “Tackling the objective inconsistency problem in heterogeneous federated optimization.” *Advances in neural information processing systems*, **33**:7611–7623, 2020.

[WLZ18] Donglai Wei, Joseph J Lim, Andrew Zisserman, and William T Freeman. “Learning and using the arrow of time.” In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8052–8060, 2018.

[WMH17] Olga Wichrowska, Niru Maheswaranathan, Matthew W Hoffman, Sergio Gomez Colmenarejo, Misha Denil, Nando de Freitas, and Jascha Sohl-Dickstein. “Learned optimizers that scale and generalize.” In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 3751–3760. JMLR. org, 2017.

[WRL18] Yuhuai Wu, Mengye Ren, Renjie Liao, and Roger Grosse. "Understanding short-horizon bias in stochastic meta-optimization." *arXiv preprint arXiv:1803.02021*, 2018.

[WSC23] Ziyang Wang, Yi-Lin Sung, Feng Cheng, Gedas Bertasius, and Mohit Bansal. "Unified Coarse-to-Fine Alignment for Video-Text Retrieval." In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2816–2827, 2023.

[WSM19] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. "GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding." 2019. In the Proceedings of ICLR.

[WY19] Huaxia Wang and Chun-Nam Yu. "A direct approach to robust deep learning using adversarial networks." *arXiv preprint arXiv:1905.09591*, 2019.

[WYC23] Haixin Wang, Xinlong Yang, Jianlong Chang, Dian Jin, Jinan Sun, Shikun Zhang, Xiao Luo, and Qi Tian. "Parameter-efficient Tuning of Large-scale Multimodal Foundation Model." In *NIPS*, 2023.

[WYK19] Yaqing Wang, Quanming Yao, James Kwok, and Lionel M Ni. "Generalizing from a few examples: A survey on few-shot learning." *arXiv preprint arXiv: 1904.05046*, 2019.

[WYS19] Hongyi Wang, Mikhail Yurochkin, Yuekai Sun, Dimitris Papailiopoulos, and Yasaman Khazaeni. "Federated Learning with Matched Averaging." In *International Conference on Learning Representations*, 2019.

[WZ19] Jianyu Wang and Haichao Zhang. "Bilateral adversarial training: Towards fast training of more robust models against adversarial attacks." In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 6629–6638, 2019.

[WZL21] Teng Wang, Ruimao Zhang, Zhichao Lu, Feng Zheng, Ran Cheng, and Ping Luo. "End-to-end dense video captioning with parallel decoding." In *ICCV*, 2021.

[WZP22] Kai Wang, Bo Zhao, Xiangyu Peng, Zheng Zhu, Shuo Yang, Shuo Wang, Guan Huang, Hakan Bilen, Xinchao Wang, and Yang You. "CAFE: Learning to Condense Dataset by Aligning Features." *arXiv preprint arXiv:2203.01531*, 2022.

[WZT18] Tongzhou Wang, Jun-Yan Zhu, Antonio Torralba, and Alexei A Efros. "Dataset distillation." *arXiv preprint arXiv:1811.10959*, 2018.

[WZY20] Teng Wang, Huicheng Zheng, Mingjing Yu, Qian Tian, and Haifeng Hu. "Event-centric hierarchical representation for dense video captioning." *IEEE Transactions on Circuits and Systems for Video Technology*, 2020.

[WZZ23] Teng Wang, Jinrui Zhang, Feng Zheng, Wenhao Jiang, Ran Cheng, and Ping Luo. "Learning Grounded Vision-Language Representation for Versatile Understanding in Untrimmed Videos." *arXiv preprint arXiv:2303.06378*, 2023.

[XDK19] Zhen Xu, Andrew M Dai, Jonas Kemp, and Luke Metz. "Learning an adaptive learning rate schedule." *arXiv preprint arXiv:1909.09712*, 2019.

[XDL18] Yilei Xiong, Bo Dai, and Dahua Lin. "Move forward and tell: A progressive generator of video descriptions." In *ECCV*, 2018.

[XDL22] Jiarui Xu, Shalini De Mello, Sifei Liu, Wonmin Byeon, Thomas Breuel, Jan Kautz, and Xiaolong Wang. "Groupvit: Semantic segmentation emerges from text supervision." In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 18134–18144, 2022.

[XGH21a] Hu Xu, Gargi Ghosh, Po-Yao Huang, Prahal Arora, Masoumeh Aminzadeh, Christoph Feichtenhofer, Florian Metze, and Luke Zettlemoyer. "VLM: Taskagnostic Video-Language Model Pre-training for Video Understanding." In *Findings of the Association for Computational Linguistics*, pp. 4227–4239, 2021.

[XGH21b] Hu Xu, Gargi Ghosh, Po-Yao Huang, Dmytro Okhonko, Armen Aghajanyan, Florian Metze, Luke Zettlemoyer, and Christoph Feichtenhofer. "VideoCLIP: Contrastive Pre-training for Zero-shot Video-Text Understanding." In *Empirical Methods in Natural Language Processing*, 2021.

[XH20] Yuanhao Xiong and Cho-Jui Hsieh. "Improved Adversarial Training via Learned Optimizer." *arXiv preprint arXiv:2004.12227*, 2020.

[XKG19] Cong Xie, Sanmi Koyejo, and Indranil Gupta. "Asynchronous federated optimization." *arXiv preprint arXiv:1903.03934*, 2019.

[XLH20] Qizhe Xie, Minh-Thang Luong, Eduard H. Hovy, and Quoc V. Le. "Self-Training With Noisy Student Improves ImageNet Classification." In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pp. 10684–10695. IEEE, 2020.

[XLL20] Yuanhao Xiong, Xuanqing Liu, Li-Cheng Lan, Yang You, Si Si, and Cho-Jui Hsieh. "How much progress have we made in neural network training? A New Evaluation Protocol for Benchmarking Optimizers." *arXiv preprint arXiv:2010.09889*, 2020.

[XLZ18] Chaowei Xiao, Bo Li, Jun-Yan Zhu, Warren He, Mingyan Liu, and Dawn Song. "Generating adversarial examples with adversarial networks." In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pp. 3905–3911, 2018.

[XMY16a] Jun Xu, Tao Mei, Ting Yao, and Yong Rui. "Msr-vtt: A large video description dataset for bridging video and language." In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5288–5296, 2016.

[XMY16b] Jun Xu, Tao Mei, Ting Yao, and Yong Rui. "MSR-VTT: A Large Video Description Dataset for Bridging Video and Language." IEEE International Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[XPE21] Mengmeng Xu, Juan-Manuel Pérez-Rúa, Victor Escorcia, Brais Martinez, Xiatian Zhu, Li Zhang, Bernard Ghanem, and Tao Xiang. "Boundary-sensitive pre-training for temporal localization in videos." In *IEEE International Conference on Computer Vision*, pp. 7220–7230, 2021.

[XPZ21] Mengmeng Xu, Juan Manuel Perez Rua, Xiatian Zhu, Bernard Ghanem, and Brais Martinez. "Low-fidelity video encoder optimization for temporal action localization." In *Advances in Neural Information Processing Systems*, volume 34, pp. 9923–9935, 2021.

[XQW17] Chang Xu, Tao Qin, Gang Wang, and Tie-Yan Liu. "Reinforcement learning for learning rate control." *arXiv preprint arXiv:1705.11159*, 2017.

[XRV17] Han Xiao, Kashif Rasul, and Roland Vollgraf. "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms." *arXiv preprint arXiv:1708.07747*, 2017.

[XWM19] Cihang Xie, Yuxin Wu, Laurens van der Maaten, Alan L Yuille, and Kaiming He. "Feature denoising for improving adversarial robustness." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 501–509, 2019.

[XZR20] Mengmeng Xu, Chen Zhao, David S Rojas, Ali Thabet, and Bernard Ghanem. "G-tad: Sub-graph localization for temporal action detection." In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 10156–10165, 2020.

[XZX17] Dejing Xu, Zhou Zhao, Jun Xiao, Fei Wu, Hanwang Zhang, Xiangnan He, and Yueting Zhuang. "Video question answering via gradually refined attention over appearance and motion." In *ACM International Conference on Multimedia*, pp. 1645–1653, 2017.

[YHC01] A Steven Younger, Sepp Hochreiter, and Peter R Conwell. "Meta-learning with backpropagation." In *IJCNN'01. International Joint Conference on Neural Networks. Proceedings (Cat. No. 01CH37222)*, volume 3. IEEE, 2001.

[YHO19] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. "Cutmix: Regularization strategy to train strong classifiers with localizable features." In *IEEE International Conference on Computer Vision*, pp. 6023–6032, 2019.

[YIL22] Xiang Yue, Huseyin A Inan, Xuechen Li, Girish Kumar, Julia McAnallen, Huan Sun, David Levitan, and Robert Sim. "Synthetic Text Generation with Differential Privacy: A Simple and Practical Recipe." *arXiv preprint arXiv:2210.14348*, 2022.

[YJC19] I Zeki Yalniz, Hervé Jégou, Kan Chen, Manohar Paluri, and Dhruv Mahajan. "Billion-scale semi-supervised learning for image classification." *arXiv preprint arXiv:1905.00546*, 2019.

[YLW19] Kaichao You, Mingsheng Long, Jianmin Wang, and Michael I Jordan. "How does learning rate decay help modern neural networks?" *arXiv preprint arXiv:1908.01878*, 2019.

[YLY18] Jiaxuan You, Bowen Liu, Rex Ying, Vijay Pande, and Jure Leskovec. "Graph convolutional policy network for goal-directed molecular graph generation." In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pp. 6412–6422, 2018.

[YNS23] Antoine Yang, Arsha Nagrani, Paul Hongsuck Seo, Antoine Miech, Jordi Pont-Tuset, Ivan Laptev, Josef Sivic, and Cordelia Schmid. "Vid2seq: Large-scale pretraining of a visual language model for dense video captioning." In *CVPR*, 2023.

[YQC22] Liangzhe Yuan, Rui Qian, Yin Cui, Boqing Gong, Florian Schroff, Ming-Hsuan Yang, Hartwig Adam, and Ting Liu. "Contextualized spatio-temporal contrastive learning with self-supervision." In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 13977–13986, 2022.

[YSG21] Rui Yan, Mike Zheng Shou, Yixiao Ge, Alex Jinpeng Wang, Xudong Lin, Guanyu Cai, and Jinhui Tang. "Video-text pre-training with learned regions." *arXiv preprint arXiv:2112.01194*, 2021.

[YSH21] Nishant Yadav, Rajat Sen, Daniel N. Hill, Arya Mazumdar, and Inderjit S. Dhillon. "Session-Aware Query Auto-Completion Using Extreme Multi-Label Ranking." In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, KDD '21, p. 3835–3844, New York, NY, USA, 2021. Association for Computing Machinery.

[YWQ22] Qihang Yu, Huiyu Wang, Siyuan Qiao, Maxwell Collins, Yukun Zhu, Hartwig Adam, Alan Yuille, and Liang-Chieh Chen. "k-means Mask Transformer." In *European Conference on Computer Vision*, pp. 288–307, 2022.

[YXN23] Shen Yan, Xuehan Xiong, Arsha Nagrani, Anurag Arnab, Zhonghao Wang, Weina Ge, David Ross, and Cordelia Schmid. "UnLoc: A Unified Framework for Video Localization Tasks." In *ICCV*, 2023.

[YZW19] Ronghui You, Zihan Zhang, Ziye Wang, Suyang Dai, Hiroshi Mamitsuka, and Shanfeng Zhu. "AttentionXML: Label Tree-based Attention-Aware Deep

Model for High-Performance Extreme Multi-Label Text Classification." In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 5812–5822, 2019.

[YZZ20] Hsiang-Fu Yu, Kai Zhong, Jiong Zhang, Wei-Cheng Chang, and Inderjit S Dhillon. "PECOS: Prediction for enormous and correlated output spaces." *Journal of Machine Learning Research*, 2020.

[ZB21a] Bo Zhao and Hakan Bilen. "Dataset condensation with differentiable siamese augmentation." In *International Conference on Machine Learning*, pp. 12674–12685. PMLR, 2021.

[ZB21b] Bo Zhao and Hakan Bilen. "Dataset Condensation with Distribution Matching." *arXiv preprint arXiv:2110.04181*, 2021.

[ZCS23] Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. "Minigpt-4: Enhancing vision-language understanding with advanced large language models." *arXiv preprint arXiv:2304.10592*, 2023.

[ZCY21] Jiong Zhang, Wei-cheng Chang, Hsiang-fu Yu, and Inderjit S Dhillon. "Fast Multi-Resolution Transformer Fine-tuning for Extreme Multi-label Text Classification." In *Advances in Neural Information Processing Systems*, 2021.

[ZGL20] Barret Zoph, Golnaz Ghiasi, Tsung-Yi Lin, Yin Cui, Hanxiao Liu, Ekin Dogus Cubuk, and Quoc Le. "Rethinking pre-training and self-training." *Advances in neural information processing systems*, **33**:3833–3845, 2020.

[ZK16] Sergey Zagoruyko and Nikos Komodakis. "Wide residual networks." *arXiv preprint arXiv:1605.07146*, 2016.

[ZKC19] Luowei Zhou, Yannis Kalantidis, Xinlei Chen, Jason J Corso, and Marcus Rohrbach. "Grounded video description." In *CVPR*, 2019.

[ZL16] Barret Zoph and Quoc V Le. "Neural architecture search with reinforcement learning." *arXiv preprint arXiv:1611.01578*, 2016.

[ZLB23] Hang Zhang, Xin Li, and Lidong Bing. "Video-LLaMA: An Instruction-tuned Audio-Visual Language Model for Video Understanding." *arXiv preprint arXiv:2306.02858*, 2023.

[ZLL22] Rowan Zellers, Jiasen Lu, Ximing Lu, Youngjae Yu, Yanpeng Zhao, Mohammadreza Salehi, Aditya Kusupati, Jack Hessel, Ali Farhadi, and Yejin Choi. "Merlot reserve: Neural script knowledge through vision and language and sound." In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 16375–16387, 2022.

[ZLX24] Chunting Zhou, Pengfei Liu, Puxin Xu, Srinivasan Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, et al. "Lima: Less is more for alignment." *NeurIPS*, **36**, 2024.

[ZMB20] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. "Dataset condensation with gradient matching." *arXiv preprint arXiv:2006.05929*, 2020.

[ZMK23] Yue Zhao, Ishan Misra, Philipp Krähenbühl, and Rohit Girdhar. "Learning Video Representations from Large Language Models." In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6586–6597, 2023.

[ZPF20] Songyang Zhang, Houwen Peng, Jianlong Fu, and Jiebo Luo. "Learning 2d temporal adjacent networks for moment localization with natural language." In *AAAI*, 2020.

[ZPM20] Yanlin Zhou, George Pu, Xiyao Ma, Xiaolin Li, and Dapeng Wu. “Distilled one-shot federated learning.” *arXiv preprint arXiv:2009.07999*, 2020.

[ZPT22] Wanrong Zhu, Bo Pang, Ashish V Thapliyal, William Yang Wang, and Radu Soricut. “End-to-end Dense Video Captioning as Sequence Generation.” In *COLING*, 2022.

[ZRU18] Chris Zhang, Mengye Ren, and Raquel Urtasun. “Graph hypernetworks for neural architecture search.” *arXiv preprint arXiv:1810.05749*, 2018.

[ZSC23] Shilong Zhang, Peize Sun, Shoufa Chen, Min Xiao, Wenqi Shao, Wenwei Zhang, Kai Chen, and Ping Luo. “Gpt4roi: Instruction tuning large language model on region-of-interest.” *arXiv preprint arXiv:2307.03601*, 2023.

[ZSJ20] Hao Zhang, Aixin Sun, Wei Jing, and Joey Tianyi Zhou. “Span-based Localizing Network for Natural Language Video Localization.” In *ACL*, 2020.

[ZSJ22] Qi Zhang, Yuqing Song, and Qin Jin. “Unifying event detection and captioning as sequence generation via pre-training.” In *ECCV*, 2022.

[ZVS18] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. “Learning transferable architectures for scalable image recognition.” In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8697–8710, 2018.

[ZXC18] Luowei Zhou, Chenliang Xu, and Jason J Corso. “Towards Automatic Learning of Procedures From Web Instructional Videos.” In *AAAI*, 2018.

[ZXH20] Runhao Zeng, Haoming Xu, Wenbing Huang, Peihao Chen, Mingkui Tan, and Chuang Gan. “Dense regression network for video grounding.” In *CVPR*, 2020.

[ZY20] Linchao Zhu and Yi Yang. "Actbert: Learning global-local video-text representations." In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8746–8755, 2020.

[ZYJ19] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric Xing, Laurent El Ghaoui, and Michael Jordan. "Theoretically Principled Trade-off between Robustness and Accuracy." In *International Conference on Machine Learning*, pp. 7472–7482, 2019.

[ZYW22] Can Zhang, Tianyu Yang, Junwu Weng, Meng Cao, Jue Wang, and Yuexian Zou. "Unsupervised pre-training for temporal action localization tasks." In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 14031–14041, 2022.

[ZZC18] Luowei Zhou, Yingbo Zhou, Jason J Corso, Richard Socher, and Caiming Xiong. "End-to-end dense video captioning with masked transformer." In *CVPR*, 2018.