Automated Data Engineering for Deep Learning

by

Mingjun Zhao

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

Doctor of Philosophy

in

Software Engineering and Intelligent Systems

Department of Electrical and Computer Engineering
University of Alberta

Abstract

Due to the advancement of computational hardware and the abundance and variety of data, deep learning has achieved significant success in natural language processing, computer vision, recommendation systems, and other domains in recent years. Because of this, recent large models have gained the ability to learn from massive amounts of data and are capable of handling tasks such as generating pictures based on user instructions or producing code with clear comments. However, effective exploitation of massive amounts of data is not a straightforward process, given that the quality and diversity of data can greatly influence the model performance.

Data engineering is the practice of enabling the collection and usage of data, and plays a critical role in making data accessible to deep learning by collecting, managing and converting raw data into useful information. Commonly used data engineering techniques in deep learning include data collection, filtering, preprocessing, cleaning, and modification. However, manually crafted data engineering policies require expert knowledge and a significant amount of tedious work, and can hardly reach the full potential of these techniques.

In order to avoid human intervention and unleash the potential of data, we automate the data engineering process to facilitate model learning and inference. In particular, we learn automated training curriculums and data augmentation strategies that enable the model to learn better from diverse and high-quality data samples. Moreover, we also improve the inference process by pruning and accelerating the redundant and unimportant part of input data to reduce computation cost. Specifically, we make the following contributions:

First, we focus on data selection by analyzing the problem of curriculum learning in neural machine translation (NMT) with the goal of improving a pre-trained NMT model. To achieve this, we propose a data selection framework based on reinforcement learning, which learns to re-select influential data samples from the original training set by identifying the most effective sample in a mini-batch. By simple fine-tuning, the selected subset of data can further improve the performance of the pre-trained model when original batch training reaches its ceiling, without utilizing additional new training data.

Second, we propose a label-aware auto-augmentation algorithm, to automatically learn augmentation policies separately for samples of different labels to overcome the limitation of sample-invariant augmentation. Our algorithm incorporates a predictor-based Bayesian optimizer to identify effective augmentations for each label, and constructs complementary augmentation policies based on minimum-redundancy maximum-reward principle. It produces effective label-aware augmentation policies which achieve significant performance boosts on image recognition tasks at a low search cost.

Third, we introduce a frame selection framework for the task of video action recognition to extract the most informative and representative frames to help a model better understand video content. We propose a Search-Map-Search learning paradigm which combines the advantages of heuristic search and supervised learning to select the best combination of frames from a video as one entity. By combining search with learning, the proposed method can better capture frame interactions while incurring a low inference overhead.

Finally, we study the embedding dimension pruning problem for recommendation systems. Specifically, we propose a low-cost embedding dimension search approach for recommender systems. By assessing information overlapping between the dimensions within each feature field and pruning unimportant and redundant dimensions progressively during model training via a two-level polarization regularizer, our method

efficiently reduces the model parameters, and achieves strong recommendation performance while introducing minimum overhead.

Preface

This dissertation proposes automated data engineering approaches to improve the learning and inference of deep models in various applications. Chapter 1 provides background information and motivations, while Chapter 2 reviews related research directions. The remaining sections of this thesis are the result of collaborative works with Dr. Di Niu and other co-authors.

Chapter 3 has been published as "Reinforced Curriculum Learning on Pre-trained Neural Machine Translation Models" by Zhao, Mingjun, et al. in the Proceedings of the AAAI Conference on Artificial Intelligence in 2020.

Chapter 4 has been published as "LA3: Efficient Label-Aware AutoAugment" by Zhao, Mingjun, et al. in the Proceedings of the European Conference on Computer Vision in 2022.

Chapter 5 has been published as "Search-Map-Search: A Frame Selection Paradigm for Action Recognition" by Zhao, Mingjun, et al. in the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition in 2023.

Chapter 6 is a collaborative work currently under submission.

Acknowledgements

The last five years of my graduate study at the University of Alberta have been an incredibly pleasant time in my life. During my time here, I have gained valuable knowledge and research experience. I have also had the pleasure of meeting many wonderful people from whom I have learned a great deal and received much support.

I would like to express my sincere gratitude to my supervisor, Dr. Di Niu, for his unwavering support and guidance throughout my graduate studies. He provided an excellent research environment where I received all the necessary assistance and encouragement to pursue my research goals. Di has not only provided me with academic instructions but also helped me navigate other challenges of life in general. He is both my mentor and friend. His mentorship has been invaluable to me, and I will always be grateful for his contributions to my education and personal growth.

I would also like to thank Professor Marek Reformat and Matthew Guzdial for being members of my supervisory committee, and Professor Lili Mou, Li Cheng and Zhan Shu for being members of my PhD candidacy exam committee. Their invaluable feedback on this work has helped me tremendously.

I have had an enjoyable life during the past five years, thanks to my wonderful friends, including but not limited to Chenglin Li, Bang Liu, Haolan Chen, Fred X. Han, Yakun Yu, Shan Lu, Qikai Lu, Liyao Jiang, Zhou Mu, Chang Dong, Zhuoran Chen, etc. I am also grateful to my colleges, Haijiang Wu, Songling Yuan, Ruichen Wang, Hao Fu, Chucheng Chen, Siyuan Qiu, Xiaoli Wang, Jian Ma, Chen Chen, Haonan Lu and so on for their company and support to my research works.

Thanks to my parents Hua Zhao and Rongrong Xin, my grandparents Xichong

Zhao and Shuqin Jiang and all my family members. Your love is what makes me strong. Finally, I thank Jing Zhang for her caring and support with love and understanding. She is my project manager and my cheerleader, my partner and my advisor, my friend and my lover. She is my present and I look forward to her being my future.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Deep learning is a subset of machine learning that uses deep artificial neural networks to model and solve complex problems. It has been around for decades but has gained popularity in recent years due to the availability of large datasets and powerful computing resources. The recent emergence of large generative models such as GPT-4 [1] and Stable Diffusion [2] has shown great capability in generating high-quality contents according to user instructions. Their successes are attributed to the large models' capability as well as the massive amounts of training data.

With the opportunity of deep models comes the challenge of effective and efficient utilization of the data. First of all, the quality and diversity of data samples can greatly affect the learning of models, as proven by many studies [3–5]. Therefore, how to best utilize the existing data to build the best model becomes an important question. To address this question, several directions have been explored. Data cleaning aims to eliminate noisy data samples that may damage the model performance. Data augmentation adds modified sample copies to the training data with the goal of improving the generalization ability. By actively selecting and synthesizing data samples to participate in the model learning process, deep learning models can achieve high accuracy and robustness in various applications.

Second, in many real-world scenarios such as video understanding and deep learning

recommendation models, the deployment of deep models often suffers from the high memory requirement and low inference speed due to the giant size of the model and high dimensional input data. One possible solution to alleviate this problem is to shrink the size of the models by utilizing more efficient networks derived using neural architecture search [6]. Another promising direction involves reducing the dimension of input samples by pruning the unimportant and redundant parts.

In this dissertation, we argue that automated data engineering can greatly benefit the learning and inference of deep models. Our work focuses on learning automated data engineering strategies, including curriculum learning, data augmentation, and input sample pruning. To this end, we develop novel methods for a variety of tasks and demonstrate our methods outperform existing ones.

1.2 Automated Data Engineering

Data engineering plays a critical role in both the model learning and model deployment of deep learning by extracting and organizing the useful information from raw data. Many data engineering techniques are widely adopted in modern deep learning applications, such as data collection, filtering, cleaning, preprocessing, augmentation, and modification. Through these techniques, the usage of data has been greatly improved in multiple aspects including quality, diversity and information density, which enables the deep models to function more effectively and efficiently.

However, most data engineering algorithms are constructed based on human experiences which involves tedious human efforts and are far from the optimized solution. Moreover, data engineering solutions to different tasks are often not transferable, leading to repetitive trail-and-error processes that are extremely time-consuming.

In this thesis, we propose automated data engineering which automates the building process of data engineering algorithms using automated search algorithms. By clearly defining the search space and objective and choosing appropriate optimization algorithms, we can efficiently find effective data engineering algorithms to improve

the performance and efficiency of deep models. Compared with hand-crafted methods, automated data engineering usually finds better solutions and can be generally applied to varies deep learning tasks.

We focus on constructing automated data engineering methods for different deep learning scenarios such as natural language processing, computer vision and recommender systems. Our works investigate research problems to optimize the training data for better model learning as well as reduce the dimension of input data samples for efficient model inference and deployment.

As an approach of improving the model training, curriculum learning, is pioneered by [7], which chooses what examples to present and in which order to present them to the learning algorithm. Our work mainly focuses on automatically finding an optimized training curriculum that can best improve a pre-trained model in the task of neural machine translation.

Data augmentation has proven to be an effective regularization technique that can improve the generalization of deep neural networks by adding modified copies of existing samples to increase the volume and diversity of training data. In our work, we study the automated learning of augmentation policies to achieve superior performance and generalization in image recognition tasks across different domains.

In order to reduce the computation required in video understanding, frame sampling is a common approach to reduce the redundant information of the input video by sampling a subset of frames from all video frames. We investigate the frame selection problem in video action recognition task by automatically extracting the informative and representative frames from the video to improve the model's performance.

Despite the differences in tasks and domains of our works in this thesis, we extensively incorporate a unified methodology to analyze and solve different data engineering tasks. Figure 1.1 shows the principle methodology used throughout different tasks in the thesis. The first step is to identify a research problem and formulate it as an automated data engineering task. Second, we will define the task objective, i.e.,

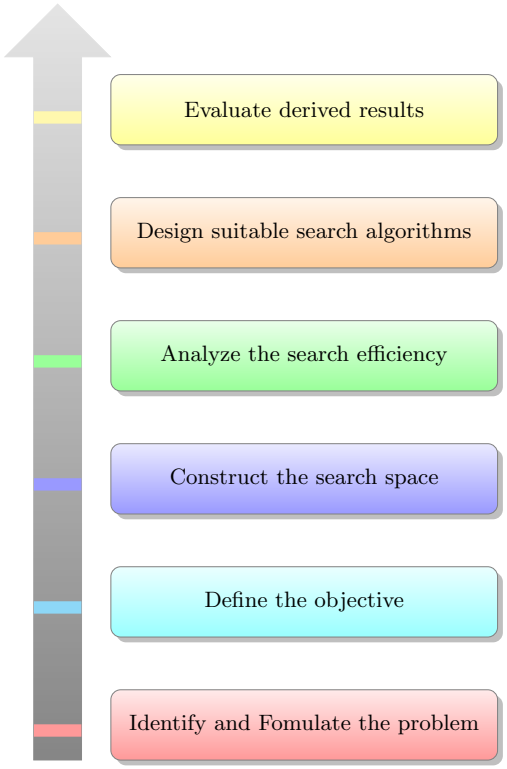


Figure 1.1: The principle methodology incorporated for all tasks in this thesis.

whether to improve the model performance or to save the computation cost. Third, based on the task information, we will construct the search space by defining all the possible candidates. Fourth, we will analyze the search cost for each iteration and design surrogate reward signals to improve search efficiency when the cost is high. After constructing the search space and designing the reward signal, we will carefully design suitable algorithms accordingly based on different task characteristics. Finally, the derived results will be evaluated to validate the effectiveness of our proposed method.

1.3 Contributions and Thesis Outline

We make contributions by proposing an organized problem setting for automated data engineering and showing its applications to a variety of NLP, CV and recommendation tasks with respect to different problems using automatically learned strategies. Figure 1.2 gives a demonstration of our works in this thesis. We can see that our works

Task	Downstream Task	Search Space	Reward Signal	Algorithm
curriculum learning	neural machine translation	entire training set	perplexity improvement	reinforcement learning
auto-augment	image recognition	augmentation operations triples	class accuracy difference	Bayesian optimization
frame selection	video action recognition	frames in videos	negative model loss	heuristic search + supervised learning
embedding dimension pruning	click-through rate prediction	embedding dimensions	negative model loss	gradient-based search

Figure 1.2: An overview of our works from the perspective of automated data engineering.

utilize different algorithms to handle different data engineering tasks across a wide range of domains. For examples, we formulate different data engineering problems of curriculum learning (data selection), augmentation selection, frame selection and embedding dimension selection as search problems, and design their search spaces, reward signals and algorithms accordingly.

In Chapter 2, we give a background introduction on the related research works. Specifically, we will introduce several aspects of the problem including the construction of search space, the search algorithms, and the evaluation strategy.

More specifically, from the perspective of automatically constructing better data engineering strategies to improve the learning and inference of deep models, we make the following contributions:

- In Chapter 3, we formulate the problem of curriculum learning in neural machine translation (NMT) as a reinforcement learning problem and propose a data selection framework to re-select influential data samples from the original training set. The selected subset of samples can further improve the pre-trained model by simple fine-tuning when original batch training reaches its ceiling, without utilizing additional new training data.

- In Chapter 4, we have also studied the automated learning of data augmentation policies. To overcome the limitation of previous sample-invariant augmentation, we propose a label-aware auto-augmentation algorithm, that learns augmentation policies separately for samples of different labels. We incorporate a predictor-based Bayesian optimizer and a complementary policy construction stage to efficiently produce label-aware augmentation policies.

- In Chapter 5, input data pruning is used to eliminate the redundant information and keep only important information of a sample, which can achieve better results with less computational power required. Therefore, we focus on the frame selection of video action recognition, propose a *Search-Map-Search* learning paradigm which combines the advantages of heuristic search and supervised learning to select the best combination of frames from a video as one entity.

- In Chapter 6, we design and implement an automated embedding dimension search algorithm for recommendation systems to reduce the model parameters while maintaining the strong recommendation performance. Our proposed method assesses information overlapping between dimensions within each feature field and prunes unimportant and redundant dimensions progressively during model training via a two-level polarization regularizer.

We conclude and provide potential future directions in Chapter 7.

Chapter 2

Background

Before we present our approaches of automated data engineering for different tasks, we will set up the context by giving a brief introduction on the topic of Automated Machine Learning (AutoML) that our methods are related to.

AutoML [8] is a process of using software tools and algorithms to automate some or all of the steps involved in building and deploying machine learning models, such as data processing, feature engineering, model selection, hyperparameter tuning, and model evaluation. It eliminates the need for professional expertise and can usually find better solutions than humans. Normally, the design of an AutoML system includes identifying and constructing the search space, choosing appropriate optimization algorithms, and designing efficient evaluation process.

The search space defines the design principles of the task, and its construction is usually task-specific. For example, the search space of Neural Architecture Search (NAS) [6] corresponds to different design choices of the network architecture, while the data augmentation search space includes combinations of different augmentation operations.

The optimization algorithms define how to guide the search to efficiently find the optimal candidate and should be carefully chosen based on the characteristics of the task and the search space. Popular optimization algorithms include random search [9], Bayesian optimization [10], evolutionary algorithm [11], reinforcement learning

[12] and gradient-based search [13].

Random search selects the best point from a set of randomly drawn points, while Bayesian optimization is a sequential design strategy for global optimization of black-box functions that does not assume any functional forms, usually employed to optimize expensive-to-evaluate functions. The evolutionary algorithm is a metaheuristic optimization algorithm with high scalability that takes inspiration from biological evolution. Reinforcement learning is also a popular choice in AutoML, yielding excellent results in many tasks. However, directly applying RL is very expensive in computation. Gradient-based search as pioneered by DARTS [14] searches over continuous and differentiable search space with high efficiency.

The evaluation process provides an assessment on the performance of each search candidate. Naive evaluation involves model training process which can be highly costly. Multiple efficient evaluation approaches have been proposed such as using smaller networks, reducing the data scale, network weight sharing, training surrogate networks to provide evaluation approximation, and evaluation early stopping.

Part I

Automated Data Selection and Augmentation for Improved Model Learning

Chapter 3

Reinforced Curriculum Learning

3.1 Introduction

Curriculum learning, as pioneered by [7], aims to improve the training of machine learning models by choosing what examples to present and in which order to present them to the learning algorithm. Curriculum learning was originally inspired by the learning experience of humans [15–17]—humans tend to learn better and faster when they are first introduced to simpler concepts and exploit previously learned concepts and skills to ease the learning of new abstractions. This phenomenon is widely observed in, e.g., music and sports training, academic training and pet shaping. Without surprise, curriculum learning is found most helpful in end-to-end neural network architectures [7], since the performance that an artificial neural network can achieve critically depends on the quality of training data presented to it.

Neural Machine Translation [18] [19] (NMT) translates text from a source language to a target language in an end-to-end fashion with a single neural network. It has not only achieved state-of-the-art machine translation results, but also eliminated hand-crafted features and rules that are otherwise required by statistical machine translation. The performance of NMT has been improved significantly in recent years, as the NMT architectures evolved from the initial RNN-based models [19] to convolutional seq2seq models [20] and further to Transformer models [21].

However, since obtaining accurately labeled training samples in machine transla-

Example 1	
zh	xianzai ta zheng kaolv zhe huijia.
zh-gloss	Now he is thinking about going home.
en	He is thinking about going home now.
Example 2	
zh	wo yao chi niupai.
zh-gloss	I want eat steak
en	I want a steak. Get me a coke.

Table 3.1: Examples with accurate/inaccurate translations.

tion is often time-consuming and requires expert knowledge, an important question in NMT is how to best utilize a limited number of available training samples, perhaps with different lengths, qualities, and noise levels. Recently, the application of curriculum learning is also studied for NMT. [22] propose to feed data to an NMT model in an easy-to-difficult order and characterize the "difficulty" of a training example by the sentence length and the rarity of words that appear in it. Other than using the straightforward difficulty or complexity as a criterion for curriculum design, [23] propose a method to calculate the noise level of a training example with the help of an additional trusted clean dataset and train an NMT model in a noise-annealing curriculum.

A limitation of the existing curriculum learning methods for NMT is that they only address the batch selection issue in a "learn-from-scratch" scenario. Unfortunately, training an NMT model is a time-consuming task and sometimes could take up to several weeks [24], depending on the amount of data available. In most practical and commercial cases, a pre-trained model often already exists, while re-training it from scratch with a new ordering of batches is a waste of time and resources. In this chapter, we study curriculum learning for NMT from a new perspective, that is given a pre-trained model and the dataset used to train it, to re-select a subset of useful

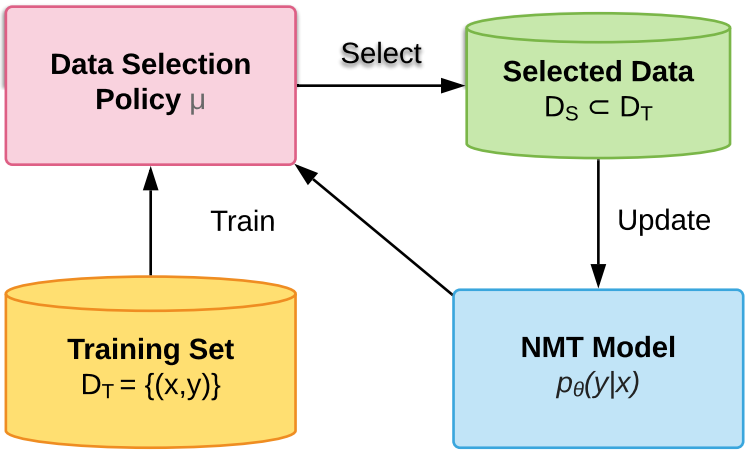


Figure 3.1: Illustration of the curriculum learning process. The RL policy μ is used to select samples D_S from the training set D_T. The selected data D_S is used to update a pre-trained NMT model $p_\theta(y|x)$.

samples from the existing dataset to further improve the model. Unlike the easy-to-difficult insights in traditional curriculum learning [7], [22], our idea is analogous to classroom training where a student first attends classes to learn general subjects with equal weights and then carefully reviews a subset of selected subjects to strengthen his/her weak aspects or to elevate ability in a field of interest.

Furthermore, while all the samples participate in batch training for the same number of epochs, it is unlikely that all data contribute equally to a best-performing model. Table 3.1 shows an example of two data samples from the dataset used in this chapter, where Example 1 is accurately translated and can potentially improve the model better, while Example 2 is poorly translated (with unexpected words in target sentence) and may even cause performance degradation when fed to the model. The objective of our curriculum design is to identify examples from the existing dataset that may further contribute to model improvement and present them again to the NMT learning system. An overview of our proposed task is given in Figure 3.1, where useful data can be selected and fed to the system repeatedly to strengthen the model iteratively.

We formulate the data re-selection task as a reinforcement learning problem where the state is the features of b randomly sampled training examples, the action is choosing one of them, and the reward is the perplexity difference on a validation set after the pre-trained model is updated with the selected sample. Thus, the primary goal of the learning problem becomes searching for a data selection policy to maximize the reward. Reinforcement learning is known to be unstable or even to diverge when the action-value function is represented by a nonlinear function, e.g., a neural network. For the sake of alleviating instability, our proposed RL framework is built based on the Deterministic Actor-Critic algorithm [25]. It consists of two networks, an actor network which learns a data selection policy, and a critic network which evaluates the action value of choosing each training sample while providing information to guide the training of the actor network. Besides introducing the framework, we also carefully design the state space by choosing a wide range of features to characterize each sample in multiple dimensions, including the sentence length, sentence-level log-likelihood, n-gram rarity, and POS and NER tagging.

Experiments on multiple translation datasets demonstrate that our method can achieve a significant performance boost, when normal batch learning cannot improve the model further, by only re-selecting influential samples from the original dataset. Furthermore, the proposed scheme outperforms a number of other curriculum learning baseline methods, including the denoising scheme based on the use of additional trusted data [23].

3.2 Related Work

High-quality machine translation corpus is costly and difficult to collect, thus it is necessary to make the best use of the corpus at hand. A straightforward method to achieve this goal is to remove the noisy samples in the training data, and train an new model with the clean ones. Unfortunately, it is hard to estimate the quality of a parallel sentence in the absence of golden reference [26]. Moreover, [23] find that

some of the noisy samples may yield some benefits to the model performance. Then they define a new method of computing noise level of a data example with the help of an extra trusted dataset and propose to train an NMT model in a noise-annealing curriculum.

Curriculum learning aims to organize the training process in a meaningful way by feeding certain samples to the model in certain training stage such that the model can learn better and faster [7]. They propose a simple strategy which organizes all training samples into bins of similar complexity and starts training from the easiest bin to include more complexed bins until all bins are covered. [27] apply this idea to NMT by binning according to simple features like sentence length and word frequency, and improve this strategy by restricting that each sample can only be trained once during an epoch. [28] conduct empirical studies on several hand-crafted curriculum and adopt a probabilistic view of curriculum learning. [22] further propose a competence function $c(t)$ with respect to training time step t as the indicator of learning progress and select samples based on both difficulty and competence. However, these heuristic-based approaches highly depend on hand-crafted curriculum and are hard to generalize.

Compared with heuristic-based approaches, RL-based policy learning models are trained end-to-end and do not rely on hand-crafted strategies. [29] use Bayesian optimization to learn a linear model for ranking examples in a work-embedding task. [30] explore bandit optimization for scheduling tasks in a multi-task problem.[31] select examples in a co-trained classifier using RL. [32] organize the dataset into bins based on the data noise proposed by [23] and utilize a DQN to learn a data selection policy deciding from which bin to select the next batch.

Different from the existing curriculum learning methods, our work focuses on learning a training curriculum with reinforcement learning for an existing pre-trained NMT model. We argue that existing curriculum learning methods are only applicable on train-from-scratch scenarios, and learning from an existing model can save training

time.

Active learning [33] is another related area which focuses on selectively obtaining labels for unlabeled data in order to improve the model with least labeling cost. [34] [35] study active learning for Statistical Machine Translation using some hard-coded heuristics. [36] design an active learning algorithm based on a deep Q-network, in which the action corresponds to binary annotation decisions applied to a stream of data. [37] make use of imitation learning to train a data selection policy.

3.3 Problem Definition

In this section, we provide a brief background of NMT and formulate the curriculum learning task on pre-trained NMT models as a reinforcement learning problem.

Machine Translation can be considered a one-to-one mapping from a source sentence x to a target sentence y. In neural machine translation, a model parameterized by θ is searched for to maximize the conditional probability $p_\theta(y|x)$ over the training samples. Modern NMT models adopt an encoder-decoder architecture where an encoder encodes the source sentence x into a hidden representation h and a decoder predicts the next target word y_i given the hidden vector h and all previously predicted words $\{y_1, ..., y_{i-1}\}$. Thus the conditional probability is decomposed as

$$\log p_\theta(y|x) = \sum_{i=1}^{L} \log p_\theta(y_i|y_{<i}, h), \tag{3.1}$$

where L is the number of tokens in each target sentence. Given a training corpus D_T, the training objective of an NMT model is to minimize

$$J(\theta) = \sum_{(x,y)\in D_T} -\log p_\theta(y|x). \tag{3.2}$$

We consider curriculum learning on a pre-trained NMT model, where the goal is to improve an existing model $p_\theta(y|x)$ by selecting a subset D_S from the training dataset D_T that led to $p_\theta(y|x)$. As compared to training from scratch, we take advantage of both the versatility of normal batch learning in the initial pre-training stage and

a carefully selected curriculum for targeted model improvements. Specifically, our objective is to find an optimal policy μ_ϕ to select D_S from D_T and update $p_\theta(y|x)$ with D_S such that the performance of the updated model is maximized, i.e.,

$$\begin{aligned} \max_{\phi} \quad & \mathit{perf}\Big(p_{\theta'}(y|x,\phi)\Big), \\ \text{s.t.} \quad & p_{\theta'}(y|x,\phi) = \mathit{train}\Big(p_\theta(y|x), D_S(\phi)\Big), \\ & D_S(\phi) = \mu_\phi(D_T), \end{aligned} \tag{3.3}$$

where $D_S(\phi)$ is a subset selected from D_T using policy μ_ϕ, $p_{\theta'}(y|x,\phi)$ is an updated model after training $p_\theta(y|x)$ with $D_S(\phi)$, and *perf* indicates the performance of a model, e.g., measured by BLEU or Perplexity.

The main challenge is to identify and select the most beneficial data samples from D_T. A naive way is to evaluate the BLEU improvement on a validation set brought by every single data sample in the training set and select the ones that improve the BLEU the most. However, this method is extremely costly and is not scalable to large datasets.

To obtain a generalizable data selection policy, we formulate the task as a reinforcement learning problem in which the environment is composed of both the dataset D_T and the model $p_\theta(y|x)$. The RL agent aims to learn a policy μ_ϕ which decides which sample to select when presented with a random batch of samples. In our framework, a state s corresponds to the representation of both a data batch to select from and the NMT model, a refers to the action of selecting the best data sample from the batch, and r is the performance improvement of the NMT model on validation set after being updated with the selected sample.

The RL agent is trained through interacting with the environment by repeatedly performing the following: 1) receiving a state s containing a random batch of samples, 2) providing an action a back to the environment according to its trained policy, and 3) updating the policy using a feedback reward r given by the environment. Once the policy is trained, it can be used to select data from an arbitrarily large dataset and is scalable.

3.4 Methods

In this section, we describe our Deterministic Actor-Critic framework for curriculum learning, as well as the design of the state, action, and reward in detail.

3.4.1 Model Overview

For the model design of the RL agent, we choose the Deterministic Actor-Critic algorithm.

Actor-critic [38] is a widely used method in reinforcement learning combining both an actor network μ_ϕ that outputs an action $a = \mu_\phi(s)$ given a state s to maximize the reward, and a critic network Q_w that predicts the action value $Q_w(s, a)$ of a state-action pair (s, a). The actor learns a near-optimal policy via policy gradient, while the critic estimates the action-value guiding the update direction of the actor. Compared with actor-only methods like REINFORCE [39], the existence of the critic reduces the update variance and accelerates convergence.

As our reward calculation involves evaluating the updated NMT model on a validation set and is thus expensive, we exploit a memory replay buffer to increase sample efficiency. Furthermore, we choose a deterministic policy setting as opposed to stochastic policy due to the fact that the deterministic policy gradient can be calculated much more efficiently as shown in [25].

The update of the framework is illustrated in Figure 3.2. The critic network Q_w takes a state-action pair (s, a), evaluates the action value, and outputs a predicted reward $\tilde{r} = Q_w(s, a)$, and updates the parameters supervised by the actual reward r from the environment. Note that the critic network provides an immediate reward per iteration. As a result, we do not need to employ additional techniques, e.g., Temporal-Difference (TD) [40], to approximate the long-term reward. The objective of the critic network is thus to minimize the squared error of δ between the predicted

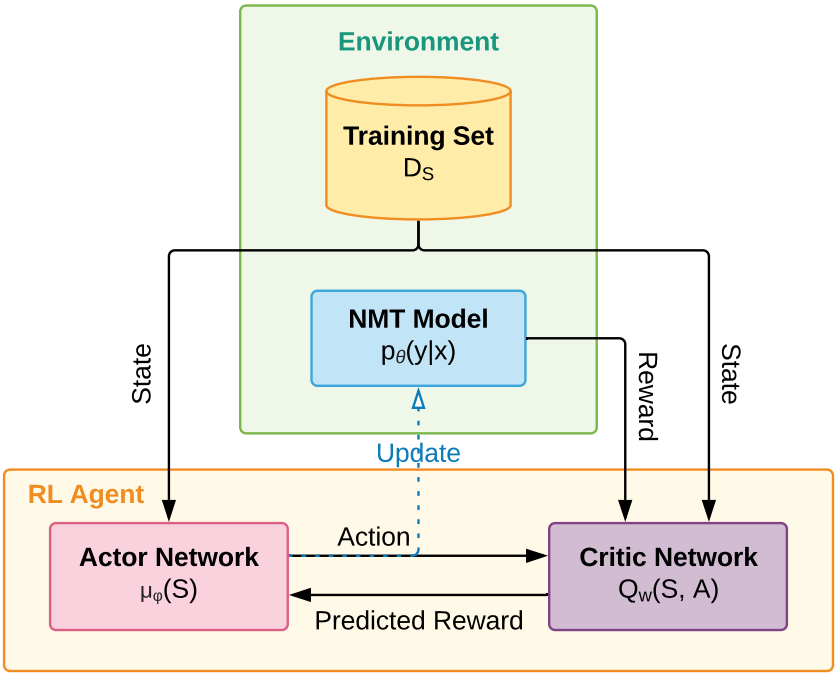


Figure 3.2: The proposed RL framework.

reward and the actual reward, given by

$$\delta_t = r_t - \tilde{r}_t = r_t - Q_w(s_t, a_t). \tag{3.4}$$

The update of parameters w is achieved through gradient descent as follows:

$$w_{t+1} = w_t + \alpha_w \delta_t \nabla_w Q_w(s_t, a_t). \tag{3.5}$$

where α_w corresponds to the learning rate of parameters w.

The actor network μ_ϕ takes in a state s from the environment, applies the learned policy, and outputs a corresponding action a. The objective of the actor network is to learn an optimal policy generating the proper action to maximize the predicted reward $Q_w(s, a)$. Policy gradient is used to update the parameters ϕ, i.e.,

$$\begin{aligned} \phi_{t+1} &= \phi_t + \alpha_\phi \nabla_\phi Q_w(s_t, a_t)|_{a=\mu_\phi(s)} \\ &= \phi_t + \alpha_\phi \nabla_\phi \mu_\phi(s_t) \nabla_a Q_w(s_t, a_t)|_{a=\mu_\phi(s)}. \end{aligned} \tag{3.6}$$

Algorithm 1 summarizes the overall learning process of the proposed framework. In each round of data selection (with K rounds in total), we first train the RL agent

Algorithm 1: The Proposed Method

Input: Training set D_T and an NMT model $p_{\theta_0}(y|x)$
Output: A better performing NMT model $p_{\theta_K}(y|x)$

1 **for** $t = 0, \ldots, K-1$ **do**
2 **for** *number of RL training iterations* **do**
3 Sample b examples from D_T and form state s
4 Generate action $a = \mu_\phi(s)$
5 Compute predicted reward $\tilde{r} = Q_w(s, a)$
6 Update $p_{\theta_k}(y|x)$ with selected sample to get $p_{\theta'_k}(y|x)$
7 Calculate the perplexity difference on validation set between $p_{\theta_k}(y|x)$ and $p_{\theta'_k}(y|x)$ as reward r
8 Update Q_w using Eq. (3.5)
9 Update μ_ϕ using in Eq. (3.6)
10 Select data D_S from D_T using μ_ϕ
11 Update $p_{\theta_k}(y|x)$ with D_S to get $p_{\theta_{k+1}}(y|x)$

for an adequate number of iterations. In each iteration, we derive a state, an action, and a reward in lines 4–7 and update the actor network and the critic network in line 8 and line 9, respectively. After the RL agent is fully trained, we apply the learned policy to select a subset D_S and use D_S to update the NMT model $p_{\theta_k}(y|x)$ and move to the next round. Usually one or two rounds are sufficient.

Figure 3.3 demonstrates the network structure of the RL agent. A *feature network* is shared between the Actor Network μ_ϕ and the Critic Network Q_w. It takes in the raw features of the sampled batch of b examples, where each feature of each sample goes through an independent single-layer MLP. The concatenation of the outputs constitutes the state representation s. Note that different examples in the sampled batch share the same network weights.

The Actor Network μ_ϕ is composed of a two-layer MLP and computes a score for each example in the sampled batch of b examples based on the input state s, and outputs the action a as a probability vector representing the probability of each example being selected, by taking a softmax operation over the computed scores of b examples.

The Critic Network Q_w also has two layers and calculates the action value of a

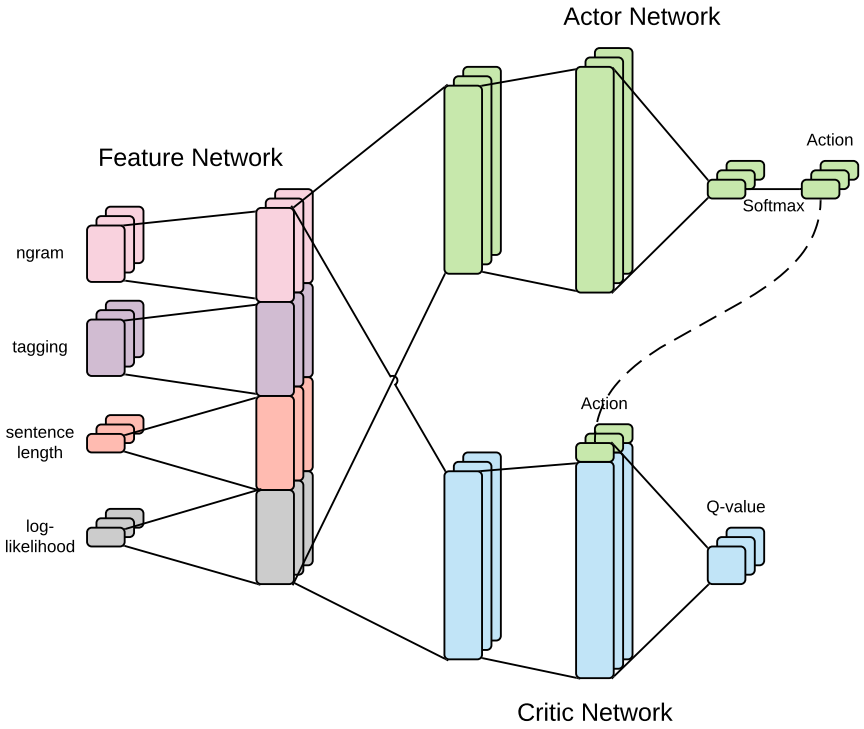


Figure 3.3: The network structure of the RL agent, where the depth represents a batch (of 3 samples in this illustration).

given state-action pair (s, a), where the action a is the output of the Actor Network, i.e., $a = \mu_\phi(s)$, and is concatenated to the second layer of the critic network.

Note that although the feature network is shared between both the actor network μ_ϕ and the critic network Q_w, we only update it with the critic network to reduce training instability.

3.4.2 State

The state s is meant to be a full summarization of the environment including a data batch of b examples to select from and information about the pre-trained model. However, the number of parameters in the pre-trained model is too large to be included in the state directly at each time step. Thus, we need to find a representation that can represent both the samples to be selected and the existing model using a limited

number of parameters. In our state design, we focus on three different dimensions, namely *informativeness*, *uncertainty*, and *diversity*. We use the sentence length as a measure for informativeness, the sentence-level log-likelihood for uncertainty, and the n-gram rarity together with NER and POS taggings for diversity.

The most intuitive representation feature of a parallel sentence is the sentence length, i.e. the number of tokens in a sentence. This simple scalar roughly measures the amount of information involved in a sentence and is also used in [22] as a "difficulty" estimate.

Following the intuition that examples yielding large uncertainty can benefit the model performance [41], another feature we use is the sentence-level log-likelihood $L_{p_\theta(y|x)}$ calculated by the pre-trained model $p_\theta(y|x)$ by summing up the log probability of each word y_i of the target sentence:

$$L_{p_\theta(y|x)} = \log p_\theta(y|x) = \sum_{y_i \in Y} \log p_\theta(y_i|x). \tag{3.7}$$

[42] and [43] suggest that selecting samples that are farthest away from the existing dataset can benefit model training. Incorporating this idea, we further utilize two other feature vectors, n-gram rarity and taggings, to represent the similarity between a given sample and the entire training set.

For n-gram rarity, we use $n = \{1, 2, 3, 4\}$. Given all the sentences in D_T, we define the relative frequency for a unique n-gram $g_j^{(n)}$ in D_T as

$$f^{(n)}(g_j^{(n)}) \triangleq \frac{1}{N^{(n)}} \sum_{s_k \in D_T} \sum_{g_i^{(n)} \in g_{(s_k)}^{(n)}} \mathbb{1}_{g_i^{(n)} = g_j^{(n)}}, \tag{3.8}$$

where $j = 1, ..., \#\{\textit{unique n-grams}\}$, s_k is a sentence from D_T, $g_{(s_k)}^{(n)}$ is all n-grams in s_k, and $N^{(n)}$ is the total number of n-grams in D_T. For n-gram representation of a sentence s_k, we form all the n-gram frequencies into a vector:

$$F^{(n)}(s_k) = \{f(g_1^{(n)}), ..., f(g_L^{(n)}) | g_l^{(n)} \in g_{(s_k)}^{(n)}\}. \tag{3.9}$$

In this chapter, we use $n = \{1, 2, 3, 4\}$ and calculate the n-gram vectors for both the source and target sentences.

For taggings, we use Named Entity Recognition (NER) and Parts of Speech (POS) and apply ideas similar to our n-gram design. As the tagging of a word is dependent of the sentence that the word lies in, a same word may be tagged differently in various sentences. To give an example in POS tagging, the word "Book" can either be tagged as "NOUN" or "VERB" depending on its meaning in the sentence. If most occurrences of the word "Book" are tagged as "NOUN" in the training set, the model may not be able to correctly learn the second meaning of it. Therefore, the model should be fed more with samples in which "Book" is tagged as "VERB". In order to reflect this phenomenon, we define a tagging value for a word w and its current tag t as

$$v(w,t) \triangleq \frac{1}{N_w} N_{tag(w)=t}, \tag{3.10}$$

where N_w is the number of times the word w has appeared in the training set and $N_{tag(w)=t}$ is the number of times the word w is tagged with tag t among all its occurrences. Similarly, for both NER and POS taggings, the tagging values of words form a vector representation of a sentence:

$$V(s_k) = \Big\{ v\Big(w_1, t_1\Big), ..., v\Big(w_L, t_L\Big) | w_l \in s_k \Big\}. \tag{3.11}$$

3.4.3 Action and Reward

In the task of curriculum learning, an action represents the process of data selection. [36] assume a stream-based setting where data examples come one by one in a stream, and design their action as making a decision on whether or not a single incoming data example should be selected. We argue that our problem setting is actually *pool-based* instead of stream-based, where a pool of data exists for selection and deciding on the selection of each individual sample per step would be inefficient. In [32], a dataset is split into several bins according to a noise measure of data samples, and the action determines from which bin the next batch of training data should be selected. However, this method highly depends on an effective heuristic criterion for bin formation and is thus hard to generalize.

Therefore, we propose our action design which samples a batch of b data examples from D_T, computes a score for each example according to the trained policy, and choose the one with the highest score. Correspondingly, in our design, we can easily control the size of D_S by varying the batch size b, i.e., $|D_S| = |D_T|/b$, since we choose one out of b samples for each batch.

For the choice of reward signal, we use the performance improvement of the NMT model evaluated on the validation set after it is updated with the selected sample. *Perplexity* is used as the performance metric instead of BLEU as it is shown to be more consistent and less noisy [44]. We assign a reward of 0 to unselected samples.

3.5 Experiments

In this section, we will first describe the datasets used in our evaluation and provide the implementation details along with the performance results.

3.5.1 Datasets & Metrics

To compare our proposed method with other curriculum learning methods on NMT task, we conduct comprehensive empirical analysis on several zh-en translation datasets:

- **MTD** is an internal news translation dataset with 1M samples in the training set and 1,892 samples in both the validation set and the test set. The average length of source sentences is 19.55 and the average length of target sentences is 21.06.

- **CASICTB, CASIA2015, NEU** are three independent datasets with 1M, 2M, and 2M examples from different data sources in WMT18 which is a public translation dataset in news area with more than 20M samples. We only use a part of data from WMT18 to evaluate our method. All three datasets share the same validation set *newsdev2017* and the test set *newstest2017* both composed of 2k samples.

Table 3.2: Description of evaluation datasets.

Dataset	Train	Val	Test	Src-Len	Tgt-Len
MTD	1,000,000	1,892	1,892	19.55	21.06
CASICTB	994,248	2,002	2,001	22.61	23.95
CASIA2015	1,049,988	2,002	2,001	23.89	25.19
NEU	1,999,946	2,002	2,001	16.95	18.54

Table 6.1 summarizes the details of the experimental datasets. The columns titled "Train", "Val", and "Test" correspond to the number of examples in training, validation, and test sets. "Src-Len" and "Tgt-Len" correspond to the average sentence length of source language and target language respectively.

For evaluation metrics, we use Perplexity for the reward calculation in RL agent training, and report BLEU [45] for the final performance of the NMT models.

3.5.2 Experimental Settings

We implement our models in PyTorch 1.1.0 [46] and train the model with a single Tesla P40. We utilize NLTK [47] to perform POS and NER taggings.

For the NMT model, we use the OpenNMT implementation [48] of Transformer [21]. It consists of a 6-layer encoder and decoder, with 8 attention heads, and 2,048 units for the feed-forward layers. The multi-head attention model dimension and the word embedding size are both set to 512. During training, we use Adam optimizer [49] with a learning rate of 2.0 decaying with a noam scheduler and a warm-up steps of 8,000. Each training batch contains 4,096 tokens and is selected with bucketing [27]. During inference, we employ beam search with a beam size of 5.

For the RL agent, we use an Deterministic Actor-Critic architecture and build our system based on [50]. In our framework, we use several tricks proven to be effective to RL training including a memory replay buffer of size 2,500, a warm-up phase of 500 steps, and a target network which is updated by mixing weights with the on-line

network with a mix factor of 0.1. For calculating rewards, we train the NMT model with the single selected sample using SGD and a learning rate of 1e-4.

The feature network maps data features of sentence length, sentence-level log-likelihood, taggings, and n-gram rarity to vectors of size 1, 8, 16, and 32 respectively with a FC layer, and concatenates them together as a shared state representation. The actor network is composed of two FC layers with hidden sizes of 300 and 400. The critic network is designed the same as the actor network except that the output action from actor network is concatenated to the second layer. Relu is used as the activation function in each FC layer in this network.

In experiments, we conduct two rounds of RL agent training and data selection. In each round, the RL agent is trained for 20k steps and the best model with the highest sum of rewards during the last 1,000 steps is used for data selection. The RL agent keeps selecting data given randomly sampled batches from the training set D_T and feed the selected data to the NMT model until no performance improvement is observed. For the training process on selected data, we keep the NMT model's optimizer and learning rate setting unchanged.

We use different batch sizes of the sampled batch b for the two rounds of training and selection with $b_1 = 16$ and $b_2 = 128$ indicating in the first round we select 1 sample from 16 and in the second round, from 128 samples we select the best one. This is because we think in order to achieve improvement further on the basis of the first round, a stricter data selection criterion must be applied.

3.5.3 Baselines

To make comparisons with other existing curriculum methods, we have conducted several baseline experiments.

We take the core ideas of existing curriculum learning methods of training on data samples with gradually increasing difficulty [22] and gradually decreasing noise [23] and apply them to our setting with pre-trained models. We evaluate the following

three baseline methods along with our proposed method.

- **Denoising** is a curriculum learning method of training an NMT model in a noise-annealing fashion [23]. They propose to measure NMT data noise with the help of a trusted dataset which contains generally cleaner data compared to the training dataset. [32] also utilize data noise in their curriculum design and achieve similar performance as [23]. For the choice of the trusted dataset, we choose a subset of 500 sentences from the validation set newsdev2017 of CASICTB, CASIA2015 and NEU.

- **Sentence Length** is an intuitive difficulty measure used in [22], since longer sentences tend to contain more information and more complicated sentence structure.

- **Word Rarity** is another metric for measuring the sample difficulty, as rare words appear less frequently in the training process and should be presented to the learning system more. The formula for calculating the word rarity of a sentence can be found in [22].

For baseline experiments, the pre-trained NMT model is further trained on 20% of the original data, which are selected by the above criteria, i.e., the least noisy, the longest, and the highest word rarity, respectively.

3.5.4 Main Results

Table 3.3 compares the performance of our method with other baseline methods on different datasets evaluated using BLEU. The result shows that our proposed method significantly out-performs other baseline methods by a great margin. We conduct two rounds of training and update in our experiments. While the result of the first round surpasses almost all the baseline methods, our second round further improves the performance and achieves a final BLEU improvement of +0.90, +0.60, +0.59, and

Table 3.3: Performance comparison of our proposed method with other baseline methods using BLEU on different datasets.

Method	MTD	CASICTB	CASIA2015	NEU
Base	18.21 (+0.00)	13.43 (+0.00)	18.65 (+0.00)	20.06 (+0.00)
Sentence Length	18.34 (+0.13)	13.52 (+0.09)	18.75 (+0.10)	20.21 (+0.15)
Word Rarity	18.35 (+0.14)	13.49 (+0.06)	18.72 (+0.07)	20.17 (+0.11)
Denoising	18.42 (+0.21)	13.55 (+0.12)	18.86 (+0.21)	20.44 (+0.38)
Ours-1 Round	18.81 (+0.60)	13.60 (+0.17)	18.91 (+0.26)	20.38 (+0.32)
Ours-2 Rounds	**19.11** (+**0.90**)	**14.03** (+**0.60**)	**19.24** (+**0.59**)	**20.79** (+**0.73**)

+0.71 on MTD, CASICTB, CASIA2015, and NEU respectively over the pre-trained model.

The reason of our success is due to our utilization of an RL framework to proactively select data samples that are potentially beneficial to the training of the NMT model. First, we formulate the task of curriculum learning on pre-trained NMT models as a reinforcement learning problem. Second, we construct an effective design of state, action and reward. Our state representation includes features of different dimensions of informativeness, uncertainty and diversity. Third, we propose a Deterministic Actor-Critic framework that learns a policy to select the best samples from the training set to improve the pre-trained model. By incorporating all these designs together, our proposed framework is able to achieve a significant performance enhancement on the pre-trained NMT model.

3.5.5 Analysis

We evaluate the impact of different modules and methods by ablation test on MTD dataset. Table 6.4 list the performance of our model variants with different features included.

We incrementally accommodate different features of examples to the state by first starting from sentence length and sentence-level log-likelihood as they are both

Table 3.4: Ablation analysis on different features of our proposed framework on MTD.

Method	MTD
Base	18.21 (+0.00)
Senlen + Logp	18.47 (+0.26)
+ N-gram	18.63 (+0.42)
+ Tagging	18.81 (+0.60)
+ 2nd Round	**19.11** (+**0.90**)

scalars. The performance increased slightly by 0.26 BLEU compared with the pre-trained base model. Then we further accommodate n-gram rarity and POS and NER taggings to the state vectors, and observe a larger increase of performance of 0.42 and 0.60 respectively. Finally, we incorporate the second round of RL agent training and data selection on the basis of the result of first round, and achieve the best performance with a 0.90 BLEU increase. Note that a stricter selection policy is applied to the second round (128 choose 1) compared with the first round (16 choose 1).

3.6 Conclusion

In this chapter, we study curriculum learning for NMT from a new perspective, to re-select a subset of useful samples from the existing dataset to further improve a pre-trained model, and formulate this task as a reinforcement learning problem. Compared with existing curriculum methods only applicable on train-from-scratch scenarios, our setting saves training time by better utilizing the existing pre-trained models. Our proposed framework is built based on the deterministic actor-critic algorithm, and learns a policy to select examples that can improve the model the most. We conduct experiments on several zh-en translation datasets and compare our method with other baseline methods including the easy-to-difficult curriculum and the denoising

scheme. Through rounds of training and data selection, our method achieves a significant performance boost on the pre-trained model, and out-performs all baselines methods by a great margin.

Chapter 4

AutoAugment

4.1 Introduction

Data augmentation has proven to be an effective regularization technique that can improve the generalization of deep neural networks by adding modified copies of existing samples to increase the volume and diversity of data used to train these networks. Traditional ways of applying data augmentation in computer vision include using single augmentation techniques, such as rotation, flipping and cutout [51], adopting randomly selected augmentations [52], and employing a manually crafted augmentation policy consisting of a combination of transformations. However, these methods either do not reach the full potential of data augmentation, or require human expertise in policy design for specific tasks.

Recently, automated learning of augmentation policies has become popular to surpass the limitation of manual design, achieving remarkable advances in both the performance and generalization ability on image classification tasks. Different search algorithms such as reinforcement learning [53], population-based training [54], and Bayesian Optimization [55] have been investigated to search effective augmentation policies from data to be used to train target networks. Dynamic augmentation strategies, e.g., PBA [54], AdvAA [56], are also proposed to learn non-stationary policies that vary during model training.

However, most existing methods focus on learning a single policy that is applied

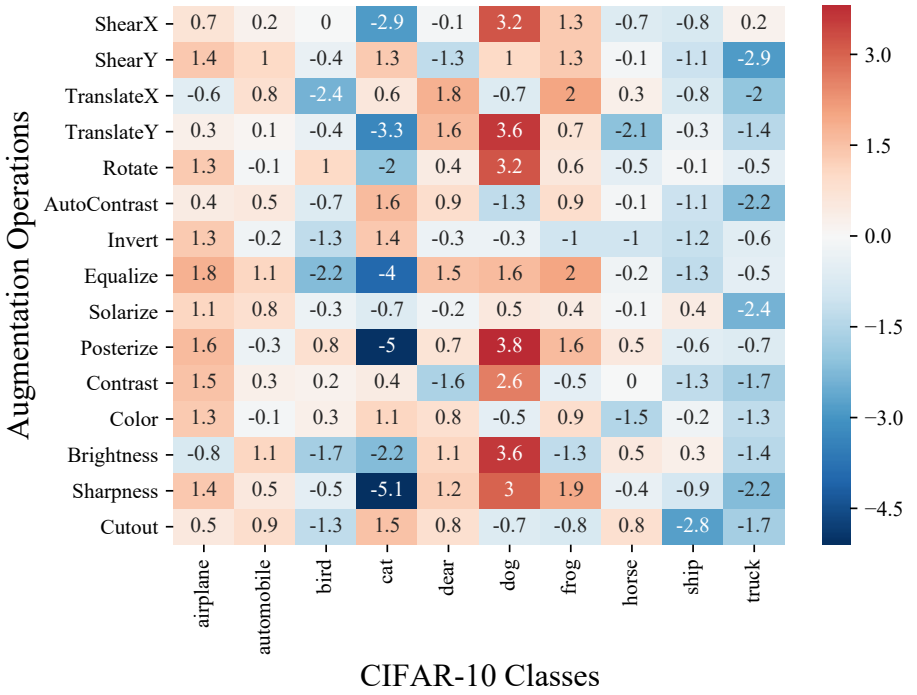


Figure 4.1: The effects of different augmentation operations on each class in CIFAR-10, demonstrated by the test accuracy change in each class after each single augmentation is applied to training WRN-40-2.

to all samples in the dataset equally, without considering variations between samples, classes or labels, which may lead to sub-optimal solutions. Figure 4.1 demonstrates the effects of different augmentation operations on different classes of samples in CIFAR-10, from which we can see that the effectiveness of augmentations is different on each class. For example, when the operation "Posterize" is applied in training, the test accuracy of "dog" class increases by 3.8%, whereas the test accuracy of "cat" drops significantly by 5%. It is possible that a certain augmentation used in training has completely different impacts on different labels. This observation implies the limitation of label or sample-invariant dataset-level augmentation policies. MetaAugment [57] proposes to learn a sample-aware augmentation policy by solving a sample re-weighting problem. It uses an augmentation policy network to take an augmentation operation and the corresponding augmented image as inputs, and outputs a weight to adjust the augmented image loss computed by the task network.

Despite the benefit of a fine-grained sample-dependent policy, MetaAugment is time-consuming and couples policy network learning with target model training, which may not be convenient in some production scenarios that require functional decomposition.

In this chapter, we propose an efficient data augmentation strategy named *Label-Aware AutoAugment (LA3)*, which produces label-aware augmentation policies to overcome the limitation of sample-invariant augmentation while still being computationally efficient as compared to sample-aware or dynamic augmentation strategies. *LA3* achieves competitive performance matching or outperforming a wide range of existing static and dynamic auto-augment methods, and attains the highest ImageNet accuracy on ResNet-50 among all existing augmentation methods including dynamic ones. In the meantime, *LA3* is also a simple scheme which separates augmentation policy search from target network model training, and produces stationary augmentation policies that can easily be applied to enhance deep learning with minimum perturbation to the original target model training routine.

LA3 adopts a two-staged design, which first explores a search space of combinations of operations and evaluates the effectiveness of promising augmentation operations for each class, while in the second stage, forms a composite policy to be used in target model training.

In the first stage of *LA3*, a neural predictor is designed to estimate the effectiveness of operation combinations on each class and is trained online through density matching as the exploration process iterates. We use Bayesian Optimization with a predictor-based sampling strategy to guide search into meaningful regions, which greatly improves the efficiency and reduces search cost.

In the second stage, rather than only selecting top augmentation operations, we introduce a policy construction method based on the minimum-redundancy maximum-reward (mRMR) principle [58] to enhance the performance of the composite augmentation policy when applied to the target model. This is in contrast to most prior methods [53, 55], which simply put together best performing augmentations in

evaluation, ignoring their complementary effects.

Extensive experiments show that using the same set of augmentation operations, the proposed *LA3* achieves excellent performance outperforming other low-cost static auto-augmentation strategies, including FastAA and DADA, on CIFAR-10 and CIFAR-100, in terms of the accuracy. On ImageNet, *LA3*, using stationary policies, achieves a new state-of-the-art top-1 accuracy of 79.97% on ResNet-50, which outperforms prior auto-augmentation methods including dynamic strategies such as AdvAA and MetaAug, while being 2× and 3× more computationally efficient, respectively.

4.2 Related Work

Data augmentation is a popular technique to alleviate overfitting and improve the generalization of neural network models by enlarging the volume and diversity of training data. Various data augmentation methods have been designed, such as Cutout [51], Mixup [59], CutMix [60], etc. Recently, automated augmentation policy search has become popular, replacing human-crafted policies by learning policies directly from data. AutoAugment [53] adopts a reinforcement learning framework that alternatively evaluates a child model and trains an RNN controller to sample child models to find effective augmentation policies. Although AutoAugment significantly improves the performance, its search process can take thousands of GPU hours which greatly limits its usability.

Multiple strategies are proposed to lower the search cost. Fast AutoAugment [55] proposes a density matching scheme to avoid training and evaluating child models, and uses Bayesian Optimization as the search algorithm. Weight-sharing AutoAugment [61] adopts weight-sharing settings and harvests rewards by fine-tuning child models on a shared pre-trained target network. Faster AutoAugment [62] further reduces the search time by making the search of policies end-to-end differentiable through gradient approximations and targeting to reduce the distance between the original and augmented image distributions. Similarly, DADA [63] relaxes the dis-

crete policy selection to a differentiable optimization problem via Gumbel-Softmax [64] and introduces an unbiased gradient estimator.

Instead of producing stationary augmentation policies that are consistent during the target network training, PBA [54] learns a non-stationary augmentation schedule, inspired by population based training [65], by modeling the augmentation policy search task as a process of hyperparameter schedule learning. AdvAA [56] adopts an adversarial framework that jointly optimizes target network training and augmentation search to find harder augmentation policies that produce the maximum training loss. However, AdvAA must rely on the batch augment trick, where each training batch is enlarged by multiple times with augmented copies, which significantly increases its computational cost. In general, one concern of these dynamic strategies is that they intervene the standard model training procedure, causing extra deployment overhead and may not be applicable in many production environments.

While most previous studies focus on learning augmentation policies for the entire dataset, MetaAugment [57] proposes to learn sample-aware augmentation policies during model training by formulating the policy search as a sample re-weighting problem, and constructing a policy network to learn the weights of specific augmented images by minimizing the validation loss via meta learning. Despite its benefits, MetaAugment is computationally expensive, requiring three forward and backward passes of the target network in each iteration. LB-Aug [66] is a concurrent work that also searches policies dependent on labels, but focuses on a different task under multi-label scenarios, where each sample has multiple labels rather than a single classification label. LB-Aug uses an actor-critic reinforcement learning framework and policy gradient approach for policy learning. Despite the benefits from label-based policies, LB-Aug has potential stability issues due to the use of reinforcement learning, which is generally harder and computational costly to train. In fact, the search cost of LB-Aug is not reported. In contrast, *LA3* targets the classical single-label image classification tasks, e.g., on CIFAR-10/100 and ImageNet benchmarks, on

which most other auto-augmentation methods are evaluated. It adopts Bayesian Optimization coupled with a neural predictor to sample and search for label-dependent augmentation policies efficiently. In addition, a policy construction stage is proposed to further form a more effective composite policy for target network training.

4.3 Methodology

In this section, we first review the task of conventional augmentation search and introduce the formulation of the proposed label-aware augmentation search task. Then we describe the two-stage design of *LA3*, and present the algorithm in detail.

4.3.1 Conventional Augmentation Search

Given an image recognition task with a training dataset $D^{tr} = \{(x_i, y_i\}_{i=1}^{|D^{tr}|}$, with x_i and y_i representing the image and label respectively, augmented samples $\mathcal{T}(x_i)$ are derived by applying augmentation policy $\mathcal{T}$ to sample x_i. Usually, the policy $\mathcal{T}$ is composed of multiple sub-policies τ, and each sub-policy is made up by K augmentation operations O, optionally with their corresponding probabilities and magnitudes, which are adopted in the original design of AutoAugment [53], but not included in some of the recent methods such as Weight-sharing AutoAugment [61] and MetaAugment [57].

Conventional augmentation search methods focus on the task whose goal is to construct the optimal policy $\mathcal{T}^*$ from given augmentations so that the performance $\mathcal{R}$ of the task network $\theta_{\mathcal{T}}$ on the validation dataset D^{val} is maximized:

$$\begin{aligned} \mathcal{T}^* &= \arg\max_{\mathcal{T}} \mathcal{R}(\theta_{\mathcal{T}} | D^{val}), \\ \text{where} \quad \theta_{\mathcal{T}} &= \arg\min_{\theta_{\mathcal{T}}} \frac{1}{|D^{tr}|} \sum_{i=1}^{|D^{tr}|} \mathcal{L}_{\theta}(\mathcal{T}(x_i), y_i), \end{aligned} \tag{4.1}$$

and $\mathcal{L}_{\theta}$ is the loss function of target network θ.

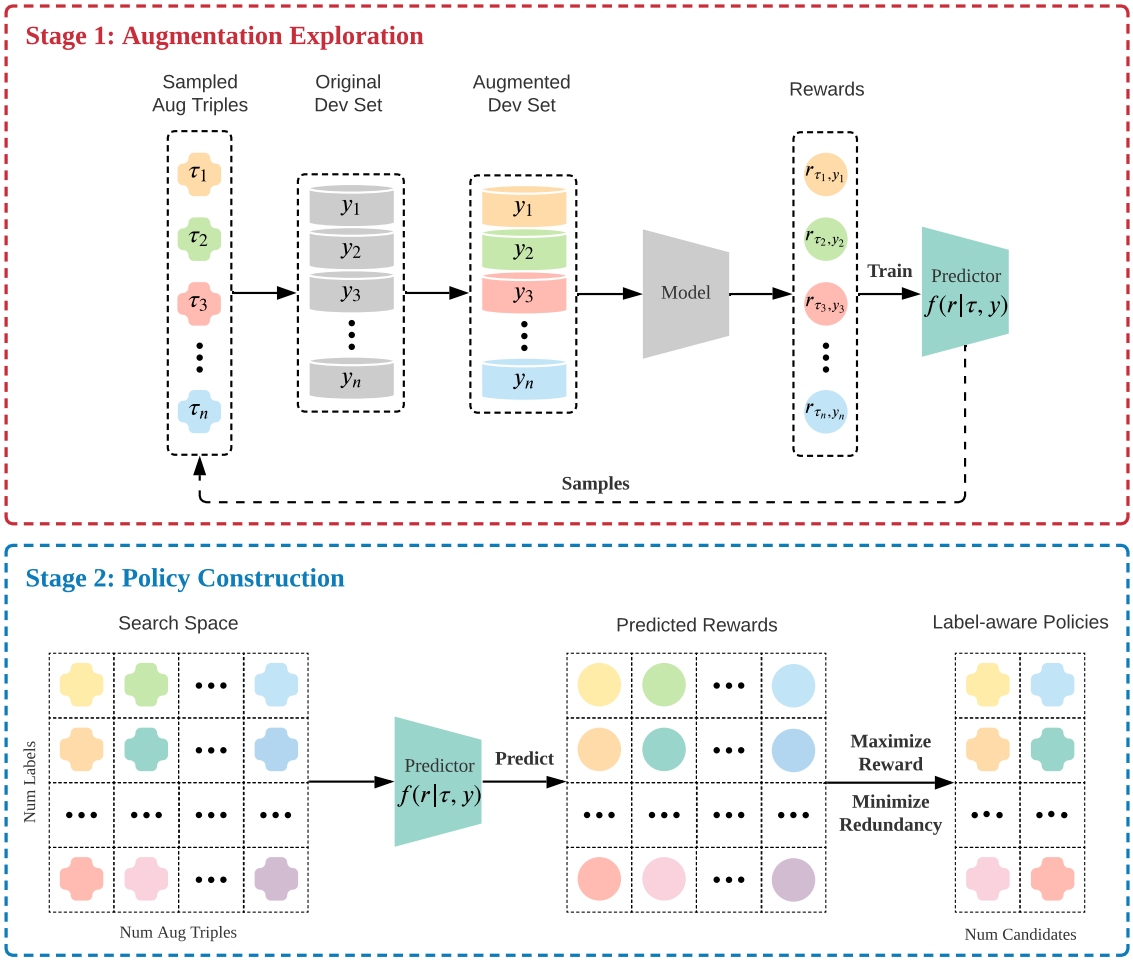


Figure 4.2: An overview of the proposed *LA3* method. It contains two stages, where in the first stage, augmentation triples are individually evaluated for each label via Bayesian Optimization with the help of an label-aware neural predictor. In the second stage, the best combination of complementary augmentation triples is selected based on the minimum-redundancy maximum-reward principle.

4.3.2 Label-Aware Augmentation Search

Though learning a dataset-level policy achieves considerable improvements, it is unlikely the optimal solution due to the lack of consideration of sample variations and utilization of label information.

In this chapter, we aim to learn a label-aware data augmentation policy $\mathcal{T}^* = \{\mathcal{T}^*_{y_0}, \cdots, \mathcal{T}^*_{y_n}\}$, where for samples of each label y_j, an individual policy $\mathcal{T}_{y_j}$ is learned

by maximizing the label-specific performance $\mathcal{R}_{y_j}$ of label y_j:

$$\begin{aligned} \mathcal{T}^*_{y_j} &= \arg\max_{\mathcal{T}_{y_j}} \mathcal{R}_{y_j}(\theta_{\mathcal{T}}|D^{val}), \\ \text{where} \quad \theta_{\mathcal{T}} &= \arg\min_{\theta_{\mathcal{T}}} \frac{1}{|D^{tr}|} \sum_{i=1}^{|D^{tr}|} \mathcal{L}_\theta(\mathcal{T}_{y_i}(x_i), y_i). \end{aligned} \tag{4.2}$$

Similar to conventional augmentation, in our label-aware setting, we define that each policy for a label is composed of multiple augmentation triples, each consisting of three augmentation operations. The magnitude of each augmentation operation is chosen randomly from ranges defined in AutoAugment [53], and is excluded from the search space in order to introduce randomness and diversity into the policy, and allocate more computational resources to assessing the fitness of operations to different classes of samples.

In this chapter, we propose a label-aware augmentation policy search algorithm called *LA3*, composed of two stages as presented in Figure 5.1. The first augmentation exploration stage aims to search for effective augmentation triples with density matching, and train a neural predictor to provide evaluations on all seen and unseen augmentation triples in the search space. And the goal of the second policy construction stage is to build a composite policy for each label based on the evaluation results from stage 1 by selecting a subset of complementary augmentation triples based on the minimum-redundancy maximum-reward principle.

4.3.3 Stage 1: Augmentation Exploration

Density Matching is an efficient mechanism originally proposed by Fast AutoAugment [55] to simplify the search process for effective augmentations, since the problem defined by Equation (4.1) and Equation (4.2) is a bi-level optimization problem, and is extremely hard to solve directly. It calculates the reward of each augmentation triple without the need of repeatedly training the target network. Specifically, given a model θ pre-trained on the training set D^{tr} and a validation set D^{val}, the performance of a certain augmentation triple τ can be evaluated by approximately measuring the

distance between the density of D^{tr} and density of augmented validation set $\tau(D^{val})$ with the model performance $\mathcal{R}(\theta|\tau(D^{val}))$. And the reward r is measured by the performance difference caused by applying the augmentation triple τ:

$$r_\tau = \mathcal{R}(\theta|\tau(D^{val})) - \mathcal{R}(\theta|D^{val}). \tag{4.3}$$

Similarly, in our label-aware setting, the reward r for a certain augmentation triple τ_y at label y is given by

$$r_{\tau,y} = \mathcal{R}_y(\theta|\tau_y(D^{val})) - \mathcal{R}_y(\theta|D^{val}). \tag{4.4}$$

Bayesian Optimization with a Neural Predictor is a widely adopted framework in many applications such as neural architecture search [67, 68] to find the optimal solution within a search space. In standard BO setting, over a sequence of iterations, the results from previous iterations are used to model a posterior distribution to guide the candidate selection of next iteration. And a neural predictor is a neural network that is repeatedly trained on the history evaluated candidates, and provides evaluations on unseen candidates, which increases the utilization efficiency of history evaluations and notably accelerates the search process.

In our *LA3* algorithm, we incorporate a label-aware neural predictor $f(r|\tau, y)$ which takes in an augmentation triple τ and the label y it is evaluated on, and predicts the reward r. In each iteration, the sampled augmentation triples for different labels are evaluated according to Equation (4.4), and together with the previous evaluated augmentation triples, are passed to train a new predictor.

Next, we select 100 candidate augmentation triples at the balance of exploration and exploitation, based on the following selection procedure: 1) Generate 10 new candidates by randomly mutating 1 or 2 operations in the chosen augmentation triples of the previous iteration; 2) Randomly sample 50 candidates from all unexplored augmentation triples; 3) Sample 40 candidates from the explored augmentation triples according to their real reward values. Then, for each label y, we choose the augmentation triple τ with the highest predicted reward $\tilde{r}_{\tau,y}$ for evaluation.

Algorithm 2: Stage 1: Augmentation Exploration

Input: Pre-trained target network θ, warm up iterations T_0, total iterations T

Output: Well-trained predictor $f^T(r|\tau, y)$

`/* warm-up phase */`

for $t = 0, \cdots, T_0$ **do**
  randomly generate augmentation triples $\{\tau^t_{y_0}, \cdots, \tau^t_{y_n}\}$ for all labels $\{y_0, \cdots, y_n\}$
  obtain rewards $\{r^t_{\tau,y_0}, \cdots, r^t_{\tau,y_n}\}$ by Equation (4.4)

`/* search phase */`

for $t = T_0, \cdots, T$ **do**
  train $f^t(r|\tau, y)$ with data collected from previous t iterations $\{(\tau, y, r_{\tau,y})\}^t$
  for $y_i = y_0, \cdots, y_n$ **do**
    generate 100 candidate augmentation triples by exploration and exploitation
    obtain predicted rewards $\tilde{r}_{\tau,y_i} = f^t(\tau, y_i)$ for 100 candidates
    $\tau^t_{y_i} = \arg\max_\tau(\tilde{r}_{\tau,y_i})$
  obtain real rewards $\{r^t_{\tau,y_0}, \cdots, r^t_{\tau,y_n}\}$ for $\{\tau^t_{y_0}, \cdots, \tau^t_{y_n}\}$ by Equation (4.4)

train predictor $f^T(r|\tau, y)$ with all collected data $\{(\tau, y, r_{\tau,y})\}^T$

Overall workflow of the first stage is summarized in Algorithm 2. To begin with, a warm-up phase of T_0 iterations is incorporated to randomly explore the search space, and retrieve the initial training data for learning a label-aware neural predictor $f(r|\tau, y)$. Then, for the following $T - T_0$ iterations, the search phase is adopted. In each iteration, we first train a neural predictor from scratch with data collected from previous iterations. Then, for each label, we apply the fore-mentioned selection procedure to select a set of candidate augmentation triples, and use the trained predictor to choose the augmentation triple for evaluation. After enough training data is collected, a well-trained label-aware neural predictor can be derived to provide accurate evaluations on all augmentation triples for different labels.

4.3.4 Stage 2: Policy Construction

Policy construction is a process of mapping the evaluation results of stage 1 to the final augmentation policy for training target networks. It is needed because augmentation

Algorithm 3: Stage 2: Policy Construction

Input: Well-trained predictor $f^T(r|\tau, y)$, search space A, number of candidates N_{cand}

Output: Label-aware policy $\mathcal{T}^*$

for $y_i = y_0, \cdots, y_n$ **do**
  for $\tau \in A$ **do**
    predict the reward $\tilde{r}_{\tau,y_i} = f^T(\tau, y_i)$
  initialize label-specific policy $\mathcal{T}_{y_i} \leftarrow \emptyset$
  for $k = 0, \cdots, N_{cand}$ **do**
    for $\tau \in (A \setminus \mathcal{T}_{y_i})$ **do**
      calculate $v(\tau, y_i)$ using Equation (4.5)
    find augmentation triple with highest score $\tau^k = \arg\max_\tau(v(\tau, y_i))$
    $\mathcal{T}_{y_i} \leftarrow \mathcal{T}_{y_i} \cup \tau^k$
$\mathcal{T}^* = \{\mathcal{T}_{y_0}, \cdots, \mathcal{T}_{y_n}\}$

policies are usually searched on light-weight proxy tasks such as density matching, but are evaluated on the complete tasks of image classification. Even for methods that search on complete tasks such as AutoAugment [53], they still naively concatenate multiple searched policies into a final policy. However, the policies for concatenation usually share a great potion of overlapped transformations, resulting in a high degree of redundancy.

In this chapter, we propose an effective policy construction method to iteratively select candidate augmentation triples for the final policy, based on the mutual information criteria of minimum-redundancy maximum-relevance (mRMR) [58]. Specifically, in *LA3*, the relevance metric is defined as the predicted reward $\tilde{r}$ as it provides a direct evaluation on the performance of a certain augmentation triple. And the redundancy of an augmentation triple τ is defined as the average number of intersecting operations between it and the already selected augmentation triples $\mathcal{T}_s$. Formally, in each iteration of policy construction, we define the score $v(\tau, y)$ of each unselected augmentation triple τ at label y as

$$v(\tau, y) = \tilde{r}_{\tau,y} - \alpha \times \overline{r} \times \frac{1}{|\mathcal{T}_s|} \sum_{\tau_s \in \mathcal{T}_s} |\tau \cap \tau_s|, \tag{4.5}$$

where $|\tau \cap \tau_s|$ refers to the number of overlapped operations between τ and τ_s, $\overline{r}$ is the average predicted reward of all augmentation triples in search space and is used to scale the redundancy, and α is a hyper-parameter adjusting the weight between the reward value and the redundancy value.

Algorithm 3 illustrates the overall process of the policy construction stage where the goal is to find a label-aware policy containing a collection of augmentation triples that maximizes the rewards while keeping a low degree of redundancy. Specifically, for each label y_i, we retrieve the predicted reward $\tilde{r}_{\tau,y_i}$ for each augmentation triple τ in the search space A. Afterwards, a label-specific policy $\mathcal{T}_{y_i}$ is constructed iteratively by calculating the score $v(\tau, y_i)$ of unselected augmentation triples with Equation (4.5) and add the augmentation triple with the highest score to the policy until the required number of candidates N_{cand} is met. Eventually, the label-aware policy $\mathcal{T}^*$ is built with each label y_i corresponding to a label-specific policy $\mathcal{T}_{y_i}$.

4.4 Experiments

In this section, we first describe the details of our experiment settings. Then we evaluate the proposed method, and compare it with previous methods in terms of both performance and search cost. Finally, we perform thorough analysis on the design of different modules in our algorithm. Code and searched policies are released at https://github.com/Simpleple/LA3-Label-Aware-AutoAugment.

4.4.1 Datasets, Metrics and Baselines

Following previous work, we evaluate our *LA3* method on CIFAR-10/100 [69] and ImageNet [70], across different networks including ResNet [71], WideResnet [72], Shake-Shake [73] and PyramidNet [74]. Test accuracy is reported to assess the effectiveness of the discovered policies, while the cost is assessed by the number of GPU hours measured on Nvidia V100 GPUs. For a fair comparison, we list results of stationary policies produced by static strategies, AutoAugment [53], FastAA [55], and DADA

[63]. We also include results from dynamic strategies, PBA [54], AdvAA [56], and MetaAug [57], producing non-stationary policies as target model training progresses.

4.4.2 Implementation Details

Policy Composition. For a fair comparison, we use the same 15 augmentation operations as PBA and DADA do, which is also the same set used by AA and FastAA with SamplePairing [75] excluded. Additionally, "Identity" operation that returns the original image is introduced in our search space to prevent images from being excessively transformed. Each label-specific policy consists of $N_{cand} = 100$ augmentation triples, while in evaluation, each sample is augmented by an augmentation triple randomly selected from the policy with random magnitudes.

Neural Predictor. The network structure of the neural predictor is composed of two embedding layers of size 100 that map labels and augmentation operations to latent vectors and three fully-connected layers of hidden size 100 with Relu activation function. The representation of an augmentation triple is constructed by combining the three augmentation operation embedding vectors with mean-pooling and concatenating it with the label embedding vector. Then it is passed into the FC layers to derive the predicted reward. The predictor network is trained for 100 epochs with Adam optimizer [76] and a learning rate of 0.01.

Search Details. For CIFAR-10/100, we split the original training set of $50,000$ samples into a training set D^{tr} of size $46,000$ to pre-train the model θ, and a valid set D^{val} of $4,000$ for density matching. We search our policy on WRN-40-2 network and apply the found policy to other networks for evaluation. For ImageNet, we randomly sample 50 examples per class from the original training set, and collect $50,000$ examples in total to form the valid set, where the remaining examples are used as the training set. In the augmentation exploration stage, the total number of iterations is set to $T = 500$, and the warm-up iterations is set to $T_0 = 100$. In the policy construction stage, $\alpha = 2.5$ is used to calculate the reward values of

Table 4.1: Top-1 test accuracy (%) on CIFAR-10 and CIFAR-100. We mainly compare our method *LA3* with methods that also produce stationary augmentation policies, including AA, FastAA and DADA. Results of dynamic policies (PBA, AdvAA and MetaAug) are also provided for reference.

Dataset	**Model**	**Baseline**	**AA**	**FastAA**	**DADA**	**LA3**	**PBA**	**AdvAA**	**MetaAug**
			static	static	static	static	dynamic	dynamic	dynamic
CIFAR-10	WRN-40-2	94.7	96.3	96.4	96.4	**97.08 ± 0.08**	–	–	96.79
	WRN-28-10	96.1	97.4	97.3	97.3	**97.80 ± 0.15**	97.42	98.10	97.76
	Shake-Shake (26 2x96d)	97.1	98.0	98.0	98.0	**98.07 ± 0.11**	97.97	98.15	98.29
	Shake-Shake (26 2x112d)	97.2	98.1	98.1	98.0	**98.12 ± 0.08**	97.97	98.22	98.28
	PyramidNet+ShakeDrop	97.3	98.5	98.3	98.3	**98.55 ± 0.02**	98.54	98.64	98.57
CIFAR-100	WRN-40-2	74.0	79.3	79.4	79.1	**81.09 ± 0.28**	–	–	80.60
	WRN-28-10	81.2	82.9	82.8	82.5	**84.54 ± 0.03**	83.27	84.51	83.79
	Shake-Shake (26 2x96d)	82.9	**85.7**	85.4	84.7	85.17 ± 0.13	84.69	85.90	85.97
	PyramidNet+ShakeDrop	86.0	**89.3**	88.3	88.8	89.02 ± 0.03	89.06	89.58	89.46

augmentation triples.

Evaluation. The evaluation is performed by training target networks with the searched policies, and the results are reported as the mean test accuracy and standard deviation over three runs with different random seeds. We do not specifically tune the training hyperparameters and use settings consistent with prior work. We include the details in the supplementary materials.

4.4.3 Experimental Results

CIFAR-10/100. Table 4.1 summarizes the CIFAR-10 and CIFAR-100 results of different auto-augmentation methods on a wide range of networks. Among all static methods that produce stationary policies, *LA3* achieves the best performance for all 5 target networks on CIFAR-10 and for 2 out of 4 target networks on CIFAR-100. When extending the comparison to also include dynamic strategies, *LA3* still achieves the best CIFAR-10 and CIFAR-100 accuracies on WRN-40-2, which is the original network on which policy search was performed. When transferring these augmentation policies found on WRN-40-2 to other target network models for evaluation, *LA3* also achieves excellent performance comparable to the current best methods. In par-

Table 4.2: ResNet-50 top-1 test accuracy (%) and computational cost on ImageNet. Batch Augment (BA) trick is used in the training of *LA3* (BA), AdvAA (BA) and MetaAug (BA). The number of transformations used in batch augment is also given in the table.

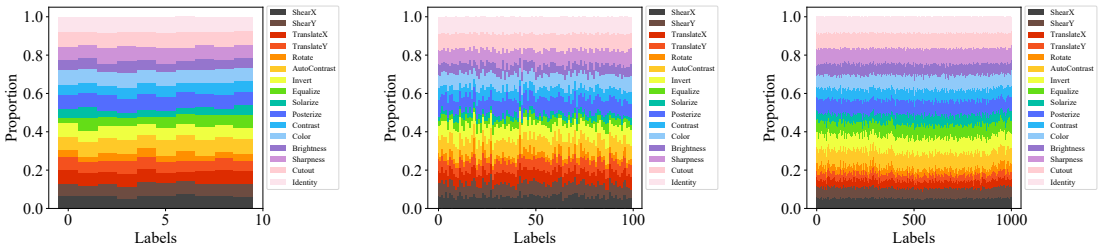
	Baseline	**AA** static	**FastAA** static	**DADA** static	**LA3** static	**LA3 (BA)** static	**AdvAA (BA)** dynamic	**MetaAug (BA)** dynamic
Batch Augment (BA)	n/a	n/a	n/a	n/a	n/a	×4	×8	×4
ResNet-50 Acc (%)	76.3	77.6	77.6	77.5	**78.71 ± 0.07**	**79.97 ± 0.07**	79.40	79.74
Search Cost (h)	–	15,000	450	1.3	29.3	29.3	–	–
Train Cost (h)	160	160	160	160	160	640	1,280	1,920
Total Cost (h)	160	15,160	610	161.3	189.3	669.3	1,280	1,920

ticular, *LA3* achieves the highest score for WRN-28-10 on CIFAR-100. These results evidently proves the effectiveness of *LA3* as an augmentation strategy to improve model performance, and demonstrates the strong transferability of our label-aware policies across different neural networks.

ImageNet Performance. In Table 4.2, we list the top-1 accuracy of different methods evaluated on ResNet-50, as well as their computational cost. For a fair comparison, we also indicate whether the Batch Augment (BA) trick [56], which forms a large batch with multiple copies of transformed samples, is used for each method, with "(BA)" after the method name. We also indicate the number of transformations used in the batch augment. Note that the search cost for dynamic methods is included in the training cost, since they learn a dynamic augmentation policy during the training of the target model. We include the results for *LA3* both with and without batch augment.

From Table 4.2 we can observe that among all methods without the batch augment trick, *LA3* achieves the best ImageNet top-1 accuracy of 78.71%, while the search only took 29.3 GPU hours, which is 15 times faster than FastAA. Although DADA is faster, *LA3* is substantially better in terms of the ImageNet accuracy achieved.

Meanwhile, *LA3 (BA)* achieves a new state-of-the-art ImageNet accuracy of 79.97% surpassing all existing auto-augmentation strategies including dynamic strategies Ad-



(a) CIFAR-10 policy (b) CIFAR-100 policy (c) ImageNet policy

Figure 4.3: The proportion of different augmentation operations in policies for different labels in *LA3* searched label-aware policies on CIFAR-10, CIFAR-100 and ImageNet.

vAA and MetaAug, with a total computational cost 2 times and 3 times lower than theirs, respectively. The high cost of these dynamic policies is due to the fact that augmentation policies may vary for each sample or batch and must be learnt together with model training. By generating static policies, *LA3* is a simpler solution that decouples policy search from model training and evaluation, which is easier to deploy in a production environment, without introducing specialized structures, e.g., the policy networks in AdvAA and MetaAug, into target model training.

4.4.4 Ablation Study and Analysis

The reason of the success can be attributed to the following designs in our *LA3* algorithm.

Label-Awareness. One of the main contributions of the paper is to leverage the label information and separately learn policies for samples of different classes, which captures distinct characteristics of data and produces more effective label-aware policies. The results of *LA3* variant without label-awareness (i.e., searching for label-invariant policies) are shown in the first row of Table 4.3, which are constantly lower than *LA3* in all experimental settings. This confirms that label-aware augmentation policies are effective at improving target network accuracy.

Figure 4.3 gives an overview of the searched label-aware policies on CIFAR-10, CIFAR-100 and ImageNet, where we calculate the occurrences of different operations

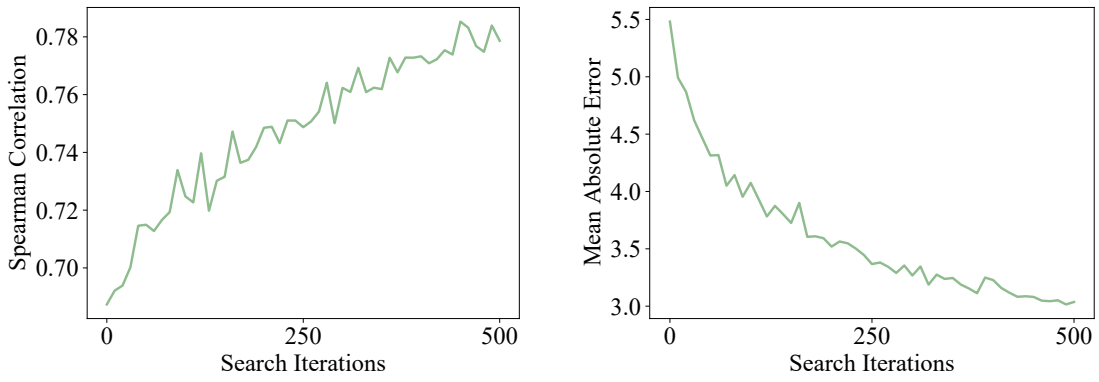
Table 4.3: Ablation analysis results in top-1 test accuracy (%) on CIFAR-10 and CIFAR-100 with different designs removed from the full *LA3* method.

	CIFAR-10		CIFAR-100	
	WRN-40-2	WRN-28-10	WRN-40-2	WRN-28-10
w/o Label-aware	96.70	97.11	80.08	82.76
w/o Stage 2 (top-100)	96.53	97.49	78.57	82.76
w/o Stage 2 (top-500)	96.70	97.26	79.85	84.04
LA3	**97.08**	**97.80**	**81.09**	**84.54**

in each label-specific policy and plot their proportions in different colors. We can see that the derived policies possess a high diversity by having all the operations contributing to the final policy, meanwhile making the individual policies notably different among labels. This observation further proves the need for separately treating samples of different labels in augmentation policy search.

Neural Predictor. In addition to using density matching to simplify augmentation assessment during search, we have adopted a label-aware neural predictor to learn the mapping from an augmentation triple to its label-specific reward. We now conduct a thorough evaluation to assess the performance of the neural predictor. For each search iteration, the predictor is trained on 80% of the history data and tested on the remaining 20% data in terms of both the Spearman's Rank Correlation and Mean Absolute Error (MAE). As shown in Figure 4.4, as the policy search on ImageNet progresses and more samples are explored, the predictor can produce more accurate predictions of rewards, obtaining a 0.78 Spearman Correlation and a decreased MAE when the search ends. This allows the predictor to properly guide the search process and find effective policies.

Furthermore, the use of the predictor better utilizes the search history and improves the sample efficiency during searching. As a result, the search cost of our method is significantly reduced and is 15 times lower than FastAA.



(a) Spearman Correlation (b) Mean Absolute Error

Figure 4.4: The evaluation of the predictor during the policy search on ImageNet given by the Spearman's Rank Correlation and Mean Absolute Error over search iterations.

Policy Construction. We evaluate the impact of our two-stage design on CIFAR-10 and CIFAR-100 datasets, by showing the performance of model variants with different policy construction methods in row 2 and 3 of Table 4.3.

We compare our policy construction method based on mRMR to the commonly used Top-k selection method adopted in AA [53], FastAA [55] and DADA [63]. We use two different k value settings of $k = 100$ equaling the number of candidates used in *LA3*, and $k = 500$ following the FastAA setting. We can see that the policy that includes 500 augmentation triples per label with top predicted rewards yields a better performance than the policy with top 100 augmentation triples on both CIFAR-10 and CIFAR-100. This can be attributed to the better diversity as more possibilities of augmentations are contained. However, increasing the k value is not the best solution to improve augmentation diversity as the augmentation triples with high rewards tend to have similar compositions and may result in a high redundancy in the final policy. Our *LA3* incorporates a policy construction method that selects high-reward augmentation triples, and at the same time, keeping the lowest redundancy of the final policy. With the two-stage design, our *LA3* method beats the top-k variants and produces significant improvements in all settings.

Limitation. Unlike dataset-level augmentation policies that can be learned from one dataset and transferred to other datasets [53, 54, 56], *LA3* learns label-aware policies where labels are specific to a dataset, and hence lacks the transferability across datasets, although *LA3* demonstrates transferability across networks as shown in Table 4.1. However, when dealing with a large dataset, *LA3* can work on a reduced version of the dataset to search for label-dependent policies efficiently, and requires no tuning on training recipes when applying the found policy to the entire dataset.

4.5 Conclusion

In this chapter, we propose a label-aware data augmentation search algorithm where label-specific policies are learned based on a two-stage algorithm, including an augmentation exploration stage based on Bayesian Optimization and neural predictors as well as a composite policy construction stage. Compared with existing static and dynamic augmentation algorithms, *LA3* is computationally efficient and produces stationary policies that can be easily deployed to improve deep learning performance. *LA3* achieves the state-of-the-art ImageNet accuracy of 79.97% on ResNet-50 among all auto-augmentation methods, at a substantially lower search cost than AdvAA and MetaAugment.

Part II

Automated Input Dimension Pruning for Efficient Model Inference and Deployment

Chapter 5

Video Frame Selection

5.1 Introduction

Videos have proliferated online in recent years with the popularity of social media, and have become a major form of content consumption on the Internet. The abundant video data has greatly encouraged the development of deep learning techniques for video content understanding. As one of the most important tasks, action recognition aims to identify relevant actions described in videos, and plays a vital role to other downstream tasks like video retrieval and recommendation.

Due to the high computational cost of processing frames in a video, common practices of action recognition involve sampling a subset of frames or clips uniformly [77] or densely [78, 79] from a given video a serve as the input to a content understanding model. However, since frames in a video may contain redundant information and are not equally important, simple sampling methods are often incapable of capturing such knowledge and hence can lead to sub-optimal action recognition results.

Prior studies attempt to actively select relevant video frames to overcome the limitation of straightforward sampling, achieving improvements to model performance. Heuristic methods are proposed to rank and select frames according to the importance score of each frame/clip calculated by per-frame prediction [80, 81]. Despite the effectiveness, these methods heavily rely on per-frame features, without considering the interaction or diversity among selected frames. Reinforcement learning (RL)

has also been proposed to identify informative frames by formulating frame selection as a Markov decision process (MDP) [82–86]. However, existing RL-based methods may suffer from training stability issues and rely on a massive amount of training samples. Moreover, RL methods make an MDP assumption that frames are selected sequentially depending on observations of already selected frames, and thus cannot adjust prior selections based on new observations.

In this work, we propose a new learning paradigm named Search-Map-Search (SMS), which directly searches for the best combination of frames from a video as one entity. SMS formulates the problem of frame selection from the perspective of heuristic search in a large space of video frame combinations, which is further coupled with a learnable mapping function to generalize to new videos and achieve efficient inference.

Specifically, we propose a hierarchical search algorithm to efficiently find the most favorable frame combinations on training videos, which are then used as explicit supervision information to train a feature mapping function that maps the feature vectors of an input video to the feature vector of the desirable optimal frame combination. During inference on an unseen query video, the learned mapping function projects the query video onto a target feature vector for the desired frame combination, where another search process retrieves the actual frame combination that approximates the target feature vector. By combining search with learning, the proposed SMS method can better capture frame interactions while incurring a low inference cost.

The effectiveness of SMS is extensively evaluated on both the long untrimmed action recognition benchmarks, i.e., ActivityNet [87] and FCVID [88], and the short trimmed UCF101 task [89]. Experimental results show that SMS can significantly improve action recognition models and precisely recognize and produce effective frame selections. Furthermore, SMS significantly outperforms a range of other existing frame selection methods for the same number of frames selected, while can still generate performance higher than existing methods using only 10% of all labeled video

samples for training.

5.2 Related Work

Action Recognition. 2D ConvNets have been widely utilized for action recognition, where per-frame features are first extracted and later aggregated with different methods such as temporal averaging [77], recurrent networks [79, 90, 91], and temporal channel shift [92–94]. Some studies leveraged both the short-term and long-term temporal relationships by two-stream architectures [95, 96]. To jointly capture the spatio-temporal information of videos, 3D ConvNets were proposed, including C3D [97], I3D [98] and X3D [99]. Transformer architecture [21] have also been applied to video understanding by modeling the spatio-temporal information with attention [100, 101].

In this chapter, we follow the previous frame selection work and apply our method mainly on the temporal averaging 2D ConvNets.

Frame Selection. The problem of selecting important frames within a video has been investigated in order to improve the performance and reduce the computational cost.

Many researchers focused on selecting frames based on the per-frame heuristic score. SCSampler [80] proposed to select frames based on the predicted scores of a lightweight video model as the usefulness of frames. SMART [81] incorporated an attention module that takes randomly selected frame pairs as input to model the relationship between frames. However, these methods select frames individually regardless of interactions between selected frames, which may lead to redundant selections.

Reinforcement learning (RL) approaches are widely adopted in frame selection to find the effective frames in a trail-and-error setting. FastForward [85] and AdaFrame [82] adopted a single RL agent to generate a decision on the next frame, and updated the network with policy gradient. MARL [83] formulated the frame sampling process

as multiple parallel Markov Decision Processes, and adopted multiple RL agents each responsible for determining a frame.Although the RL-based approaches are effective, the training stability issue and the requirement of huge amount of training samples with high computational overhead remain a problem.

Recent studies combined frame selection with other techniques to improve the model efficiency. LiteEval [102] adopted a two-level feature extraction procedure, where fine expensive features were extracted for important frames, and coarse frames were used for the remaining frames. ListenToLook [103] proposed to use audio information as an efficient video preview for frame selection. AR-Net [104] aimed at selecting the optimal resolution for each frame that is needed to correctly recognize the actions, and learns a differentiable policy using Gumbel Softmax trick [64].

Our work focuses on the classic task of selecting a subset of frames based on visual information. Different from existing methods, our work incorporates a new "Search-Map-Search" paradigm that leverages efficient search and supervised feature mapping to explicitly find the best frame combinations, and achieves excellent performance outperforming other frame selection methods.

5.3 Methodology

5.3.1 Overall Architecture

Figure 5.1 gives an overview of our proposed framework, which consists of three stages: a search stage, a feature mapping stage, and another search stage. The training process of our method involves the first two stages, while the inference process involves the last two.

Specifically, the goal of the first search stage is to find the best frame combinations in training videos with the lowest model losses, which serve as the supervisory target information for the feature mapping stage. We design an efficient hierarchical algorithm coupled with Guided Local Search [105] to identify the effective frame

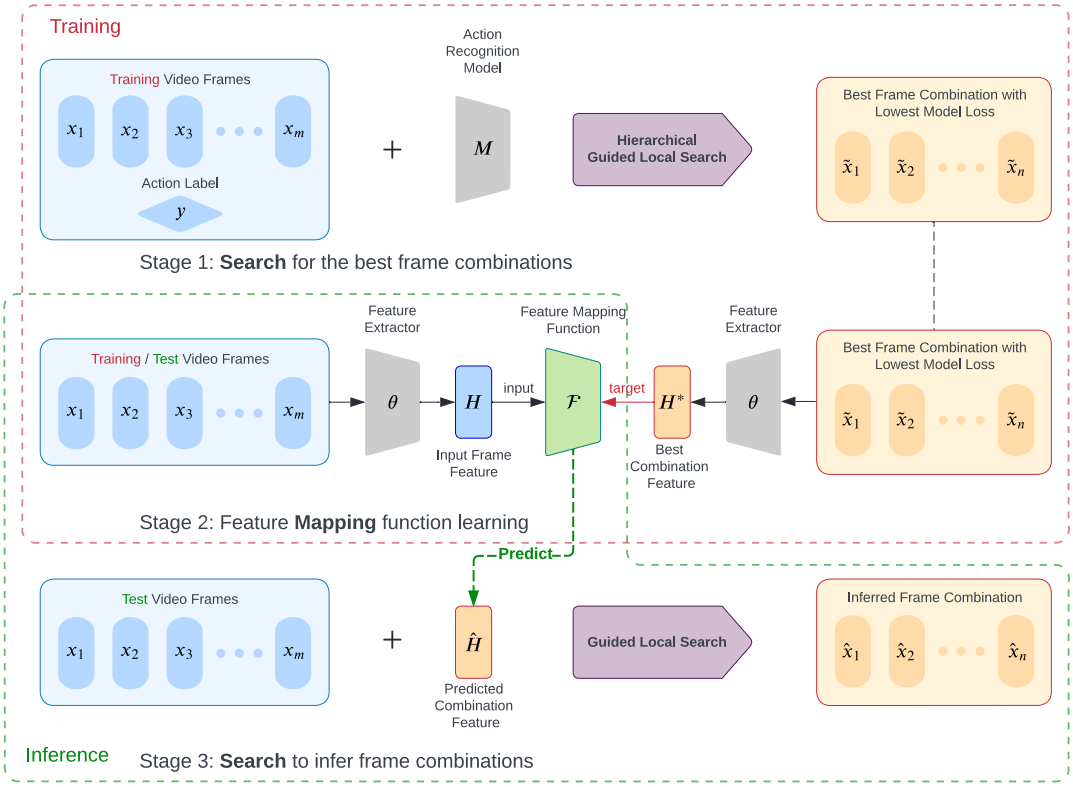


Figure 5.1: An overview of the proposed "Search-Map-Search" method. It contains three stages, where in the first stage, an efficient hierarchical search algorithm is used to derive the best frame combinations with lowest losses, which are utilized as the supervised information to train a feature mapping function in the second stage. In the third stage, for a query video, we incorporate another search process to infer the frame combination whose features are closest to the combination feature predicted with the trained feature mapping function.

combinations at a low search cost.

In the second stage, a feature extractor is employed to extract input frame features from the training video frames, and the feature of the best combination from search results. Then, a feature mapping function is trained via supervised learning by taking the input frame features as input, and transforming it to the target feature of the best combination.

In the third stage, we incorporate another search process to infer the effective frame combination whose feature is closest to the predicted feature from the well-trained

feature mapping function.

5.3.2 Stage 1: Search for Best Frame Combinations

Given an action recognition task with a pre-trained model M and a training dataset $D^{tr} = \{X, y\}^{|D^{tr}|}$, where $X = \{x_i\}_{i=1}^{m}$ represents a video sample made up of a collection of m frames, and y is the action label for the video, our goal of stage 1 is to find the best frame combination $\tilde{X}^*$ with n frames for each training video X that minimizes the model loss:

$$\begin{aligned} \tilde{X}^* &= \arg\min_{\tilde{X}} \mathcal{L}\big(M(\tilde{X}), y\big), \\ \textit{where} \quad \tilde{X} &= \{x_k | x_k \in X\}^n, \end{aligned} \tag{5.1}$$

where $\mathcal{L}$ is the loss function of the action recognition task. Note that repetitive selection of the same frame is allowed in our setting, as we believe that repeated important frames are better than meaningless frames.

In order to efficiently find the best frame combinations, we have designed a hierarchical search algorithm which exploits the high similarities of adjacent frames by performing search hierarchically on coarse-grained clips first, and then on the fine-grained frames. Besides, we incorporate Guided Local Search [105] in our algorithm to exploit per-frame losses as prior information for a good starting search point, which further reduces search costs and empirically outperforms other strong search algorithms such as Genetic Algorithm [106].

The overall workflow of our hierarchical search is summarized in Algorithm 4. To begin with, the video is first split into coarse-grained clips each consisting of a collection of non-overlapped frames. Then, we calculate the model loss for each clip by representing it with the averaged feature vector of all frame inside it, and utilize the information to create an initial solution composed of the clip with the lowest model loss repeated n times. On top of the initial searching point, we adapt Guided Local Search to find the best clip combination by defining a problem-specific penalty to escape from local optimum points. The details of Guided Local Search

Algorithm 4: Hierarchical Search

Input: Video X, Clip Length K, Combination Length n
Output: Frame Combination $\tilde{X}^*$

/* clip search phase */
1 Split the video into clips each consisting of K frames
2 Prepare an initial solution $\tilde{C}_0$ containing the clip with the lowest loss repeated n times
3 Perform Guided Local Search to improve $\tilde{C}_0$ and get $\tilde{C}^* = \{C_k\}_{k=1}^n$

/* frame search phase */
4 Define search space for each position in combination $S^F = \{S_k | S_k = \{x_j | x_j \in C_k\}\}_{k=1}^n$ on top of $\tilde{C}^*$
5 Randomly initialize solution $\tilde{X}_0$ from S^F
6 Perform Guided Local Search to improve $\tilde{X}_0$ and get $\tilde{X}^* = \{x_k\}_{k=1}^n$

will be introduced in Appendix. After searching on coarse-grained clips, we again incorporate Guided Local Search to find the best fine-grained frame combinations by replacing each derived clip with a frame inside it. Via hierarchical search, we have greatly reduced the search space and lowered the search cost significantly, while obtaining satisfying solutions.

5.3.3 Stage 2: Feature Mapping Function

The goal of the second stage is to identify the best frame combination produced in stage 1, given the input video frames by learning a feature mapping function $\mathcal{F}$.

Specifically, the feature mapping function $\mathcal{F}$ takes in the features of input frames H_0 generated by a pre-trained feature extractor θ, and outputs a predicted feature $\hat{h} \in \mathcal{R}^d$ representing a frame combination where d denotes the feature dimension:

$$\begin{aligned} \hat{h} &= \mathcal{F}(H_0) \\ \textit{where} \quad H_0 &= \{h_i | h_i = \theta(x_i)\}_{i=1}^m, \end{aligned} \tag{5.2}$$

where θ is the feature extractor, and $h_i \in \mathcal{R}^d$ is the extracted feature vector for frame x_i.

For the network structure of mapping function $\mathcal{F}$, we choose to incorporate transformer layers [21] to construct the spatio-temporal representations of the input frame

features, and an aggregation function to aggregate the representations of variable lengths into a predicted feature vector $\hat{h}$:

$$\begin{aligned} H_l &= \mathit{transformer}(H_{l-1}) \\ \hat{h} &= \mathit{aggr}(H_l), \end{aligned} \tag{5.3}$$

where l is the number of transformer layers in the mapping function.

The objective of the mapping function is to minimize the distance between the predicted feature $\hat{h}$ and the aggregated feature vector of the searched frame combination h^*:

$$\begin{aligned} \min \quad & \mathit{dist}(\hat{h}, h^*) \\ \mathit{where} \quad & h^* = \mathit{aggr}(\{h_k | h_k = \theta(x_k), x_k \in \tilde{X}^*\}). \end{aligned} \tag{5.4}$$

In our implementation, we incorporate cosine distance and mean-pooling as the distance function and aggregation function respectively, while other function choices can be further explored.

5.3.4 Stage 3: Search to Infer Frame Combinations

After the mapping function is learned, it can accurately predict the features of the best frame combinations for unseen videos without relying on the ground truth labels. The goal of this stage is to incorporate another search process to infer the frame combinations from the predicted features. Formally, the objective of the search is to find a frame combination $\hat{X}$ whose aggregated feature h' is closest to the given predicted feature $\hat{h}$:

$$\begin{aligned} \hat{X} &= \arg\min_{\tilde{X}} (\mathit{dist}(h', \hat{h})) \\ \mathit{where} \quad & h' = \mathit{aggr}(\{h_k | h_k = \theta(x_k), x_k \in \tilde{X}\}). \end{aligned} \tag{5.5}$$

As the evaluation in the search only involves the calculation of the cosine distance between vectors, which requires little computation, we directly apply Guided Local Search on the fine-grained frame level without the hierarchical setting applied in stage 1.

5.4 Experiments

In this section, we conduct extensive experiments aiming at investigating the following research questions:

- **RQ1:** Can the proposed SMS improve model performance over the base frame sampling method?
- **RQ2:** How does SMS perform compared to other state-of-the-art frame selection methods?
- **RQ3:** What's the computation efficiency of SMS for video inference?
- **RQ4:** How do the different components affect the performance of the proposed SMS?
- **RQ5:** Can SMS generalize well to spatio-temporal models, e.g., transformer based video models?

5.4.1 Experimental Setup

Datasets.

We evaluate our SMS method on 3 action recognition benchmarks including ActivityNet V1.3 [87], FCVID [88] and UCF101 [89]. The videos in ActivityNet and FCVID are untrimmed with average video lengths of 117 and 167 seconds respectively, while UCF101 dataset contains trimmed short videos with an average length of 7.21 seconds. Table 5.1 summarizes the detailed information of the experimental datasets.

Baselines.

We compare the proposed SMS with the base selection method and the following state-of-the-art frame selection methods:

- **Base** is a sparse sampling method proposed in TSN [77], where videos are divided into segments of equal length, and frames are randomly sampled within each segments.

- **AdaFrame** [82] incorporates reinforcement learning to adaptively select informative frames with a memory-augmented LSTM. At testing time, AdaFrame selects different number of frames for each video observed.

- **MARL** [83] adopts multiple RL agents each responsible for adjusting the position of a selected frame.

- **SCSampler** [80] proposes to select frames based on the prediction scores produced by a lightweight video model.

- **SMART** [81] combines the single-frame predictive score with pair-wise interaction score to make decision on the frame selections.

- **LiteEval** [102] selects important frames to extract fine features and adopts coarse features for the remaining frames.

- **ListenToLook** [103] proposes to use audio information as video preview for frame selection. For a fair comparison, we follow AR-Net [104] and include the variant with only the visual modality.

- **AR-Net** [104] aims to select the optimal resolutions for frames that are needed to correctly recognize the actions.

Evaluation metrics.

Following previous studies, we evaluate the performance of models using mean Average Precision (mAR), which is a commonly adopted metric in action recognition tasks, calculated as the mean value of the average precision over all action classes.

Dataset	Train	Val	Actions	Avg. Duration
ActivityNet	10,024	4,926	200	117s
FCVID	45,611	45,612	239	167s
UCF101	9,537	3,783	101	7.21s

Table 5.1: Description of evaluation datasets.

	ActivityNet			FCVID			UCF101	
# Frames	8	16	25	8	16	25	3	8
Base[1]	77.34 ± 0.06	79.41 ± 0.05	80.04 ± 0.24	83.84 ± 0.05	85.34 ± 0.03	85.65 ± 0.04	90.65 ± 0.13	90.70 ± 0.10
SMS (infer only)	82.76 ± 0.15	83.78 ± 0.08	83.85 ± 0.17	86.35 ± 0.03	86.94 ± 0.06	87.08 ± 0.04	91.45 ± 0.12	91.58 ± 0.09
SMS (train & infer)	$\mathbf{83.72 \pm 0.05}$	$\mathbf{84.35 \pm 0.08}$	$\mathbf{84.56 \pm 0.08}$	$\mathbf{86.54 \pm 0.06}$	$\mathbf{87.25 \pm 0.11}$	$\mathbf{87.59 \pm 0.01}$	$\mathbf{91.94 \pm 0.15}$	$\mathbf{92.25 \pm 0.10}$

Table 5.2: Performance comparison of the proposed SMS and the base method. In SMS (infer only), frames are selected with our method only during inference. In SMS (train & infer), frames are selected with SMS during both training and inference.

Implementation details.

In the first stage of hierarchical searching, the clip length K is set to 30. For the feature mapping network, we adopt a two-layer transformer network with a hidden dimension of $2,048$. The feature extraction network used in our implementation is a ResNet-50 network [71] pre-trained on the Kinetics dataset [98].

For the data pre-processing for action recognition tasks, we decode the video at 1 fps for long videos in ActivityNet and FCVID and 55 fps for short videos in UCF101 to retrieve the rgb frames, which are augmented during training by resizing the short side to 256, random cropped and resized to 224^2, after which a random flip with a probability of 0.5 is applied. For inference, we resize all frames to 256^2 and perform three-crop.

For the training of action recognition models, we choose ResNet-50 as the backbone and run 100 epochs using an SGD optimizer with a momentum of 0.9, and a step learning rate schedule which decays the learning rate by a factor of 10 every 40

Method	Backbone	ActivityNet			FCVID		
		# Frames	mAP	impr.	# Frames	mAP	impr.
SCSampler	ResNet-50	10	72.9	0.4	10	81.0	0.0
AdaFrame	ResNet-101	8.65	71.5	3.7	8.21	80.2	1.8
MARL	ResNet-101	8	72.9	0.4	-	-	-
SMART	ResNet-101	10	73.1	-	10	82.1	-
SMART*	ResNet-50	8	80.67	3.33	8	83.35	−0.49
SMS_10%	ResNet-50	8	82.12	4.78	8	85.69	1.85
SMS	ResNet-50	8	**83.72**	**6.38**	8	**86.54**	**2.70**

Table 5.3: Performance comparison of the proposed SMS and other state-of-the-art frame selection methods. We show both their performance and the their reported improvements over the base sampling method, due to the inconsistent base performance among different methods. Results of baselines are retrieved from literature, except for SMART*, which is implemented using the same features and implementation settings as SMS. We have also included the results of SMS learned only on 10% training data as SMS_10%.

epochs. For ActivityNet and FCVID, we use an initial learning rate of 0.005, and a batch size of 64. For UCF101 with shorter videos, we increase the batch size to 128 and adjust the initial learning rate to 0.00256. All models are trained using code adapted from MMAction2 [107] on eight Nvidia-A100 GPUs. And we report the average performance and standard deviation of three runs.

5.4.2 Effectiveness Analysis (RQ1)

To validate the effectiveness of the proposed SMS, we make a comprehensive comparison with the base method on both long-video dataset ActivityNet and FCVID, and short-video dataset UCF101. As the video lengths of different datasets vary in a large range, we choose to select different number of frames for each video in these datasets, where 8, 16 and 25 frames are selected for long videos in ActivityNet and FCVID, and 3 and 8 frames are selected for short videos in UCF101. Furthermore, we

conduct experiments to select frames with SMS only on test data during the inference of pre-trained base models, and incorporate SMS in both the training and inference process. The experimental results are shown in Table 5.2, from which we can observe that:

- When applied to test data, compared to the base sampling method, SMS (infer only) significantly improves the average mAP on different number of selected frames by 4.53% and 1.85% on ActivityNet and FCVID, respectively. While SMS (infer only) significantly improves model performance, SMS (train & infer) improves the mAP by 0.75% and 0.34% mAP on ActivityNet and FCVID. This observation demonstrates that the frames selected with the SMS method are beneficial to both model training and inference.

- While most other frame selection methods only focus on long untrimmed video tasks, SMS is also effective on short trimmed video tasks. Despite that in UCF101, the irrelevant parts of videos are trimmed off, which to a great degree limits the potential of frame selection, SMS still achieves steady improvement of 1.42% average mAP over the base method with the same number of selected frames.

5.4.3 Performance Comparison (RQ2)

In order to justify the benefit of our new learning paradigm of frame selection, we compare SMS with other classic frame selection methods on ActivityNet and FCVID. We choose ResNet as the backbone architecture of the action recognition model and report the performance with similar number of selected frames ranges from 8 to 10.

As the implementation and training details are different among the frame selection methods, the performances of the base models are inconsistent. Therefore, directly comparing the reported performances may be unfair. Moreover, due to the lack of

[1]The base performance in our paper is higher compared to it in other papers, due to the better codebase and training settings.

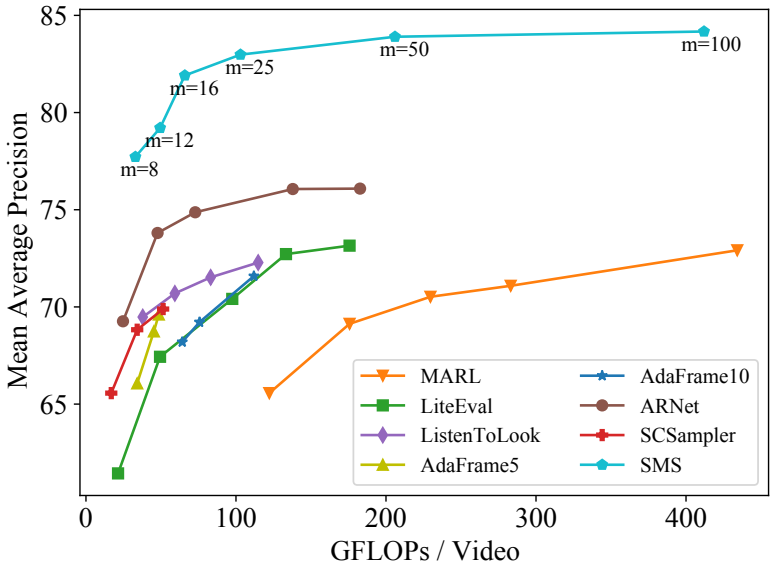


Figure 5.2: Comparison of SMS with other approaches in terms of performance vs. inference cost (model size per video) evaluated on ActivityNet. We control the inference cost by varying m, the number of candidate frames in a video from which $n = 8$ best frames are to be selected.

source codes and models, we are unable to compare all methods under the same settings, especially for the RL-based methods whose results are difficult to reproduce. To make a fairer comparison, in addition to the absolute performance, we also compare the relative improvements of the frame selection methods over the base sampling method. Also, we have implemented the most recent frame selection method, SMART [81], using the same features and implementation settings as our proposed SMS, and include its results as SMART*.

The comparison results are demonstrated in Table 5.3, from which we can see that with the least number of selected frames and the smallest backbone model, the proposed SMS achieves the best performance of 83.72% and 86.54% mAP on ActivityNet and FCVID respectively. Moreover, even with the higher-performance base models which are harder to improve, SMS still obtains the largest improvements of 6.48% and 2.70% among all frame selection methods. In a fair comparison with the same implementation settings, our method significantly outperforms SMART* by

Feature Extractor (Training Source)	Mapping Model Arch	Performance	Inference GFLOPs
ResNet-50 (Kinetics)	Transformer	83.72	1.50
ResNet-50 (Kinetics)	MLP	83.22	0.01
ResNet-50 (ImageNet)	Transformer	79.97	1.50
ResNet-50 (ActivityNet)	Transformer	81.06	1.50
MobileNet-V2 (Kinetics)	Transformer	79.53	0.93

Table 5.4: Ablation analysis results in mAP (%) on ActivityNet using 8 frames with different feature extractors and feature mapping model architectures. The feature mapping inference GFLOPs per video is also provided.

Backbone	# Frames	Method	Performance
TimeSFormer	8	Dense	84.33
		Base	90.11
		SMART*	90.53
		SMS	**91.97**

Table 5.5: The ActivityNet evaluation results in mAP (%) using different frame sampling strategies on TimeSFormer.

3.05% and 3.19% respectively on ActivityNet and FCVID.

The reason of our success is due to our novel training paradigm of "Search-Map-Search", where an efficient hierarchical search method is incorporated to find the best frame combinations, which better models the frame interactions. Moreover, in SMS, a feature mapping function is learned to map an input video directly to the optimal frame combination, which is theoretically superior to the one-by-one frame selection adopted by existing methods.

5.4.4 Efficiency Analysis (RQ3)

We now analyze the efficiency of SMS and compare it with other frame selection methods in terms of action recognition performance versus inference cost (evaluated by the model size per video. For a fair comparison, we only include the results that

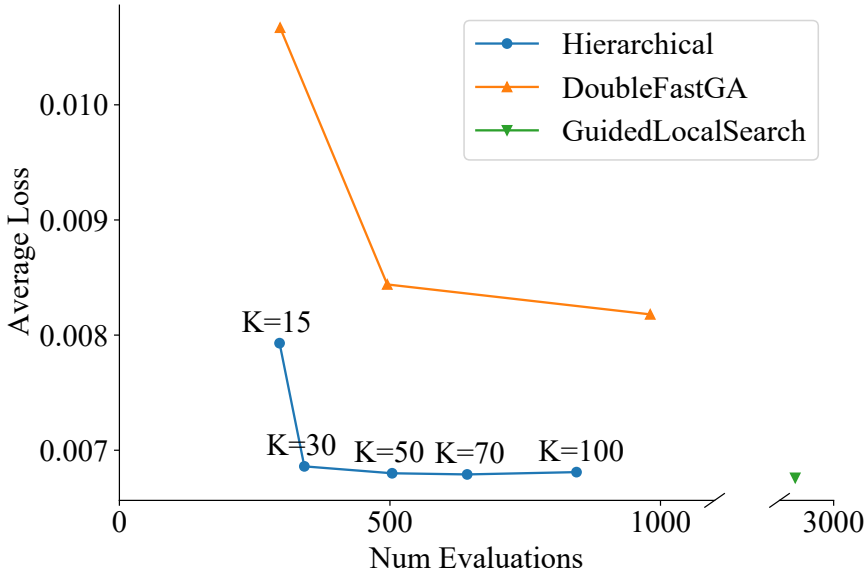


Figure 5.3: The evaluation of different search algorithms on 100 videos, given by the average loss over the number of evaluations.

use ResNet as the backbone model.

In order to evaluate the tradeoff between performance and inference efficiency, we first uniformly sub-sample m candidate frames that constitute the search space for each video from which $n = 8$ best frames are to be selected. As m increases, features are extracted from more candidate frames, and the resulting frame selection becomes more effective, while the inference cost becomes larger. As demonstrated in Figure 5.2, the action recognition performance of SMS grows rapidly from $m = 8$ to $m = 25$, while further increasing m to 50 or 100 incurs only slight increase in performance but huge inference overhead. In practice, one can easily achieve a good tradeoff between performance and cost accordingly by tuning the number of candidate frames m.

In comparison with other approaches, SMS achieves higher action recognition performance and beats other methods under different computation resource constraints, showing the effectiveness of the proposed search-mapping-search paradigm in frame selection.

5.4.5 Ablation Study (RQ4)

This subsection aims to analyze the effects of different components designed in our proposed SMS.

Search algorithm. We have conducted experiments to compare the performance and efficiency of our designed Hierarchical Guided Local Search algorithm with other powerful search algorithms such as Fast Genetic Algorithm [108] and the frame-level Guided Local Search [105].

In Figure 5.3, we show the best loss averaged on 100 randomly selected videos with different search algorithms, and their corresponding search cost measured by the number of evaluations. By adopting different clip length K, our proposed hierarchical search algorithm can trade off between the search performance and the search cost. From Figure 5.3, we can see that using the same number of evaluations, our hierarchical search achieves better results compared to Fast Genetic Algorithm, by more effectively exploiting the prior knowledge of the per-frame loss information. The original Guided Local Search without hierarchical design is extremely costly which requires nearly $3,000$ evaluations per video. Contrastively, our algorithm is more efficient with the hierarchical design, and achieves comparable performance with far less computation cost.

Feature mapping network. The process of feature mapping aims to transform the input frame features to the feature of target combination. We have conducted experiments to explore the impact of different network architectures and training data sources for feature mapping.

For the network architecture, we adopt transformers to sequentially modeling the frame features with their spatio-temporal relations taken into consideration. Another simple applicable design is to adopt a mean-pooling layer that aggregates all the features of frames into a single feature vector, followed by a simpler two-layer MLP network. In Table 6.4, row 1 and 2 compares the performance and the inference

efficiency of the two designs. As we can see, using transformer model as the mapping function outperforms MLP design by 0.5% mAP due to the better representation ability, while the inference cost of both designs are small and negligible (less than 2 GFLOPs) compared to the cost of frame feature extractions (tens or hundreds GFLOPs).

Feature extractor. We have analyzed the effect of different feature extractor settings by trying smaller model structure, e.g., MobileNet-V2 [109], and different pre-trained data sources including ImageNet [70], Kinetics [98] and ActivityNet [87].

Comparing the results of row 1, 3 and 4 in Table 6.4, we can see that the pre-training data of feature extractor can make a difference. The feature extractor trained on the largest Kinetics dataset achieves the best performance as it better captures the semantics of actions by training on more related samples, compared to the ones trained on smaller ActivityNet dataset and out-domain ImageNet dataset. Besides, as shown in row 5 of Table 6.4, using smaller models such as MobileNet-V2 for extractor can lead to performance decline. In general, the representation capability of feature extractors is valuable for frame selection to recognize the important frames and find the best frame combinations.

5.4.6 Generalizability Analysis (RQ5)

It is a natural question to ask if the frames selected by SMS can also be beneficial to spatio-temporal video models. To find out the answer to this question, we have conducted an experiment to apply the SMS selected frames on TimeSFormer [100], which incorporates the transformer architecture and is one of the most advanced video models.

The results are shown in Table 5.5. The dense frame sampling strategy incorporated in the original TimeSFormer implementation randomly samples a clip containing 8 successive frames from videos, and achieves 84.33% mAP on ActivityNet. However, in untrimmed video dataset, dense sampling can only captures a small part of the

video and may miss important information. In contrast, the base sampling strategy selects frames uniformly from videos and achieves 90.11% mAP, while SMART* achieves 90.53% mAP.

By using the input frames selected by SMS, we obtain a significant performance gain of 1.86% over the base sampling strategy and 1.44% over SMART*, and achieve 91.97% mAP. This improvement demonstrates the strong generalizability of SMS across different model architectures, and that SMS is not only effective on 2D video modeling, but can also capture the spatio-temporal relationship among frames and is beneficial to 3D video model learning.

5.5 Conclusion

In this chapter, we propose a new learning paradigm for frame selection, called "Search-Map-Search", which consists of a search stage to efficiently find the best frame combination with a hierarchical search algorithm, a feature mapping stage that learns to transform the input frame features directly into the feature of the searched combination, and another search stage that selects frames based on the mapped feature. Compared with existing frame selection methods, SMS is a more accurate learning paradigm that takes advantage of efficient search and supervised feature mapping to directly select the best combination of frames as one entity, which better captures the frame interactions. Experimental results show that SMS achieves significant performance gains on multiple action recognition benchmarks, and outperforms other strong baseline methods.

Chapter 6

Embedding Dimension Pruning

6.1 Introduction

The amount of information on the web has been growing exponentially, and becomes overwhelming for users, as they look for interesting and relevant information on the web. As a decent solution to this problem, recommender systems are a subclass of information filtering systems, making recommendations of items that are of particular interests to a user among the vast amount of available items, and are commonly deployed in a variety of areas such as online shopping [110], advertisement [111, 112], music apps [113], news apps [114], video apps [115], etc.

Compared with traditional recommender systems, modern deep-learning-based recommender systems (DLRSs) have achieved great success due to its capability of obtaining meaningful feature representations to model user preferences and item characteristics, especially for high-dimensional categorical features, e.g., age, address, product category, etc. Embedding tables are usually adopted in DLRSs and are responsible for mapping the high-dimensional sparse encoding vector for each category in each feature field into a low-dimensional dense feature vector. Although playing a critical role to recommendation performance, the embedding tables are usually exceptionally huge in real-world recommender systems with numerous users and items, leading to a dominating number of parameters in the model. Therefore, enhancing the design of the embedding layers has a great value to both reducing the model size

in DLRS and improving model performance, yet is an under-explored area of study.

Most existing recommender systems assign a uniform embedding dimension for all feature fields ignoring the discrepancy in importance of different features, which may lead to inefficient memory usage and sub-optimal recommendation performance. The reason is that, the embedding dimension of a feature field indicates the amount of useful information contained in the field. Assigning a small dimension to a highly informative and predictive feature, for example, the "last viewed movie" feature in a movie recommender system, may greatly limit the expressiveness of the feature and hence lead to performance degradation. On the other hand, assigning a large dimension to non-predictive features can incur unnecessarily large models, which leads to high memory usage and computational cost as well as overfitting issues.

Many studies have been conducted to address this issue by assigning different dimensions to different feature fields in embedding tables, e.g., by hand-crafted rules based on the popularity of features [116]. Recently, automated embedding dimension search has greatly advanced the recommendation performance and memory saving. Different strategies have been investigated to search for non-uniform embedding dimensions, including reinforcement learning [117, 118], evolutionary algorithms [119], gradient-based approaches [120–122], one-shot learning strategies [123, 124], and network pruning methods [125].

In this chapter, we propose, **DimReg**, which performs low-cost embedding dimension search in recommender systems only through regularized optimization. Not only does DimReg achieve better performance than prior methods under the same model reduction ratio, but it is also efficient, barely increasing the learning cost of the original deep recommender model. In particular, our contributions are summarized as follows:

- We use a scaling factor to control the magnitude of each embedding dimension in a feature field, which can be seen as an importance indicator [126]. We

achieve pruning of embedding by numerically regularizing the scaling factors of different dimensions in the feature vector during model training. By applying a polarization regularizer [127] generally on scaling factors of all embedding dimensions, the unimportant dimensions are well separated from important ones, and can be pruned without any performance loss.

- We observe that embedding dimensions in a feature field can be correlated and dimensions with similar information could be redundant and unnecessary. In order to reduce the redundancy, we propose a pruning criterion based on the intuition that different embedding dimensions should convey different information. In particular, we impose a polarization regularizer on pairs of highly correlated dimensions, which are identified iteratively during model training, in order to keep informative and representative dimensions and prune redundant ones.

- DimReg performs dimension pruning through regularized scaling factor optimization in conjunction with model weight training, adding minimum overhead to learning cost. Unlike other methods that require retraining the model weights after dimension search [122–124], DimReg does not require a separate re-training stage, which further reduces the computational cost and makes it easy to deploy to real-world large-scale recommender systems.

Extensive experiments are conducted on two classic click-through rate (CTR) prediction tasks, Avazu and Criteo. The experimental results show that DimReg can effectively prune the unimportant and redundant embedding dimensions without hurting the recommendation performance. With more than 98% of model parameters pruned, the recommendation model derived by DimReg still outperforms the original model and beats other strong embedding dimension search baselines.

6.2 Related Work

In this section, we give an introduction on existing embedding dimension search methods, which can be roughly categorized as heuristic methods, AutoML based methods, and pruning based methods.

The heuristic methods adopt hand-crafted rules to determine the embedding dimensions, where MDE [116] assigns features of higher popularity with larger embedding dimension and give fewer dimensions to the less popular ones.

Aiming at increasing the generality and reducing the human effort, automated learning of embedding dimension selection has also been widely explored. Some studies formulate this problem as a hyper-parameter optimization task by pre-defining a set of candidate dimensions for each feature field as the search space, and employ different search algorithms. ESAPN [117] and NIS [118] adopt reinforcement learning approaches to search for effective embedding solutions. RULE [119] splits the embedding tables into small blocks, and identifies the optimal elastic embedding composition with evolutionary algorithms. Despite the effectiveness, the induced search cost of these methods is very high and often unacceptable for real-world application deployment. Inspired by DARTS [14] in neural architecture search, AutoEmb [121] and AutoDim [122] formulate the problem in a differentiable manner and searches the suitable embedding dimensions within a collection pre-defined dimension candidates by gradient descent, which significantly improves the search efficiency. Similarly, aiming at reducing the search efficiency, SSEDS [123] and OptEmbed [124] adopt one-shot learning setting, where a big supernet is trained only once, and then used for the evaluation of different candidate solutions by sampling the corresponding subset part of the supernet. Although the DARTS based methods and one-shot learning based methods achieve fast search, they share some common problem: (1) they both require a retraining phase after the search is done, which can cost weeks or months extra training time for large-scale on-line recommender systems; (2) there exists per-

formance gaps between the evaluations during search and retraining, which may lead to sub-optimal solutions.

Another approach PEP [125], treats the problem of embedding dimension search as a model pruning problem, and adopts a learnable soft threshold to remove unimportant embedding parameters with small magnitudes. However, this method results in unstructured pruning, which does not alter the model architecture and relies on hardware and framework specially designed for sparse computation to achieve actual acceleration and memory saving.

Unlike most of existing methods that rely on two separate stages of searching and re-training, our method achieves efficient and effective dimension selection by pruning unimportant and redundant dimensions during the model training, with minimum training overhead introduced.

6.3 Problem Formulation

In this section, we first briefly introduce the typical click-through rate (CTR) prediction model and then present our formulation of the embedding dimension selection problem.

6.3.1 CTR Prediction Model

Figure 6.1 provides a brief overview of common deep CTR prediction models. Let $D^{\text{tr}} = \{(\mathbf{X}, y)\}^{|D^{\text{tr}}|}$ represent a training dataset, where the binary label y indicates whether a click event happens, while the input feature vector $\mathbf{X} = \{\mathbf{x_1}, \mathbf{x_2}, \cdots, \mathbf{x_m}\}$ consists of m *feature fields*, each field associated with an item property or a user characteristic. In reality, most of the feature fields are categorical features represented by one-hot or multi-hot encoding vectors. Sometimes, the number of categories in a feature field can reach several thousands or even millions, leading to high-dimensional and extremely sparse input features, where values of most dimensions are zeros.

In order to reduce feature dimensions and construct more meaningful represen-

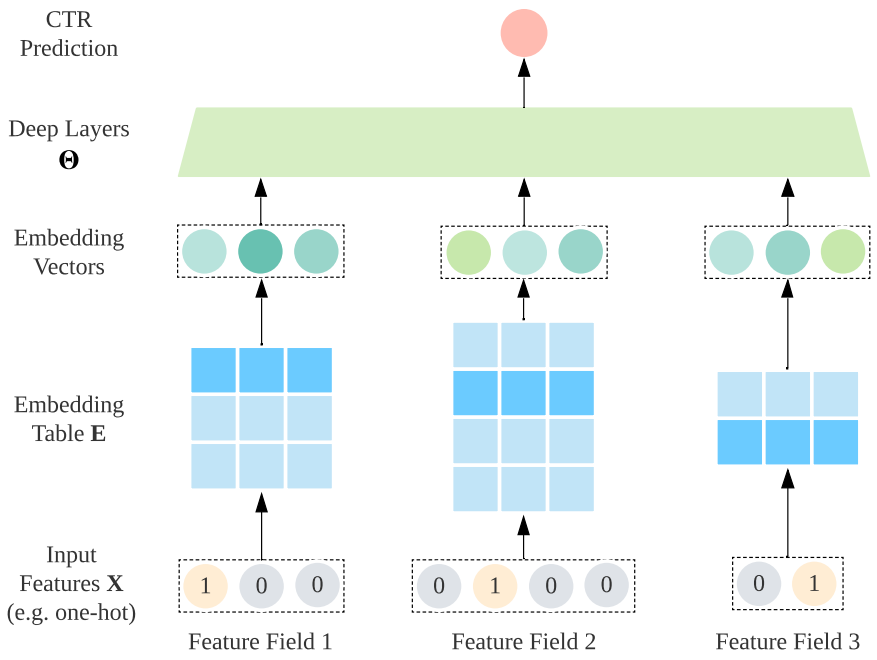


Figure 6.1: A typical CTR prediction network with three feature fields, where the input features are mapped to embedding vectors by embedding tables, where the deep layers convert the embeddings into the CTR prediction.

tations, an embedding layer is commonly employed to convert the high-dimensional sparse features into low-dimensional dense features. Technically, for a categorical feature in the i-th feature field denoted by $\mathbf{x_i} \in \mathbb{R}^{n_i}$ with n_i categories, the corresponding embedding vector $\mathbf{e_i} \in \mathbb{R}^d$ is derived as

$$\mathbf{e_i} = \mathbf{E_i x_i}, \tag{6.1}$$

where $\mathbf{E_i} \in \mathbb{R}^{d \times n_i}$ is the embedding table of i-th feature field and d denotes the embedding dimension.

Following the embedding layer, deep layers with different network structures are usually employed to transform the retrieved embeddings to predict the probability whether a click event happens. Typical network choices include feature interaction layers [128], deep neural networks [110] or both [129]. The training objective of a

CTR model f can be described as:

$$\min_{\mathbf{E},\mathbf{\Theta}} \quad \frac{1}{|D_{\text{tr}}|} \sum_{(x,y)\in D_{\text{tr}}} \mathcal{L}\big(y, f(\mathbf{X}|\mathbf{E},\mathbf{\Theta})\big), \tag{6.2}$$

where $\mathbf{E} = \{\mathbf{E_1}, \mathbf{E_2}, \cdots, \mathbf{E_m}\}$ is the collection of embedding tables of all feature fields, $\mathbf{\Theta}$ is the parameters of the network layers after the embedding layer, and $\mathcal{L}$ is the loss function (e.g., cross entropy).

6.3.2 Embedding Dimension Selection

The original CTR model adopts a uniform dimension d for all feature fields ignoring the discrepancy between different features, which may result in overly large embedding tables and sub-optimal performance. In order to generalize better while reducing model size, the task of embedding dimension selection focuses on assigning an appropriate dimension for each feature field.

Formally speaking, the objective of our embedding dimension selection problem is to optimize the training loss, regularized by the number of preserved embedding dimensions, i.e.,

$$\begin{aligned} \min_{\mathbf{E}',\mathbf{\Theta}} \quad & \frac{1}{|D_{\text{tr}}|} \sum_{(\mathbf{X},y)\in D_{\text{tr}}} \mathcal{L}\big(y, f(\mathbf{X}|\mathbf{E}',\mathbf{\Theta})\big) + \lambda \sum_{i=1}^{m} d_i \\ \text{s.t.} \quad & \mathbf{E}' = \left\{\mathbf{E_i'} \in \mathbb{R}^{n_i \times d_i} | d_i \leq d\right\}_{i=1}^{m}, \end{aligned} \tag{6.3}$$

where $\mathbf{E}_i'$ is a sub-table of $\mathbf{E}_i$ consisting of embedding dimensions selected from $\mathbf{E}_i$, and d_i is the preserved embedding dimension of field i and can be different for different feature fields.

In this chapter, we treat the embedding dimension selection in a model pruning manner. Initially, for each feature field i, we have a large original embedding table with d dimensions $\mathbf{E_i} \in \mathbb{R}^{n_i \times d}$. During the model training process, a part of unimportant and redundant embedding dimensions are progressively pruned. The important and informative dimensions form the resulting embedding table.

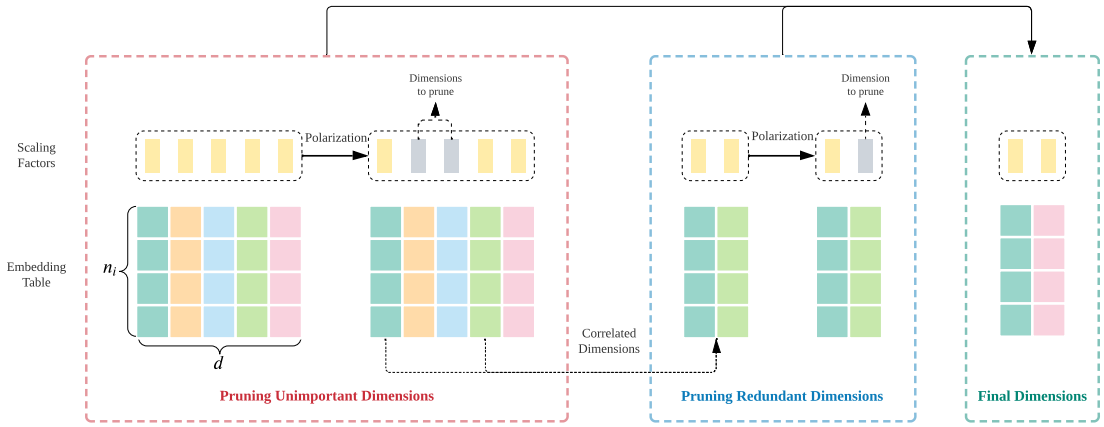


Figure 6.2: An overview of the proposed DimReg method. For each feature field, a general polarization regularizer is applied on the scaling factors of all embedding dimensions to prune the unimportant dimensions with scaling factors close to zero (grey). From the preserved dimensions, the correlated dimensions with similar meanings are selected and penalized by another regularization to prune the redundant dimensions.

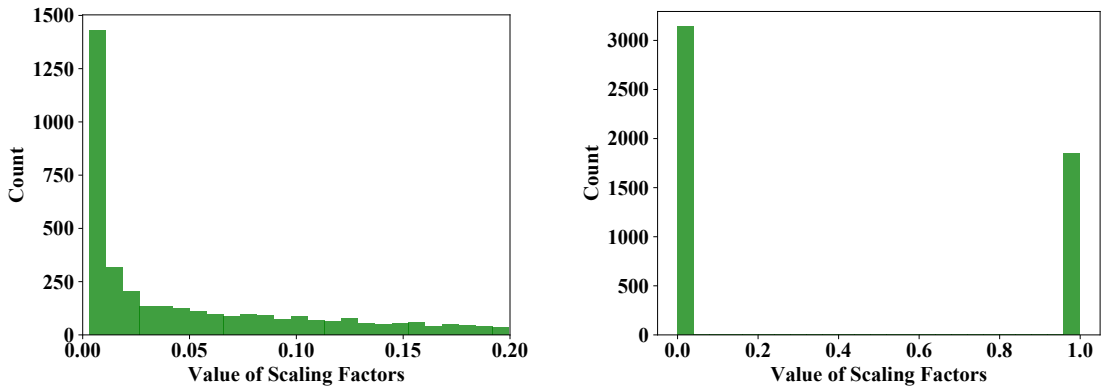
6.4 Methodology

In this section, we first provide a general overview of the proposed DimReg approach, and introduce our embedding dimension pruning method including the polarization regularization and our strategy of identifying unimportant and redundant dimensions.

6.4.1 Overview

Figure 6.2 provides an overview of the proposed DimReg approach, where for each feature field, we first multiply each of the embedding dimensions by a *scaling factor*, to obtain a scaled embedding table. By making predictions based on scaled embedding tables, we convert the hard regularized optimization problem in Equation (6.3) to regularization applied on the scaling factors, which are trained together with the model parameters. The objective is to suppress a part of the scaling factors to zero, such that embedding dimensions with nonzero scaling factors are identified as important and necessary.

In particular, we employ a two-level polarization to regularize the values of scaling



(a) L1 Regularization

(b) Polarization Regularization

Figure 6.3: The data distribution of scaling factors trained with L1 and polarization regularizers respectively.

factors. On one hand, polarization on the scaling factor of each individual dimension is performed to separate the pruned dimensions from preserved ones. On the other hand, to encourage the emergence of informative and representative dimensions and prune redundant ones, we compute the correlations between the preserved dimensions, and again apply polarization regularization on the scaling factors of highly correlated dimensions. Such correlation assessment is conducted dynamically and iteratively during model training. Our motivation is based on the hypothesis that the information carried by different embedding dimensions should be distinctive and representative to avoid redundancy.

6.4.2 Pruning through Regularization

In order to achieve the structured pruning of embedding dimensions, a common practice in the area of embedding dimension selection is to assign a switch variable to each candidate option controlling the probability whether to conduct this option or not [120–122]. In this chapter, we assign a learnable scaling parameter α_i^j to each embedding dimension j in the feature field i. Obviously, the scaling factor should be positive and bounded. Therefore, we use the sigmoid activation function upon the parameter to compute the actual scaling factor $\phi_{i,j} = \text{sigmoid}(\alpha_{i,j})$. Specifically, a scaled em-

bedding table $\tilde{\mathbf{E}}_\mathbf{i}$ is derived to replace the original embedding table $\mathbf{E}_\mathbf{i} = [\mathbf{V}_{\mathbf{i},\mathbf{j}}]_{j=1}^{d_i}$ by scaling each dimension $\mathbf{V}_{\mathbf{i},\mathbf{j}}$ with the factor $\phi_{i,j}$, i.e.,

$$\tilde{\mathbf{E}}_\mathbf{i} = [\phi_{i,j} \cdot \mathbf{V}_{\mathbf{i},\mathbf{j}}]_{j=1}^{d_i} . \tag{6.4}$$

The value of the scaling factor $\phi_{i,j}$ reflects the magnitudes of the dimension which is often deemed as a major indicator of importance [126]. Many works use this method to select embedding dimensions by dropping the options with small scaling factors, assuming a small scaling factor indicates that the corresponding embedding dimension is less important and shall be pruned. However, no matter how small the scaling factor is, removing a non-zero dimension will introduce perturbation to prediction results, which may cause performance degradation. In fact, as a remedy, most of existing methods adopt an individual re-training stage that trains the pruned network again from scratch, to reduce performance drop. Despite its effectiveness, re-training the model will induce huge computational cost in reality.

In this chapter, we propose to apply regularization techniques to push the scaling factors of unimportant and redundant dimensions down to zero and achieve sparse solutions during the model training process, which eliminates the risk of pruning non-zero dimensions as well as the costly re-training stage. The objective function for neural network training with regularization applied on scaling factors is:

$$\min_{\mathbf{E},\mathbf{\Phi},\mathbf{\Theta}} \quad \frac{1}{|D_{\text{tr}}|} \sum_{(x,y)\in D_{\text{tr}}} \mathcal{L}\big(y, f(\mathbf{X}|\tilde{\mathbf{E}}(\mathbf{E},\Phi),\mathbf{\Theta})\big) + \lambda R(\mathbf{\Phi}), \tag{6.5}$$

where λ is the hyper-parameter for the regularizer, $R(\cdot)$ is a regularization function, $\mathbf{\Phi}$ denotes a collection of all the scaling factors involved in regularization, and $\tilde{\mathbf{E}}$ is a collection of scaled embedding tables $\tilde{\mathbf{E}}_\mathbf{i}$ and a function that transforms the original embedding tables into scaled tables using $\mathbf{\Phi}$. Note that in our framework, after learning $\mathbf{E}$, $\mathbf{\Phi}$ and $\mathbf{\Theta}$ with proper regularization, we derive a pruned model with smaller size by by pruning the dimensions with zero-value scaling factors from the scaled embedding tables $\tilde{\mathbf{E}}$, which does not change the prediction results. Therefore, model retraining is not needed.

6.4.3 Regularization via Polarization

One classical regularizer that produces sparse solutions is L1 norm, i.e., $R(\mathbf{\Phi}) = ||\mathbf{\Phi}||_1$, the effect of which is to push as many scaling factors to zero as possible. Although some zero-valued factors are achieved, L1 regularization does not discriminate the pruned and preserved dimensions well. As shown in Figure 6.3a, the learned scaling factors are distributed densely over the values, and a non-trivial threshold value is required to conduct pruning.

In this chapter, rather than using an L1 regularizer, we perform the polarization regularization [127, 130] on scaling factors, which suppresses a proportion of the scaling factors while keeping others intact, better separating the pruned dimensions and preserved dimensions.

The polarization regularizer is formulated as:

$$\begin{aligned} R_p(\mathbf{\Phi}) &= t||\mathbf{\Phi}||_1 - ||\mathbf{\Phi} - \overline{\phi} \cdot \mathbb{1}||_1 \\ &= \sum_{\phi \in \mathbf{\Phi}} t|\phi| - |\phi - \overline{\phi}|, \end{aligned} \tag{6.6}$$

where t is a temperature hyper-parameter, $\overline{\phi} = \frac{1}{|\mathbf{\Phi}|} \sum_{\phi \in \mathbf{\Phi}} \phi$ is the mean value of all scaling factors, and $\mathbb{1}$ is a all-ones vector with the same size as $\mathbf{\Phi}$. In Equation 6.6, a new term $-||\mathbf{\Phi} - \overline{\phi} \cdot \mathbb{1}||_1$ is appended to the L1 regularization, which aims to push each scaling vector ϕ from the mean value $\overline{\phi}$. As the value of each scaling vector is bounded by sigmoid function $\phi \in (0, 1)$, the minimum value of $-||\mathbf{\Phi} - \overline{\phi} \cdot \mathbb{1}||_1$ is achieved when half of the values in $\mathbf{\Phi}$ are equal to zero while the other half are equal to one.

The temperature hyper-parameter t controls the proportion of zero-value scaling factors ρ. And the optimal solution of the polarization regularization term is achieved when:

$$\rho = \begin{cases} -t/4 + 1/2, & -2 \leq t \leq 2 \\ 0, & t > 2 \\ 1, & t < -2. \end{cases} \tag{6.7}$$

where the proof of Equation 6.7 can be referred to [118].

Figure 6.3b shows the data distribution of the scaling factors learned through polarization regularization. We can clearly observe that the scaling factors are well separated into two groups with few factors locate in between, where the scaling factors in the first group are very close to zero while the factors in the second group are very close to one. This polarization phenomenon provides a sparse solution where the pruned dimensions with clear zero-value scaling factors induces no performance drop, and the preserved dimensions are well distinguished from the pruned ones with no ambiguity.

6.4.4 Pruning of Unimportant Dimensions

By applying the polarization regularizer on different scaling factor groups $\mathbf{\Phi}$, we can achieve embedding dimension pruning in multiple levels.

In the first level, we aim to prune the embedding dimensions according to their importance to the model, evaluated by the values of their corresponding scaling factors, which represents the scale of the dimensions and serves as an excellent indicator of importance commonly adopted in model pruning methods [126]. In DimReg, we adopt the same importance pruning standard for embedding dimensions in different feature fields, by setting the group of scaling factors in Equation 6.6 as all scaling factors in m feature fields:

$$\mathbf{\Phi}^{\text{imp}} = \{\phi_{i,j}\}_{j=1\,i=1}^{d_i\quad m}. \tag{6.8}$$

Through fair competition, more important feature fields will have higher chance to keep more dimensions.

6.4.5 Pruning of Redundant Dimensions

Other than the importance pruning as all previous work focuses on, we propose to further prune the dimensions with redundant information also using polarization regularization.

In this chapter, we characterize the dimension redundancy as the information overlap between different embedding dimensions. Specifically, we define the information overlap $o_{i,jk}$ as the redundancy score between a pair of embedding dimensions $V_{i,j}$ and $V_{i,k}$ within the same feature field i using Pearson correlation coefficient [131] as:

$$o_{i,jk} = \text{Pearson}(V_{i,j}, V_{i,k}), \tag{6.9}$$

where Pearson$(\cdot)$ is the correlation function. The highly correlated dimension pairs convey redundant information and hence can be reduced to one dimension.

For determining which dimension pairs are redundant, we first calculate the redundancy scores for all feature fields $\mathbf{O}$:

$$\begin{aligned} \mathbf{O} &= \bigcup_{i=1}^{m} \mathbf{O_i} \\ \text{where} \quad \mathbf{O}_i &= \{o_{i,jk}\}_{j=1}^{d_i} {}_{k=j+1}^{d_i}, \end{aligned} \tag{6.10}$$

where $\mathbf{O}_i$ represents the redundancy scores of all dimension pairs in feature field i. Then, we rank the redundancy scores of dimension pairs in $\mathbf{O}$ and select a proportion with the top scores as the redundant dimension pairs set $\mathbb{V}$. The ratio of redundant pairs to all pairs is denoted as β.

Once the redundant pairs are found, we again apply polarization regularization on each redundant pairs $(V_{i,j}, V_{i,k})$ to softly push the scaling factor of one dimension to zero with Equation 6.6 by setting the factor group in Equation 6.6 as the corresponding two scaling factors:

$$\mathbf{\Phi}_{i,jk}^{red} = \{\phi_{i,j}, \phi_{i,k}\}. \tag{6.11}$$

6.4.6 Training Process of DimReg

The overall training process of DimReg is illustrated in Algorithm 5 where the goal is to derive a lightweight CTR model with excellent performance by pruning the unimportant and redundant embedding dimensions.

Algorithm 5: Training Process of DimReg

Input: Embedding Layer $\mathbf{E}$, Network Layers $\mathbf{\Theta}$, Scaling Factors $\mathbf{\Phi}$, Training Dataset D^{tr}, Training Iterations T, Redundant Pairs Update Interval l, Pruning Threshold γ

Output: Trained Pruned Model M_p

1 **for** step $= 0, 1, \cdots, T-1$ **do**
2 **if** step $\mid l$ **then**
3 Update $\mathbb{V}$ using Eq.6.10
4 Sample a batch $(\mathbf{X}, y)$ from D^{tr}
  /* Compute CTR Loss */
5 $L^{\text{CTR}} \leftarrow \mathcal{L}\big(y, f(\mathbf{X}|\mathbf{E}, \mathbf{\Phi}, \mathbf{\Theta})\big)$
  /* Compute importance regularization */
6 $R^{\text{imp}} \leftarrow R_p(\mathbf{\Phi}^{\text{imp}})$
  /* Compute redundancy regularization */
7 $R^{\text{red}} \leftarrow 0$
8 **for** $(V_{i,j}, V_{i,k}) \in \mathbb{V}$ **do**
9 $R^{\text{red}} \mathrel{+}= R_p\left(\mathbf{\Phi}^{\text{red}}_{i,jk}\right) \;/\; |\mathbb{V}|$
  /* Compute total loss */
10 $L^{\text{total}} = L^{\text{CTR}} + \lambda(R^{\text{imp}} + R^{\text{red}})$
11 Update $\mathbf{E}, \mathbf{\Phi}, \mathbf{\Theta}$ by minimizing L^{total} through gradient descent
12 Get pruned embedding layer $\mathbf{E}^* \subset \mathbf{E}$ by pruning unimportant and redundant dimensions with scaling factor less than γ

We first compute the redundant embedding dimension pairs using the current embedding table every l training iterations (line 2-3), the purpose of which is to stabilize the training and reduce the cost, because the redundancy pruning needs time to take effect. Then, for each training batch, we first compute the standard CTR loss (line 5), the importance regularization term with respect to all the scaling factors (line 6), and the redundancy regularization term as the average of all the regularization terms of each redundant dimension pairs (line 7-9). Finally, the total loss for this batch is computed by adding the CTR loss with the importance and redundancy regularization term scaled by λ, and is used to update the parameters of the CTR model including the embedding layer $\mathbf{E}$, the scaling parameters $\mathbf{\Phi}$, and the following layers $\mathbf{\Theta}$. After the training process is finished, the parameters of the model are

Table 6.1: The detailed information of datasets.

Dataset	# Instances	# Fields	# Features
Criteo	45M	39	1M
Avazu	40M	22	0.6M

well-trained, where the scaling factors are well separated into a group close to zero and a group close to one. According to the learned scaling factors, we can derive the pruned model by pruning the embedding dimensions with zero-value factors without any performance loss (line 12).

6.5 Experiments

In this section, we first describe the details of our experimental settings including the used datasets, network models, baseline methods, and the implementation details of DimReg. Then we evaluate the proposed DimReg, and compare it with previous methods in terms of both the recommendation performance and the efficiency i.e., the number of model parameters after pruning. Finally, we perform thorough ablative analysis on the design of different modules in DimReg. Our code will be released for validation and future studies.

6.5.1 Datasets, Network Models and Baselines

Datasets. Our experiments are conducted on two widely used public datasets. For both datasets, we use 80% for training, 10% for validation, and 10% for testing. We summarize the details of datasets in Table 6.1.

- **Criteo**[1] dataset is a real-world industry dataset for CTR prediction of online advertisements. It consists of 45 million user-click logs on ads. It has 24 categorical feature fields and 13 numerical feature fields. Following the best practice [132], we transform each numerical feature x value by $log^2(x)$, if $x > 2$.

[1] https://ailab.criteo.com/ressources/

In addition, we replace low frequency (less than 10) categorical features with a default "OOV" (i.e. out-of-vocabulary) token.

- **Avazu**[2] dataset consists of 40 million user-click logs on ads over 11 days. It has 22 categorical features. Following [132], we replace low frequency (less than 4) categorical features with a default "OOV" (i.e. out-of-vocabulary) token.

Network Models. We select three well known CTR prediction models including FM [128], DeepFM [129], and Wide&Deep [110] as the base network models in our experiments.

Baselines. We have compared our proposed DimReg method with the following state-of-the-art embedding dimension search methods.

- **MDE [116]**: Mixed Dimension Embedding (MDE) is a heuristic embedding dimension search method. It assigns the embedding dimension for each feature field based on its popularity.

- **AutoDim [122]**: AutoDim utilizes a differentiable neural architecture search [14] approach to find the best embedding dimensions from a pre-defined dimension search space.

- **PEP [125]**: PEP (short for Plug-in Embedding Pruning) learns trainable thresholds from data to remove redundant dimensions of the embedding table so that mixed-dimension sparse embeddings can be obtained to cut down the number of parameters while maintaining or boosting the performance.

- **DeepLight [133]**: DeepLight method proposes to prune not only the embedding parameters but also the DNN layers. For a fair comparison, Deeplight is only used for embedding table pruning while the DNN layers are not pruned.

[2]https://www.kaggle.com/competitions/avazu-ctr-prediction/data

- **SSEDS [123]**: SSEDS proposes a single-shot embedding dimension search method, which identifies the importance of each embedding dimension of each feature field efficiently through only one forward and backward pass. SSEDS uses a parameter budget hyperparameter to control the sparsity of the mixed-dimensional embedding table.

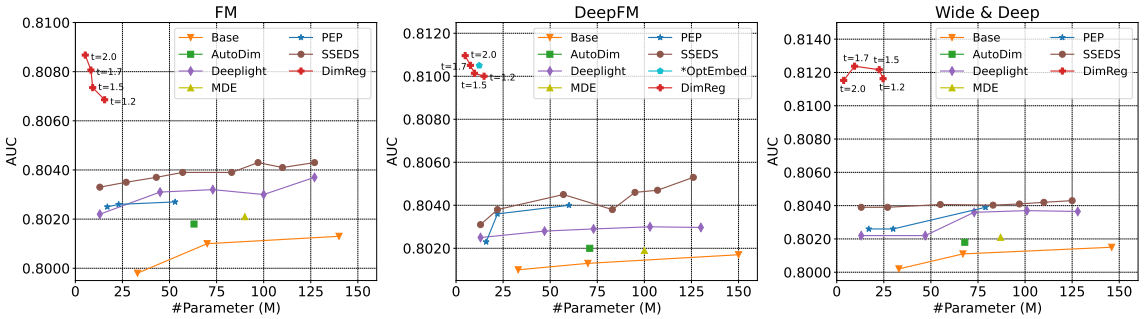
- **OptEmbed [124]**: OptEmbed method proposes to prune both the number of items as well as the dimensions of embedding tables. It utilizes learnable thresholds to prune embedding features based on its importance. OptEmbed uses a single supernet to train all candidate architectures in the embedding feature search space. In addition, it adopts a evolutionary search approach to find the embedding dimensions of each feature fields.

6.5.2 Implementation Details

We implement the proposed DimReg method based on PyTorch [134] and pytorch-fm[3]. Following previous work [123], we set the initial embedding dimension $d = 128$ for all feature fields, and incorporate two fully connected hidden layers with $1,024$ units in each layer and RELU activation function as the deep layers $\mathbf{\Theta}$ in DFM and Wide&Deep. As the feature interaction layer in FM based models (e.g., FM and DFM) requires the dimensions of all features to be the same, we incorporate a transform layer for each feature field that maps the pruned embeddings back to vectors of 128 dimensions. Adam optimizer [76] with a learning rate of $1e-3$ is used to update the network parameters. The batch size is set to $2,048$ for all datasets.

As to the hyper-parameters incorporated in DimReg, we use a regularization rate λ of $5e-5$, and set the ratio of redundant dimension pairs β to 0.1. During training, we re-calculate and update the redundant dimension pairs every 5 epochs. The pruning threshold for scaling factors γ is set to 0.001.

[3]https://github.com/rixwew/pytorch-fm



*OptEmbed adopts different training hyper-parameters derived using grid search.

Figure 6.4: Comparison results of different embedding dimension search methods in terms of test AUC over different number of model parameters on the Criteo dataset.

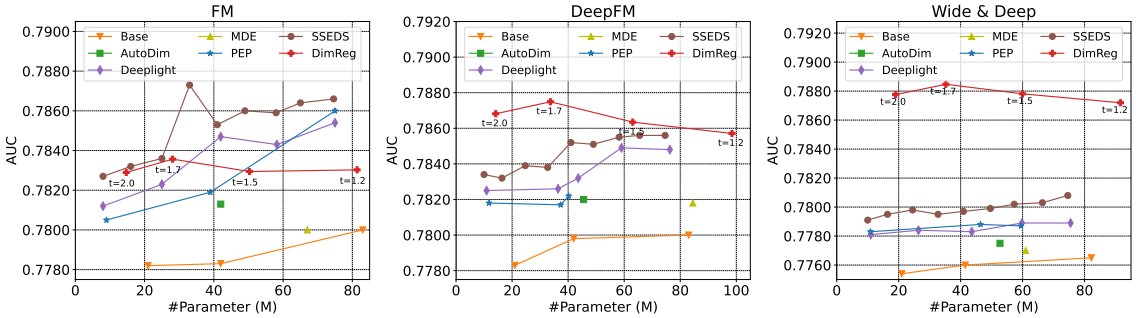


Figure 6.5: Comparison results of different embedding dimension search methods in terms of test AUC over different number of model parameters on the Avazu dataset.

The recommendation performance of DimReg is evaluated with the pruned embedding tables other than embedding tables of the original sizes with filling zeros in the pruned dimensions [124]. All experiments are conducted on a single Nvidia A100 GPU.

6.5.3 Experimental Results

To validate the effectiveness of the proposed DimReg, we conduct thorough comparison experiments to compare it with other embedding dimension search methods on three classic CTR networks, i.e., FM, DeepFM, and Wide&Deep. We follow the experimental setup of SSEDS [123], and adopt the same training hyper-parameters in the evaluation of our method and other baseline methods except for OptEmbed [124], which carefully tuned the hyperparameters of the recommendation

Table 6.2: Time and memory cost of the proposed DimReg method measured on Avazu evaluated using Wide&Deep on Avazu.

	DimReg	Re-train Based Methods
Training cost (GPU hours)	19.33	32.50
	DimReg	Unpruned Model
Inference cost (seconds)	39	45
	DimReg	Unpruned Model
Memory cost (megabytes)	73	1,009

model and produced results that are directly comparable to other baselines. We report the performance of DimReg under different sparsity levels using temperatures $t = \{2.0, 1.7, 1.5, 1.2\}$. The experimental results on the Criteo and Avazu datasets are demonstrated in Figure 6.4 and 6.5 respectively. We summarize our observations below.

First, all the models incorporating embedding dimension search methods have surpassed the base models with a unified embedding dimension in both the recommendation performance and the model efficiency, which verifies the superiority of assigning embedding dimensions to different feature fields with pertinence. Furthermore, automated learning methods tend to achieve better results than the heuristic method MDE.

DimReg achieves significant improvement on the recommendation performance compared to existing methods. On Criteo, DimReg achieves remarkable performance boost of 8-12‰ AUC scores compared to the uniform embedding dimension, and improves the previous best method by 6-8‰, which are huge improvements in recommender systems. Note that the results of OptEmbed are referenced from the paper which are reported on a different set of training hyper-parameters and can not be directly compared. As an indirect comparison, OptEmbed improves its baseline model by only 0.1‰which is not as significant. On Avazu, DimReg also beats other state-of-

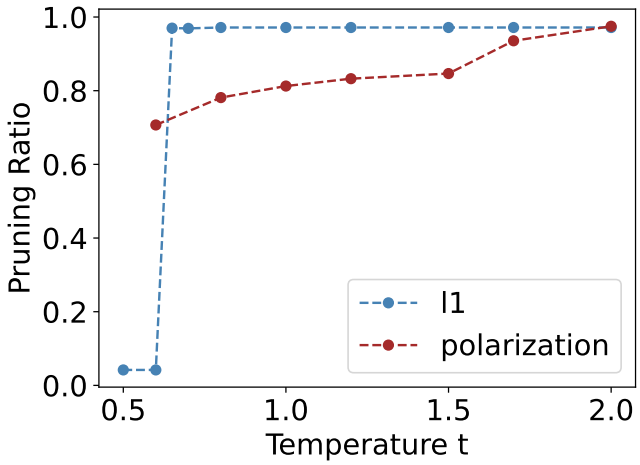


Figure 6.6: The hyper-parameter sensitivity test of L1 and polarization regularizers by plotting the pruning ratio of different temperature values.

the-art methods on DeepFM and Wide&Deep, and achieves comparable performance on FM.

In addtional to the exceptional recommendation performance, DimReg achieves a high average model suppression rate of 82.6% and 98.6% on Avazu and Criteo respectively. The extremely high suppression rate on Criteo is because DimReg significantly reduces the dimension of the feature fields with top-3 number of items from 128 to $1, 3, 2$ respectively, which greatly reduces the number of model parameters. This result indicates that DimReg can effectively recognize and prune the unimportant and redundant dimensions which contributes little or negatively to the recommendation performance.

6.5.4 Memory and Time Cost

The main purpose of embedding dimension pruning is to save cost of the recommendation systems. We have also conducted experiments to evaluate the memory cost as well as the training and inference cost of DimReg on Avazu using Wide&Deep, as shown in Table 6.2.

By pruning over 80% of the dimensions, DimReg significantly reduces the size

Table 6.3: Comparison results of the polarization regularization vs the L1 regularization on Criteo in terms of AUC and the number of parameters in pruned models.

	FM		DeepFM		Wide&Deep	
	AUC	# Params.	AUC	# Params.	AUC	# Params.
L1	0.8082	$5.07M$	0.8108	$4.85M$	0.8108	$3.70M$
Polarization	**0.8086**	$4.53M$	**0.8108**	$4.61M$	**0.8115**	$3.77M$

of the recommendation model from 1,009 megabytes to 73 megabytes reducing the memory cost by 93%.

For the training cost, compared with other embedding dimension search methods which require re-training the recommendation model under the same pruning ratio, DimReg can directly produce pruned model ready for deployment without re-training which significantly reduces the training cost from 32.50 GPU hours to 19.33 GPU hours.

As to the inference speed, the unpruned model takes 45 seconds to complete the inference of the validation set, while the pruned model by DimReg reduces the required time to 39 seconds, achieving 1.15× inference speedup.

6.5.5 Ablation Analysis

We have conducted comprehensive ablation analysis to evaluate the impact of different designs incorporated in our method, including the polarization regularization, the redundancy pruning, and the retrain-less setting.

Polarization Regularizer vs L1 Regularizer. As illustrated in Figure 6.3, the major advantage of the polarization regularizer is its capability of separating the scaling factors of the preserved and pruned dimensions, while L1 regularization pushes all scaling factors towards zero.

In Figure 6.6, we demonstrate the temperature sensitivity over the pruning ratio of the L1 regularizer i.e., $R(\mathbf{\Phi}) = t||\mathbf{\Phi}||_1$, and the polarization regularizer shown in

Table 6.4: Ablative experimental results on Criteo, where "DimReg w/o RP" removes the redundancy pruning from DimReg, and "DimReg + Retrain" re-trains the pruned model by DimReg.

	FM		DeepFM		Wide&Deep	
	AUC	# Params.	AUC	# Params.	AUC	# Params.
DimReg w/o PR	0.8085	$5.53M$	0.8108	$4.61M$	0.8109	$3.93M$
DimReg	**0.8087**	$5.26M$	**0.8110**	$4.85M$	0.8115	$3.70M$
DimReg + Retrain	0.8082	$5.26M$	**0.8110**	$4.85M$	**0.8116**	$3.70M$

Equation 6.6, by plotting the pruning ratio of the models over different values of the temperature hyper-parameter t. As we can see, the pruning ratio of L1 regularization is extremely sensitive to the temperature value, where the pruning ratio rises rapidly from 0.03 to 0.95 with a small increase on t from 0.6 to 0.65. The reason of this high sensitivity of L1 regularization is that a major part of the scaling factors regularized by L1 are located around the pruning threshold $\gamma = 0.001$ as L1 regularization pushes all factors towards zero, where a slight change on the temperature t would cause a great number of scaling factors to cross the threshold γ and lead to significant change on pruning ratio. This phenomenon of L1 regularization is undesired as in most cases, we want to have good control on the size of the pruned model to achieve the best trade-off between efficiency and performance. In contrast, the polarization regularizer can achieve easy control on the size of the recommendation model.

We have also conducted experiments to compare the recommendation performance of polarization regularization to L1 regularization on the Criteo dataset. To make a fair comparison, we compare the pruned models with similar number of parameters produced by polarization and L1 regularization. The results are shown in Table 6.3. We can see that under the similar model size, the polarization regularization beats the L1 regularization with all network structures, which proves its superiority.

Redundancy Pruning & Re-training. We conduct experiment to evaluate

Table 6.5: Comparison results of different functions to calculate redundancy scores of embedding dimensions shown in AUC evaluated on Avazu.

	Wide&Deep
Cosine	0.7875
L2	0.7870
Pearson	0.7878

the impact of the redundancy pruning. Table 6.4 demonstrates the results where we can see that with similar model size, DimReg achieves better AUC scores compared to the version without redundancy pruning, which validates that redundancy is an important criterion in dimension pruning, and removing redundant information is beneficial.

We have also assessed whether the re-training stage is needless in DimReg. In Table 6.4, we can observe that unlike previous methods, re-training the model pruned by DimReg achieves no significant performance change. This is because DimReg progressively prunes embedding dimensions during the model training, which produces no performance gap between the trained model and the pruned model for evaluation.

Redundancy score calculation. During the redundancy pruning, we have also conducted experiments on Avazu with Wide&Deep to explore the optimal function to calculate the dimension redundancy. In Table 6.5, we compare different functions including Cosine Similarity, L2 Similarity and the Pearson Correlation Coefficient. We can see that, using Pearson Correlation Coefficient to calculate the redundancy score of dimensions achieves an AUC of 0.7878 which is better than the AUC performance of 0.7875 and 0.7870 achieved by using cosine similarity and L2 similarity. This is probably because the Pearson correlation better captures the relationship between embedding dimensions in the semantic level, while the other measures focus more on the vector similarities.

6.6 Conclusion

In this chapter, we propose DimReg, an embedding dimension pruning method which prunes dimensions based on importance and redundancy via a two-level polarization regularizer. By effectively recognizing the important embedding dimensions and removing the redundant dimensions progressively during model training, DimReg achieves significant improvement over existing methods on both the recommendation performance and model efficiency, while introducing minimum overhead to model training.

Chapter 7

Conclusions and Future Directions

This chapter summarizes the main contributions of this dissertation and discusses possible future directions.

7.1 Conclusions

First, we discuss the automated data engineering approaches to improve the model learning including reinforced curriculum learning, and automated data augmentation.

In Chapter 3, we present our work on curriculum learning for neural machine translation with the goal of improving a pre-trained model when original batch training reaches its ceiling. We propose to formulate curriculum learning as a reinforcement learning problem to automate the learning of the training curriculum by iteratively re-selecting influential data samples from the original training set. We base our data selection framework on Deterministic Actor-Critic, in which a critic network predicts the expected change of model performance due to a certain sample, while an actor network learns to select the best sample out of a random batch of samples presented to it. By simple fine-tuning with the selected samples, we achieve significant improvement on several evaluation datasets and outperform other competitive methods.

In Chapter 4, we describe our experience on label-aware auto-augment ($LA3$) which takes advantage of the label information, and learns augmentation policies separately for samples of different labels. LA3 consists of two learning stages, where in the

first stage, individual augmentation methods are evaluated and ranked for each label via Bayesian Optimization aided by a neural predictor, which allows us to identify effective augmentation techniques for each label under a low search cost. And in the second stage, a composite augmentation policy is constructed out of a selection of effective as well as complementary augmentations, which produces significant performance boost and can be easily deployed in typical model training. *LA3* achieves excellent performance while maintaining a low computational cost.

Besides the model learning, we also propose to employ data engineering techniques to improve the model inference.

In Chapter 5, we propose a new automatic frame selection paradigm called *Search-Map-Search* for video action recognition tasks. We first propose a hierarchical search method conducted on each training video to search for the optimal combination of frames with the lowest error on the downstream task. A feature mapping function is then learned to map the frames of a video to the representation of its target optimal frame combination. During inference, another search is performed on an unseen video to select a combination of frames whose feature representation is close to the projected feature representation. By combining search with supervised learning, our method can better capture frame interactions while incurring a low inference overhead. Extensive experiments show that our frame selection method effectively improves performance of action recognition models.

In Chapter 6, we have focused on automating the search for better network architectures by proposing *DimReg* to search for the optimized embedding dimensions for recommendation system models. *DimReg* assesses information overlapping between the dimensions within each feature field and pruning unimportant and redundant dimensions progressively during model training via a two-level polarization regularizer, while introducing minimum overhead. Moreover, our method does not require retraining after embedding dimension search, which significantly reduces the computational cost and is more friendly to deployment in real-world recommender systems.

7.2 Future Directions

This thesis focuses on improving the learning and inference of deep models through automated data engineering approaches, which can be further extended in the following directions:

Interpretability. Although automated data engineering can find more promising data strategies than humans, there is a lack of scientific evidence for illustrating why the found policies perform better. For example, in the found augmentations of *LA3*, samples of certain classes prefer some augmentation operation, e.g., posterize, while other classes would suffer from this augmentation, where the reason is unclear. Therefore, increasing the mathematical interpretability is an important future research direction.

Generalization. Many strategies are learned specifically for targeted datasets and cannot be transferred directly to other datasets or domains. Learning a well-generalized strategy for datasets sharing similar characteristics is an important direction that can save plenty of time and efforts. A similar direction is Meta-RL [135]. Pre-training data strategies on large general datasets and fine-tuning them on specific target datasets may be a worth exploring direction.

Jointly Learning. My works are all focused on a single data engineering problem at a time. However, the construction of a deep learning system involves multiple data processing procedures, and learning them separately is highly costly in both computation and time. Therefore, constructing an automated data engineering pipeline that allows jointly learning of multiple data processing problems is a promising future direction.

Bibliography

[1] OpenAI, *Gpt-4 technical report*, 2023. arXiv: 2303.08774 [cs.CL].

[2] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 10 684–10 695.

[3] S. E. Whang and J.-G. Lee, "Data collection and quality challenges for deep learning," *Proceedings of the VLDB Endowment*, vol. 13, no. 12, pp. 3429–3432, 2020.

[4] M. Drosou, H. V. Jagadish, E. Pitoura, and J. Stoyanovich, "Diversity in big data: A review," *Big data*, vol. 5, no. 2, pp. 73–84, 2017.

[5] L. Ouyang *et al.*, "Training language models to follow instructions with human feedback," *Advances in Neural Information Processing Systems*, vol. 35, pp. 27 730–27 744, 2022.

[6] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," *The Journal of Machine Learning Research*, vol. 20, no. 1, pp. 1997–2017, 2019.

[7] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proceedings of the 26th annual international conference on machine learning*, 2009, pp. 41–48.

[8] X. He, K. Zhao, and X. Chu, "Automl: A survey of the state-of-the-art," *Knowledge-Based Systems*, vol. 212, p. 106 622, 2021.

[9] Z. B. Zabinsky *et al.*, "Random search algorithms," *Department of Industrial and Systems Engineering, University of Washington, USA*, 2009.

[10] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. De Freitas, "Taking the human out of the loop: A review of bayesian optimization," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 148–175, 2015.

[11] P. A. Vikhar, "Evolutionary algorithms: A critical review and its future prospects," in *2016 International conference on global trends in signal processing, information computing and communication (ICGTSPICC)*, IEEE, 2016, pp. 261–265.

[12] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of artificial intelligence research*, vol. 4, pp. 237–285, 1996.

[13] S. Santra, J.-W. Hsieh, and C.-F. Lin, "Gradient descent effects on differential neural architecture search: A survey," *IEEE Access*, vol. 9, pp. 89 602–89 618, 2021.

[14] H. Liu, K. Simonyan, and Y. Yang, "Darts: Differentiable architecture search," in *International Conference on Learning Representations*, 2018.

[15] B. F. Skinner, "Reinforcement today.," *American Psychologist*, vol. 13, no. 3, p. 94, 1958.

[16] G. B. Peterson, "A day of great illumination: Bf skinner's discovery of shaping," *Journal of the experimental analysis of behavior*, vol. 82, no. 3, pp. 317–328, 2004.

[17] K. A. Krueger and P. Dayan, "Flexible shaping: How learning in small steps helps," *Cognition*, vol. 110, no. 3, pp. 380–394, 2009.

[18] N. Kalchbrenner and P. Blunsom, "Recurrent continuous translation models," in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 2013, pp. 1700–1709.

[19] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[20] J. Gehring, M. Auli, D. Grangier, and Y. Dauphin, "A convolutional encoder model for neural machine translation," in *Proceedings of the 55th ACL*, 2017, pp. 123–135.

[21] A. Vaswani *et al.*, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[22] E. A. Platanios, O. Stretcu, G. Neubig, B. Poczos, and T. Mitchell, "Competence-based curriculum learning for neural machine translation," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2019, pp. 1162–1172.

[23] W. Wang, T. Watanabe, M. Hughes, T. Nakagawa, and C. Chelba, "Denoising neural machine translation training with trusted data and online data selection," in *Proceedings of the Third Conference on Machine Translation: Research Papers*, 2018, pp. 133–143.

[24] M. van der Wees, A. Bisazza, and C. Monz, "Dynamic data selection for neural machine translation," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2017, pp. 1400–1410.

[25] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *International Conference on Machine Learning*, 2014, pp. 387–395.

[26] K. Fan, J. Wang, B. Li, F. Zhou, B. Chen, and L. Si, ""bilingual expert" can find translation errors," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 6367–6374.

[27] T. Kocmi and O. Bojar, "Curriculum learning and minibatch bucketing in neural machine translation," in *Proceedings of RANLP 2017*, 2017, pp. 379–386.

[28] X. Zhang *et al.*, "An empirical exploration of curriculum learning for neural machine translation," *arXiv preprint arXiv:1811.00739*, 2018.

[29] Y. Tsvetkov, M. Faruqui, W. Ling, B. MacWhinney, and C. Dyer, "Learning the curriculum with bayesian optimization for task-specific word representation learning," in *Proceedings of the 54th ACL (Volume 1: Long Papers)*, 2016, pp. 130–139.

[30] A. Graves, M. G. Bellemare, J. Menick, R. Munos, and K. Kavukcuoglu, "Automated curriculum learning for neural networks," in *Proceedings of the 34th ICML-Volume 70*, JMLR. org, 2017, pp. 1311–1320.

[31] J. Wu, L. Li, and W. Y. Wang, "Reinforced co-training," *arXiv preprint arXiv:1804.06035*, 2018.

[32] G. Kumar, G. Foster, C. Cherry, and M. Krikun, "Reinforcement learning based curriculum optimization for neural machine translation," in *Proceedings of the 2019 NACCL*, 2019, pp. 2054–2061.

[33] B. Settles, "Active learning literature survey," University of Wisconsin-Madison Department of Computer Sciences, Tech. Rep., 2009.

[34] G. Haffari, M. Roy, and A. Sarkar, "Active learning for statistical phrase-based machine translation," in *Proceedings of Human Language Technologies: NACCL 2009*, Association for Computational Linguistics, 2009, pp. 415–423.

[35] M. Bloodgood and C. Callison-Burch, "Bucking the trend: Large-scale cost-focused active learning for statistical machine translation," in *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics, 2010, pp. 854–864.

[36] M. Fang, Y. Li, and T. Cohn, "Learning how to active learn: A deep reinforcement learning approach," in *Proceedings of the 2017 EMNLP*, 2017, pp. 595–605.

[37] M. Liu, W. Buntine, and G. Haffari, "Learning to actively learn neural machine translation," in *Proceedings of the 22nd Conference on Computational Natural Language Learning*, 2018, pp. 334–344.

[38] V. R. Konda and J. N. Tsitsiklis, "Actor-critic algorithms," in *Advances in neural information processing systems*, 2000, pp. 1008–1014.

[39] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, no. 3-4, pp. 229–256, 1992.

[40] G. Tesauro, "Practical issues in temporal difference learning," in *Advances in neural information processing systems*, 1992, pp. 259–266.

[41] M. Li and I. K. Sethi, "Confidence-based active learning," *IEEE transactions on pattern analysis and machine intelligence*, vol. 28, no. 8, pp. 1251–1261, 2006.

[42] Y. Yang, Z. Ma, F. Nie, X. Chang, and A. G. Hauptmann, "Multi-class active learning by uncertainty sampling with diversity maximization," *International Journal of Computer Vision*, vol. 113, no. 2, pp. 113–127, 2015.

[43] O. Sener and S. Savarese, "Active learning for convolutional neural networks: A core-set approach," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=H1aIuk-RW.

[44] D. So, Q. Le, and C. Liang, "The evolved transformer," in *International Conference on Machine Learning*, 2019, pp. 5877–5886.

[45] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: A method for automatic evaluation of machine translation," in *Proceedings of the 40th ACL*, Association for Computational Linguistics, 2002, pp. 311–318.

[46] A. Paszke, S. Gross, S. Chintala, and G. Chanan, "Pytorch: Tensors and dynamic neural networks in python with strong gpu acceleration," *PyTorch: Tensors and dynamic neural networks in Python with strong GPU acceleration*, vol. 6, 2017.

[47] S. Bird, E. Klein, and E. Loper, *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.", 2009.

[48] G. Klein, Y. Kim, Y. Deng, J. Senellart, and A. M. Rush, "OpenNMT: Open-source toolkit for neural machine translation," in *Proc. ACL*, 2017. DOI: 10.18653/v1/P17-4012. [Online]. Available: https://doi.org/10.18653/v1/P17-4012.

[49] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations (ICLR)*, 2015.

[50] Z. Shangtong, *Modularized implementation of deep rl algorithms in pytorch*, https://github.com/ShangtongZhang/DeepRL, 2018.

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

[52] E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le, "Randaugment: Practical automated data augmentation with a reduced search space," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020, pp. 702–703.

[53] E. D. Cubuk, B. Zoph, D. Mané, V. Vasudevan, and Q. V. Le, "Autoaugment: Learning augmentation strategies from data," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2019, pp. 113–123.

[54] D. Ho, E. Liang, X. Chen, I. Stoica, and P. Abbeel, "Population based augmentation: Efficient learning of augmentation policy schedules," in *International Conference on Machine Learning*, PMLR, 2019, pp. 2731–2741.

[55] S. Lim, I. Kim, T. Kim, C. Kim, and S. Kim, "Fast autoaugment," *Advances in Neural Information Processing Systems*, vol. 32, pp. 6665–6675, 2019.

[56] X. Zhang, Q. Wang, J. Zhang, and Z. Zhong, "Adversarial autoaugment," in *International Conference on Learning Representations*, 2019.

[57] F. Zhou *et al.*, "Metaaugment: Sample-aware data augmentation policy learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, 2021, pp. 11 097–11 105.

[58] H. Peng, F. Long, and C. Ding, "Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 27, no. 8, pp. 1226–1238, 2005.

[59] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "Mixup: Beyond empirical risk minimization," in *International Conference on Learning Representations*, 2018.

[60] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo, "Cutmix: Regularization strategy to train strong classifiers with localizable features," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 6023–6032.

[61] K. Tian, C. Lin, M. Sun, L. Zhou, J. Yan, and W. Ouyang, "Improving autoaugment via augmentation-wise weight sharing," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[62] R. Hataya, J. Zdenek, K. Yoshizoe, and H. Nakayama, "Faster autoaugment: Learning augmentation strategies using backpropagation," in *European Conference on Computer Vision*, Springer, 2020, pp. 1–16.

[63] Y. Li, G. Hu, T. Hospedales, N. Robertson, Y. Yang, *et al.*, "Dada: Differentiable automatic data augmentation," in *European Conference on Computer Vision 2020*, Springer, 2020.

[64] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," *arXiv preprint arXiv:1611.01144*, 2016.

[65] M. Jaderberg *et al.*, "Population based training of neural networks," *arXiv preprint arXiv:1711.09846*, 2017.

[66] Y. Wang *et al.*, "Fine-grained autoaugmentation for multi-label classification," *arXiv preprint arXiv:2107.05384*, 2021.

[67] C. White, W. Neiswanger, and Y. Savani, "Bananas: Bayesian optimization with neural architectures for neural architecture search," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, 2021, pp. 10 293–10 301.

[68] W. Wen, H. Liu, Y. Chen, H. Li, G. Bender, and P.-J. Kindermans, "Neural predictor for neural architecture search," in *European Conference on Computer Vision*, Springer, 2020, pp. 660–676.

[69] A. Krizhevsky, G. Hinton, *et al.*, "Learning multiple layers of features from tiny images," 2009.

[70] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, Ieee, 2009, pp. 248–255.

[71] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[72] S. Zagoruyko and N. Komodakis, "Wide residual networks," in *British Machine Vision Conference 2016*, British Machine Vision Association, 2016.

[73] X. Gastaldi, "Shake-shake regularization," *arXiv preprint arXiv:1705.07485*, 2017.

[74] D. Han, J. Kim, and J. Kim, "Deep pyramidal residual networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 5927–5935.

[75] H. Inoue, "Data augmentation by pairing samples for images classification," *arXiv preprint arXiv:1801.02929*, 2018.

[76] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[77] L. Wang *et al.*, "Temporal segment networks: Towards good practices for deep action recognition," in *European conference on computer vision*, Springer, 2016, pp. 20–36.

[78] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," *Advances in neural information processing systems*, vol. 27, 2014.

[79] J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici, "Beyond short snippets: Deep networks for video classification," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 4694–4702.

[80] B. Korbar, D. Tran, and L. Torresani, "Scsampler: Sampling salient clips from video for efficient action recognition," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 6232–6242.

[81] S. N. Gowda, M. Rohrbach, and L. Sevilla-Lara, "Smart frame selection for action recognition," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, 2021, pp. 1451–1459.

[82] Z. Wu, C. Xiong, C.-Y. Ma, R. Socher, and L. S. Davis, "Adaframe: Adaptive frame selection for fast video recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 1278–1287.

[83] W. Wu, D. He, X. Tan, S. Chen, and S. Wen, "Multi-agent reinforcement learning based frame sampling for effective untrimmed video recognition," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 6222–6231.

[84] S. Yeung, O. Russakovsky, G. Mori, and L. Fei-Fei, "End-to-end learning of action detection from frame glimpses in videos," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2678–2687.

[85] H. Fan, Z. Xu, L. Zhu, C. Yan, J. Ge, and Y. Yang, "Watching a small portion could be as good as watching all: Towards efficient video classification," in *IJCAI International Joint Conference on Artificial Intelligence*, 2018.

[86] W. Dong, Z. Zhang, and T. Tan, "Attention-aware sampling via deep reinforcement learning for action recognition," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 8247–8254.

[87] F. Caba Heilbron, V. Escorcia, B. Ghanem, and J. Carlos Niebles, "Activitynet: A large-scale video benchmark for human activity understanding," in *Proceedings of the ieee conference on computer vision and pattern recognition*, 2015, pp. 961–970.

[88] Y.-G. Jiang, Z. Wu, J. Wang, X. Xue, and S.-F. Chang, "Exploiting feature and class relationships in video categorization with regularized deep neural networks," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 2, pp. 352–364, 2017.

[89] K. Soomro, A. R. Zamir, and M. Shah, "Ucf101: A dataset of 101 human actions classes from videos in the wild," *arXiv preprint arXiv:1212.0402*, 2012.

[90] J. Donahue *et al.*, "Long-term recurrent convolutional networks for visual recognition and description," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 2625–2634.

[91] D. Li, Z. Qiu, Q. Dai, T. Yao, and T. Mei, "Recurrent tubelet proposal and recognition networks for action detection," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 303–318.

[92] L. Fan *et al.*, "Rubiksnet: Learnable 3d-shift for efficient video action recognition," in *European Conference on Computer Vision*, Springer, 2020, pp. 505–521.

[93] J. Lin, C. Gan, and S. Han, "Tsm: Temporal shift module for efficient video understanding," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 7083–7093.

[94] C. Luo and A. L. Yuille, "Grouped spatial-temporal aggregation for efficient action recognition," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 5512–5521.

[95] C. Feichtenhofer, H. Fan, J. Malik, and K. He, "Slowfast networks for video recognition," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 6202–6211.

[96] C. Feichtenhofer, A. Pinz, and A. Zisserman, "Convolutional two-stream network fusion for video action recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 1933–1941.

[97] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3d convolutional networks," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 4489–4497.

[98] J. Carreira and A. Zisserman, "Quo vadis, action recognition? a new model and the kinetics dataset," in *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 6299–6308.

[99] C. Feichtenhofer, "X3d: Expanding architectures for efficient video recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 203–213.

[100] G. Bertasius, H. Wang, and L. Torresani, "Is space-time attention all you need for video understanding?" In *ICML*, vol. 2, 2021, p. 4.

[101] Z. Liu *et al.*, "Video swin transformer," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 3202–3211.

[102] Z. Wu, C. Xiong, Y.-G. Jiang, and L. S. Davis, "Liteeval: A coarse-to-fine framework for resource efficient video recognition," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[103] R. Gao, T.-H. Oh, K. Grauman, and L. Torresani, "Listen to look: Action recognition by previewing audio," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 10 457–10 467.

[104] Y. Meng *et al.*, "Ar-net: Adaptive frame resolution for efficient action recognition," in *European Conference on Computer Vision*, Springer, 2020, pp. 86–104.

[105] C. Voudouris, E. P. Tsang, and A. Alsheddy, "Guided local search," in *Handbook of metaheuristics*, Springer, 2010, pp. 321–361.

[106] D. Whitley, "A genetic algorithm tutorial," *Statistics and computing*, vol. 4, no. 2, pp. 65–85, 1994.

[107] M. Contributors, *Openmmlab's next generation video understanding toolbox and benchmark*, https://github.com/open-mmlab/mmaction2, 2020.

[108] B. Doerr, H. P. Le, R. Makhmara, and T. D. Nguyen, "Fast genetic algorithms," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2017, pp. 777–784.

[109] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.

[110] H.-T. Cheng *et al.*, "Wide & deep learning for recommender systems," in *Proceedings of the 1st workshop on deep learning for recommender systems*, 2016, pp. 7–10.

[111] G. Zhou *et al.*, "Deep interest network for click-through rate prediction," in *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, 2018, pp. 1059–1068.

[112] G. Zhou *et al.*, "Deep interest evolution network for click-through rate prediction," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, 2019, pp. 5941–5948.

[113] B. L. Pereira, A. Ueda, G. Penha, R. L. Santos, and N. Ziviani, "Online learning to rank for sequential music recommendation," in *Proceedings of the 13th ACM Conference on Recommender Systems*, 2019, pp. 237–245.

[114] G. Zheng *et al.*, "Drn: A deep reinforcement learning framework for news recommendation," in *Proceedings of the 2018 world wide web conference*, 2018, pp. 167–176.

[115] J. Davidson *et al.*, "The youtube video recommendation system," in *Proceedings of the fourth ACM conference on Recommender systems*, 2010, pp. 293–296.

[116] A. A. Ginart, M. Naumov, D. Mudigere, J. Yang, and J. Zou, "Mixed dimension embeddings with application to memory-efficient recommendation systems," in *2021 IEEE International Symposium on Information Theory (ISIT)*, IEEE, 2021, pp. 2786–2791.

[117] H. Liu, X. Zhao, C. Wang, X. Liu, and J. Tang, "Automated embedding size search in deep recommender systems," in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2020, pp. 2307–2316.

[118] M. R. Joglekar *et al.*, "Neural input search for large scale recommendation models," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 2387–2397.

[119] T. Chen, H. Yin, Y. Zheng, Z. Huang, Y. Wang, and M. Wang, "Learning elastic embeddings for customizing on-device recommenders," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021, pp. 138–147.

[120] B. Yan *et al.*, "Learning effective and efficient embedding via an adaptively-masked twins-based layer," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2021, pp. 3568–3572.

[121] X. Zhaok *et al.*, "Autoemb: Automated embedding dimensionality search in streaming recommendations," in *2021 IEEE International Conference on Data Mining (ICDM)*, IEEE, 2021, pp. 896–905.

[122] X. Zhao *et al.*, "Autodim: Field-aware embedding dimension searchin recommender systems," in *Proceedings of the Web Conference 2021*, 2021, pp. 3015–3022.

[123] L. Qu, Y. Ye, N. Tang, L. Zhang, Y. Shi, and H. Yin, "Single-shot embedding dimension search in recommender system," *arXiv preprint arXiv:2204.03281*, 2022.

[124] F. Lyu *et al.*, "Optembed: Learning optimal embedding table for click-through rate prediction," *arXiv preprint arXiv:2208.04482*, 2022.

[125] S. Liu, C. Gao, Y. Chen, D. Jin, and Y. Li, "Learnable embedding sizes for recommender systems," in *International Conference on Learning Representations*, 2020.

[126] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," *arXiv preprint arXiv:1608.08710*, 2016.

[127] T. Zhuang, Z. Zhang, Y. Huang, X. Zeng, K. Shuang, and X. Li, "Neuron-level structured pruning using polarization regularizer," *Advances in neural information processing systems*, vol. 33, pp. 9865–9877, 2020.

[128] S. Rendle, "Factorization machines," in *2010 IEEE International conference on data mining*, IEEE, 2010, pp. 995–1000.

[129] H. Guo, R. Tang, Y. Ye, Z. Li, and X. He, "Deepfm: A factorization-machine based neural network for ctr prediction," in *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, 2017, pp. 1725–1731.

[130] Y. Guo, H. Yuan, J. Tan, Z. Wang, S. Yang, and J. Liu, "Gdp: Stabilized neural network pruning via gates with differentiable polarization," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 5239–5250.

[131] J. Benesty, J. Chen, Y. Huang, and I. Cohen, "Pearson correlation coefficient," in *Noise reduction in speech processing*, Springer, 2009, pp. 1–4.

[132] J. Zhu, J. Liu, S. Yang, Q. Zhang, and X. He, "Open benchmarking for click-through rate prediction," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2021, pp. 2759–2769.

[133] W. Deng, J. Pan, T. Zhou, D. Kong, A. Flores, and G. Lin, "Deeplight: Deep lightweight feature interactions for accelerating ctr predictions in ad serving," in *Proceedings of the 14th ACM international conference on Web search and data mining*, 2021, pp. 922–930.

[134] A. Paszke *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, 2019.

[135] J. Beck *et al.*, "A survey of meta-reinforcement learning," *arXiv preprint arXiv:2301.08028*, 2023.