2023

Towards Efficient and Effective Representation Learning for Image and Video Understanding

Taojiannan Yang
University of Central Florida

TOWARDS EFFICIENT AND EFFECTIVE REPRESENTATION LEARNING FOR IMAGE AND VIDEO UNDERSTANDING

by

TAOJIANNAN YANG
B.S. University of Science and Technology of China, 2017

A dissertation submitted in partial fulfilment of the requirements
for the degree of Doctor of Philosophy
in the Department of Computer Science
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Summer Term
2023

Major Professor: Chen Chen

ABSTRACT

Deep learning has achieved tremendous success on various computer vision tasks. However, deep learning methods and models are usually computationally expensive, making it hard to train and deploy, especially on resource-constrained devices. In this dissertation, we explore how to improve the efficiency and effectiveness of deep learning methods from various perspectives. We first propose a new learning method to learn computationally adaptive representations. Traditional neural networks are static. However, our method trains adaptive neural networks that can adjust their computational cost during runtime, avoiding the need to train and deploy multiple networks for dynamic resource budgets. Next, we extend our method to learn adaptive spatiotemporal representations to solve various video understanding tasks such as video recognition and action detection. Then, inspired by the proposed adaptive learning method, we propose a new regularization method to learn better representations for the full network. Our method regularizes the full network by ensuring that its predictions align with those of its sub-networks when fed with differently transformed input data. This approach facilitates the learning of more generalized and robust representations by the full network. Besides learning methods, designing good network architecture is also critical to learn good representations. Neural architecture search (NAS) has shown great potential in designing novel network structures, but its high computational cost is a significant limitation. To address this issue, we present a new short-training based NAS method that achieves superior performance compared to previous methods, while requiring significantly less search cost. Finally, with the recent advancements in large-scale image foundation models, we present an efficient fine-tuning method to adapt pre-trained image foundation models for video understanding. Our method significantly reduces training costs compared to traditional full fine-tuning, while delivering competitive performance across multiple video benchmarks. It is both simple and versatile, making it easy to leverage stronger image foundation models in the future.

EXTENDED ABSTRACT

Deep learning has demonstrated promising performance over various visual perception tasks including image classification, object detection, semantic segmentation and action recognition. However, deep neural networks are usually over-parameterized. They need to be trained on large-scale dataset and the training can take hundreds of GPU hours. During inference, large neural networks are also hard to be deployed on edge devices such as mobile phones and drones due to large memory cost and high computation complexity.

In this dissertation, we improve the efficiency of deep learning methods from multiple perspectives such as training efficiency, inference efficiency and data efficiency. Besides improving the model efficiency, our proposed methods also help deep neural networks learn better representations, which translates to better performance on various downstream tasks.

In Chapter 3, we propose a new method to train an adaptive neural network that can run at different computation complexities during inference time. We highlight the importance of simultaneously considering network width and input resolution for efficient network design. The proposed method mutually learns from different network widths and input resolutions and enables one model to meet different resource budgets during inference. Our method outperforms traditionally neural networks on various tasks under different model complexities. It also bears the benefits of training and deploying only one model.

In Chapter 4, we further extend the method in Chapter 3 to video understanding. Video understanding requires both spatial modeling and temporal modeling. Previous works proposed to process spatial and temporal dimensions asymmetrically for better performance and efficiency. Accordingly, we asymmetrically sample subnetworks, input resolutions and frames to do mutual training. After training, the adaptive network can run at different widths, resolutions and number of frames.

We demonstrate its effectiveness and efficiency on multiple video understanding tasks including video recognition and action detection.

Based on the results in Chapter 3 and 4, we find that the proposed method not only enables the network to execute at adaptive complexity, but also facilitates more effective representation learning. In light of this, in Chapter 5, we propose a new representation learning method for deep neural networks. Our idea is that a well-generalized network should provide consistent predictions for the same image with different augmentations, both for its sub-networks and for the network as a whole. Our method samples different sub-networks during training, feeds them with differently augmented samples, and pulls close their predictions. Our method demonstrates better performance than other state-of-the-art regularization and data augmentation methods on various network backbones and tasks. The improvement is more significant when labeled data is limited.

Besides learning methods, designing good network architectures is also critical to learn good representations. Neural architecture search (NAS) has demonstrated promising performance in designing new network structures, but its computationally cost is prohibitively high. Recent works have proposed training-free NAS metrics to accelerate the search process. However, in Chapter 6, we show that recent training-free NAS metrics are not fairly evaluated. Their performance is no better than the trivial number-of-parameter metric while being much more complicated to compute. Based on our observations, we proposed a new efficient training-based NAS method which outperforms previous methods with significantly smaller search cost. Our method is also more robust to different search spaces.

In Chapter 7, we propose an efficient finetuning methods to adapt recent large-scale image foundation models to video understanding. The traditional "pre-training then finetuning" paradigm is computationally expensive for video models, especially if we want to leverage large-scale image foundation models. In contrast, our proposed method freezes the pre-trained image model and only

introduces few light-weight Adapters to tune the model. The proposed method largely saves the training cost of video models and achieves even better performance than traditional full finetuning. It also brings the benefit of data efficiency compared to full finetuning.

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my parents for their love, support, and encouragement throughout my journey in pursuing my Ph.D. Their guidance, understanding and constant belief in my abilities have been instrumental in shaping my academic path. I extend my heartfelt appreciation to my advisor, Dr. Chen Chen, for his invaluable guidance, expertise, and patience. His unwavering dedication to my research and his insightful feedback have played a pivotal role in shaping my work. His mentorship has not only expanded my knowledge but also fostered my growth as a researcher. I would also like to express my gratitude to my committee members, Dr. Mubarak Sha, Dr. Liqiang Wang, Dr. Mingjie Lin, for their invaluable feedback to my dissertation and research. Many thanks to my friends and colleagues. Your friendship, encouragement, and camaraderie have provided me with a sense of belonging and motivation. The countless discussions, brainstorming sessions, and shared experiences have enriched my research and made this journey an enjoyable one. I am grateful for the laughter, support, and friendship we have shared. Finally, thanks to the difficulties I encountered along the way. They have taught me perseverance, resilience, and the importance of pushing through adversity

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

Learning good representations from data is critical for image and video understanding. Since AlexNet's tremendous success on the ImageNet 2012 Challenge, researchers have shifted attention from hand-crafted features to deep representation learning, where deep neural networks are trained to learn representations from data automatically. A series of works have been proposed to improve the power of deep neural networks by improving architectural designs, such as multi-branch structure, stacking more layers, skip connection, dense connection, etc. New learning and optimization methods are also proposed to train networks better. Although deep neural networks have achieved state-of-the-art performance on various visual understanding tasks, their computational cost is prohibitively high, which makes them hard to train and deploy. First, deep neural networks are often over-parameterized, requiring billions of multiply-add operations to forward one sample. This makes the training process expensive in terms of both GPU memory and time cost. Second, deep neural networks are prone to overfitting because of the large amount of model parameters. Therefore, they need to be trained on sufficiently large datasets. However, collecting a large-scale annotated dataset is a difficult task in many fields. Moreover, training on large-scale data will increase the time cost further. Third, because of the large memory and computational cost, it is hard to deploy deep neural networks on edge devices such as mobile phones.

To make the training and deployment of deep neural networks easier, in this dissertation, we improve the efficiency of deep representation learning from three different perspectives: training efficiency, inference efficiency, and data efficiency. By addressing these three aspects, we aim to make the process of deep representation learning more efficient and effective. Firstly, we identify the shortcomings of traditional deep neural networks and propose a new learning method to train adaptive networks. The proposed method enables us to conduct training one time while getting multiple well-performing networks at different complexities. Besides, when deploying the model

in real-life scenarios, we only need to use a single model to meet varying resource budgets. Secondly, we introduce a new regularization method to enhance the representation learning of deep neural networks. Our method significantly improves the performance and robustness of deep neural networks especially when annotated data is limited. Thirdly, we unveil the limitations of recent training-free neural architecture search (NAS) metrics, which heavily rely on the number of model parameters in ranking networks. To overcome this problem, we propose a new NAS metric which is independent of the number of model parameters while achieving better performance at lower search cost. Finally, we introduce a novel efficient finetuning method to adapt recent image foundation models to video understanding tasks. Our method only tunes a small portion of model parameters while achieving better performance than traditional full finetuning. The improvement is more significant when labeled data is scarce.

In the following sections, we first introduce the differences between traditional deep neural networks and adaptive neural networks. Then, we introduce popular regularization and data augmentation methods which aim to learn better representations from data. Next, we introduce neural architecture search, which is a new way of designing network structures. Finally, we introduce video understanding and parameter-efficient finetuning in both natural language processing and computer vision.

1.1 Adaptive Neural Network

Deep neural networks have triumphed over various perception tasks including image classification [64, 87, 70], object detection [107, 133], semantic segmentation [110, 19] and so on. However, most existing deep neural networks are static, which means they can only run at a specific resource constraint. For example, MobileNet [70] has 4.2M model parameters and 569M FLOPs. After training, the model can only do inference at this specific complexity. If we want to reduce the model

Table 1.1: Reducing the complexity of MobileNet by width or resolution at runtime. The model performance drops dramatically if without re-training.

Width	1.0×		0.75×		0.5×	
re-train	✓	✗	✓	✗	✓	✗
Acc (%)	70.6	70.6	68.4	14.2	63.3	0.4
Resolution	224×224		160×160		128×128	
re-train	✓	✗	✓	✗	✓	✗
Acc (%)	70.6	70.6	67.2	65.0	64.4	57.7

complexity, we have to re-train a smaller model; otherwise, the performance will drop substantially. As shown in Table 1.1, a regular MobileNet has 70.6% Top-1 accuracy on ImageNet. However, if we only use half of its channels ($width = 0.5\times$) to do inference, the performance drops to 0.4%. This is almost the same accuracy as a simple random guess. But if we re-train the MobileNet-0.5× from scratch, it can achieve 63.3% Top-1 accuracy [70]. A similar trend is observed if the network width is unchanged ($width = 1.0\times$) while the input image resolution is reduced. Although the performance does not drop as dramatically as by reducing the network width, the gap between the smaller resolution and full resolution is still quite large. These results indicate that regular deep neural networks can not generalize well to other network widths and image resolutions, and restrains their effectiveness to a specific resource budget.

In real-world applications, however, the computing capacity of different devices can vary significantly. A model may be small for a high-end GPU but too heavy to run on mobile devices. A common practice is to adopt a global width multiplier [70, 136, 203] to adjust the model size, but still, models of different scales need to be re-trained for many devices. Besides, even on the same device, the resource budgets can change. For example, the battery condition of mobile devices imposes constraints on the computational budget of many operations. Similarly, a task may have specific priorities at any given time, requiring a dynamic computational budget throughout

its deployment phases. To meet various resource constraints, one needs to deploy several different scaled networks on the device and switch among them. However, this naive solution is highly inefficient and not scalable. First, deploying several models will have a much higher memory footprint than a single model, and switching from one model to another is inefficient during runtime. Second, if the device needs to cover a new resource constraint, a new model has to be re-trained and deployed on the device. To address these issues, researchers have proposed the use of adaptive networks [191, 189, 186]. Adaptive networks allow a single model to operate at varying levels of complexity without requiring re-training.

1.2 Regularization and Data Augmentation

Deep neural networks are often over-parameterized and easily suffering from over-fitting. Regularization [142, 35] and data augmentation [87, 140] are widely used techniques to alleviate the over-fitting problem. Many data-level regularization methods [35, 200, 195] have achieved promising performance in image classification. These methods are similar to data augmentation where they put constraints on the input images. Although effective in image classification, these methods are hard to apply to downstream tasks such as object detection and segmentation due to their special operations. For example, the state-of-the-art CutMix [195] can not be directly applied to object detection because first, mixing samples will destroy the semantics in images; second, it is hard to interpolate the labels in these tasks. Another category of regularization methods imposes constraints on the network structures. [118] proposes that adding noises to the network gradients can improve generalization. Other methods [142, 75, 89] randomly drop some connections in the network, which implicitly introduce random noises in the training process. These methods are usually more generic but not as effective as data-level regularization.

In this dissertation, we introduce Gradient Augmentation (GradAug), which generates meaningful

disturbances to the gradients by the network itself rather than just adding random noises. The idea is that when a random transformation (e.g., random rotation, random scale, random crop, etc.) is applied to an image, a well-generalized network should still recognize the transformed image as the same object. Different from the regular data augmentation technique which only regularizes the full-network, we regularize the representations learned by a set of sub-networks, which are randomly sampled from the full network in terms of the network width (i.e., number of channels in each layer). Since the representation of the full network is composed of sub-networks' representations due to weights sharing during the training, we expect sub-networks to learn different representations from different transformations, which will lead to a well-generalized and diversified full network representation.

1.3 Neural Architecture Search

Neural Architecture Search (NAS) [211, 132, 104, 124, 170, 106, 212] is becoming an important technique in designing efficient and effective deep neural networks. Its effectiveness has been demonstrated in various computer vision tasks such as classification [124, 170, 212], object detection [29, 152] and semantic segmentation [21, 103]. Early NAS methods [211, 132, 150] leverage reinforcement learning or evolutionary algorithm to search networks. But this process is extremely expensive because they need to train thousands of candidate networks. Following works [106, 24, 181] alleviate this problem using differentiable search with candidate networks sampled from a supernet. During training, the network parameters and architecture parameters are optimized alternatively. However, training supernet can still be very slow and the accuracy of sub-networks in the supernet has a poor correlation with their ground truth accuracy [192]. To further reduce the search cost, training-free metrics [113, 22, 2] are proposed to rank the candidate networks without any training process. These metrics are largely inspired by the pruning methods

[91, 161, 153] and theoretical findings in deep neural networks [90, 177, 180, 117]. They aim to rank the networks from different aspects of the networks' properties such as trainability and expressivity. These metrics achieve competitive results with previous NAS methods at a much smaller search cost.

However, these works overlooked a straightforward training-free metric, the *number of parameters* (#Param) in a network, which is even faster to compute than those training-free metrics. In this dissertation, we show that #Param is surprisingly good on NAS-Bench-101 [188] and NAS-Bench-201 [37]. We further discover that these training-free metrics have a very high correlation with #Param, which indicates that a large portion of their ranking ability may come from the correlation with #Param. To validate our conjecture, we design systematic experiments to remove the impact of #Param. The results show that without the #Param information, recent training-free metrics [22, 113] cannot achieve a good performance.

Motivated by the above discovery, our objective is to develop a metric that has a weak correlation with #Param while still being effective so that it can provide additional information on estimating the performance of a network. Intuitively, a network's final performance is indicated by the structure (*e.g.*, #Param, #Layers), weight initialization, and the dynamics during training (*e.g.*, loss, gradients). We believe that metrics arise from the training dynamics should be weakly correlated with #Param. Angle metric is a training dynamic which is first proposed in [13] to indicate the network's generalization ability. It is defined as the angle between the vectorized network weights before and after training. We find that the angle metric at the final fully-connected (FC) layer has a high correlation with the accuracy but a low correlation with the number of parameters. This indicates that it can provide additional information other than #Param on estimating the network's performance. To reduce the computation for model training, we propose an extremely light-weight training scheme with a small proxy dataset which is thousands times faster than traditional training.

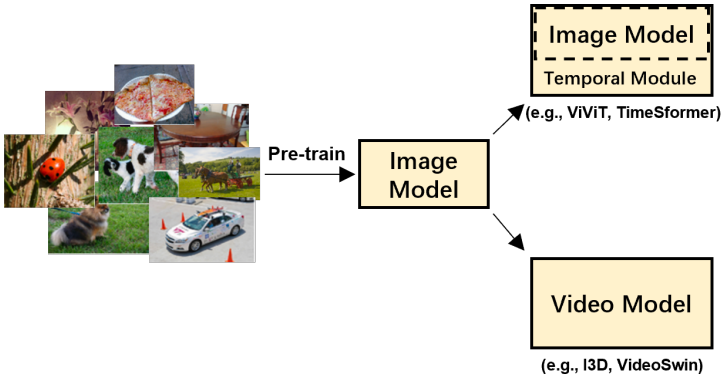


Figure 1.1: An illustration of the two popular directions in extending an image model to video model. One is to add additional temporal modules to the image model, the other is to inflate the image model to a 3D video model.

1.4 Video Understanding and Parameter-Efficient Finetuning

The "pre-training then finetuning" paradigm has played an important role in computer vision. The key to this paradigm is a well pre-trained image model, which can provide strong transferability to downstream tasks through finetuning. Recently, large foundation models [131, 193, 155, 77, 164] can even demonstrate remarkable few-/zero-shot performance given their learned superior visual representations.

In video understanding, a common practice is also bootstrapping from an image pre-trained model and then finetuning on the video data. There are two dominating directions as shown in Figure 1.1, one is to extend an image model with additional temporal module [98, 208, 3], the other is to inflate an image model to a video model [15, 109]. However, there exists at least two drawbacks for the aforementioned methods. First, most approaches require full finetuning (*i.e.*, updating all the model parameters during training) to achieve promising results on common video benchmarks. This is quite costly in terms of both computation and memory footprint, *e.g.*, 1200 Tesla V100 GPU hours to train [109]. Second, it also remains questionable that whether it is necessary to full

finetune pre-trained image models given that they have demonstrated excellent transferability. An inadequate finetuning on downstream data might destroy the well generalized representations from such foundation models.

To overcome the drawbacks, a research direction termed parameter-efficient transfer learning has been trending in natural language processing (NLP) [68, 92, 6, 71]. The goal is to only finetune a small number of (extra) parameters while keeping large pre-trained language models [34, 11] frozen to attain strong performance. With the rise of large vision transformer (ViT) models, such techniques have been recently introduced to computer vision for efficient transfer learning. However, existing works either focus on tuning a pre-trained image model for image tasks (image-to-image) [4, 80, 78], or tuning a pre-trained video model for video tasks (video-to-video) [20]. Directly leveraging pre-trained image models for efficient transfer learning to video tasks (image-to-video) is less explored, because image models lack the capability of temporal reasoning.

In this dissertation, we introduce a new way to **A**dapt pre-trained **I**mage transformer **M**odels (AIM) for efficient video action recognition. By freezing the pre-trained image model and adding a few lightweight adapters [68] during finetuning, we show that our proposed AIM can achieve competitive or even better results than previous state-of-the-art methods with substantially fewer tunable parameters. To be specific, we first introduce adapter after self-attention layer in a transformer block to perform *spatial adaptation*. We show that a well pre-trained image model is sufficiently good for spatial modeling in video understanding. Then for temporal modeling, we simply reuse the image pre-trained self-attention layer but apply it to the temporal dimension of video input, forcing it to model the relationship across different frames. An adapter is also appended for *temporal adaptation*. Finally, we perform *joint adaptation* by adding another adapter in parallel to the MLP layer in a transformer block.

1.5 Organization

In Chapter 2, we conduct a comprehensive literature review on adaptive neural networks, as well as related works in deep representation learning, neural architecture search, video understanding, and parameter-efficient finetuning. In Chapter 3, we propose a new learning method to train adaptive networks for various image understanding tasks. In Chapter 4, we further extend our proposed method to video understanding tasks. In Chapter 5, we introduce a novel deep representation learning method which improves the performance, robustness and data efficiency of modern deep neural networks. In Chapter 6, we revisit recent training-free neural architecture search metrics and unveil their shortcoming. Based on our discovery, we propose a new training-based neural architecture search metric. In Chapter 7, we propose a novel parameter-efficient finetuning method to adapt pre-trained image models for video understanding. In Chapter 8, we summarize the dissertation and highlight potential directions for future work.

CHAPTER 2: LITERATURE REVIEW

In this chapter, we provide a comprehensive literature review on works related to this dissertation. We begin by reviewing previous works on efficient neural networks and adaptive neural networks. Next, we examine popular representation learning methods including regularization and data augmentation. We then discuss the new progress made by neural architecture search methods and their limitations. Finally, we review recent image foundation models, video action recognition methods, and parameter-efficient finetuning techniques.

2.1 Efficient Networks and Adaptive Networks

There has recently been a flurry of interest in designing light-weight networks. MobileNets [70, 136] factorize the standard 3×3 convolution into a 3×3 depthwise convolution and a 1×1 pointwise convolution which reduces computation cost by several times. ShuffleNets [203, 112] separate the 1×1 convolution into group convolutions and shuffle the groups to further improve accuracy-efficiency trade-offs. ShiftNet [171] introduces a zero-flop shift operation to reduce computation cost. AdderNet [16] trades the massive multiplications in deep neural networks for much cheaper additions. GhostNet [57] leverages cheap linear transformations to generate more ghost feature maps. Most recent works [170, 150, 69] also apply neural architecture search methods to search efficient network structures. However, none of them consider the varying resource constraints during runtime in real-world applications. To meet different resource budgets, these methods need to train and deploy several models and switch among them, which is inefficient and not scalable.

Following a similar idea of 2D convolutional neural networks, spatiotemporal (3D) networks are proposed to handle video data. The works reported in[157, 15, 58] build 3D networks by extending

2D convolutional filters [87, 141, 147, 64] to 3D filters along the temporal axis; then the 3D filters are used to learn spatio-temporal representations in a similar way to their 2D counterparts. Later works [178, 129, 159, 45] propose to treat the spatial and temporal domains differently. The authors in [178] posit that a bottom-heavy structure is better than naive 3D structures in both accuracy and speed. In [129, 159], 3D filters are split into 2D+1D filters, which reduce the heavy computational cost of 3D filters and improve the performance. SlowFast [45] further shows that space and time should not be handled symmetrically and introduces a two-path structure to deal with slow and fast motion separately. Several works [28, 111, 158] also leverage the group convolution and channel-wise separable convolution in 2D networks to reduce computational cost. Recently, [125, 167, 123] explore neural architecture search (NAS) techniques to automatically learn spatio-temporal network structures. However, all these structures are static. We are the first to achieve adaptive 3D networks which also outperform state-of-the-art independently-trained models [45, 43].

In light of this, adaptive networks are proposed which can run at different complexities without re-training. One category of methods [74, 185, 168, 114] perform dynamic inference conditioned on the input. The core principle of these methods is to utilize less computation for easy samples and reserve more computation for hard ones. MSDNet [74] proposes a multi-scale and coarse-to-fine densenet framework. It has multiple classifiers and can make early predictions for easy instances. RANet [185] motivates its design with the idea that low-resolution images are enough for classifying easy samples, while only hard samples need high-resolution input. GFNet [168] processes a sequence of small patches from the original images and terminates inference once the model is sufficiently confident about its prediction. There are also many recent works [114, 176, 115, 122, 175] aiming to reduce spatial and temporal redundancies in videos for action recognition by dynamically processing input frames and fusing feature maps. Inspired by SENet [72], a series of works [184, 26, 27] also propose to learn dynamic attention for different samples. Our method is closer to another category of methods [191, 189, 84, 12, 190] where the dynamic routes are determined

by the resource budgets. NestedNet [84] uses a nested sparse network consisting of multiple levels to meet various resource requirements. SlimmableNet [191, 189] proposes to train several sub-networks together and perform inference at different network widths. However, it fails to achieve a strong accuracy-efficiency trade-off since it ignores the input dimension. Later works [12, 190] further integrate other dimensions (e.g., depth and kernel size) into the training framework, but they do so with neural architecture search (NAS) and thereby require a very complex and expensive training process. Furthermore, their effectiveness is only evaluated on image classification, while our method is thoroughly evaluated on image classification, detection, segmentation and action recognition.

2.2 Training Regularization and Data Augmentation

Data augmentation [87, 140, 31] increases the amount and diversity of training data by linear or non-linear transformations over the original data. In computer vision, it usually includes rotation, flipping, etc. Recently, a series of regularization methods use specially-designed operations on the input images to alleviate over-fitting in deep neural networks. These methods are similar to data augmentation. Cutout [35] randomly masks out a squared region on the image to force the network to look at other image context. Dropblock [49] shares a similar idea with Cutout but it drops a region in the feature maps. Although they have achieved improvements over the regular data augmentation, such region dropout operations may lose information about the original images. Mixup [200] mixes two samples by linearly interpolating both the images and labels. CutMix [195] combines Cutout and Mixup to replace a squared region with a patch from another image. Other mixed sample variants [143, 148] all share similar ideas. While effective in image classification, the mixed sample augmentation is not natural to be applied to tasks such as detection and segmentation due to semantic and label ambiguities. In contrast, the proposed GradAug

is a task-agnostic approach which leverages the most common image transformations to regularize sub-networks. This allows the method to be directly applied to different vision tasks and easily amenable for other applications.

Another category of regularization methods imposes constraints on the network weights and structure to reduce over-fitting. [118] points out that adding random noises to the gradients during training can help the network generalize better. Dropout [142] randomly drops some connections during training to prevent units from co-adapting. The random dropping operation also implicitly introduces random noises into the training process. Many following works share the idea of Dropout by randomly dropping network layers or branches. Shake-Shake [48] assigns random weights to residual branches to disturb the forward and backward passes. But it is limited to three-branch architectures. ShakeDrop [182] extends Shake-Shake to two-branch architectures (e.g., ResNet [64] and PyramidNet [56]). However, its application is still limited. [75] randomly drops a subset of layers during training. The final network can be viewed as an ensemble of many shallow networks. Although these methods have shown improvements on image classification, they are usually not as effective as data-level regularization strategies. Moreover, their generalization and effectiveness are not validated on other tasks.

GradAug leverages the advantages of both categories of methods. It uses different augmentations to regularize a set of sub-networks generated from the full network in the joint training process. This introduces self-guided disturbances to the gradients of the full network rather than adding random noises. The method is more effective and generic than previous techniques.

2.3 Neural Achitecture Search

NAS is proposed to search network structures automatically for a given task instead of time-consuming manual design. Early works [211, 132, 150, 105] leverage reinforcement learning or evolutionary algorithms to explore architectures. The controller will generate some networks and the network performance will be used as feedback information to update the controller. However, training a large amount of networks is very expensive, costing thousands of GPU days. Following works accelerate NAS algorithms by weight-sharing in a supernet. ENAS [124] proposes to share the weights among candidate networks so that they can be trained simultaneously. DARTS [106] concatenates all candidate operations into a supernet and each operation is assigned an architecture parameter denoting its importance. During training, the architecture parameters and weight parameters are optimized alternatively. Another kind of weight-sharing method is one-shot NAS [7, 10, 55], where a supernet is trained with sub-networks stochastically sampled in each iteration. However, recent study [192] shows that the network performance via weight-sharing has a poor correlation with its actual performance.

To further speedup the search process, recent works [113, 22, 2] propose to predict network performance without training. [2] evaluates the effectivenss of different pruning-at-initialization criteria [161, 153, 91] for NAS. NASWOT [113] leverages the number of linear regions [180] to rank different networks. TE-NAS [22] further combines linear regions with neural tangent kernel (NTK) [90] to rank a network by its expressivity and trainability. However, [116] shows that NTK-based metrics are unstable across different search spaces and initializations. In this dissertation, we further reveal that the effectiveness of training-free metrics (Linear Region and NTK) mainly come from the high correlation with #Param, and #Param happens to be a good metric on the evaluated benchmarks.

2.4 Image Pre-trained Foundation Models

ViT [39] and its variants [108, 165, 194, 36] have been proposed to achieve state-of-the-art performance on image recognition. Once trained, these models could also serve as good initialization for transfer learning to downstream tasks. In terms of training techniques, they are commonly trained on large-scale labeled datasests [33, 144, 198] in a supervised manner. To alleviate the labeling cost, self-supervised learning methods [25, 5, 207, 60, 179] are introduced to learn effective representations from unlabeled data. Recent works [131, 77, 193, 164] adopt large-scale multimodal data (*e.g.*, image-text pairs) for model training, which leads to even more powerful visual representations. In this work, thanks to the simplicity of our proposed method, we could take advantage of these well pre-trained image models and adapt them efficiently to solve video tasks.

2.5 Video Action Recognition

A paradigm shift from using convolutional networks [15, 160, 187, 98, 46] to transformers has been observed for video action recognition. Most works use image pre-trained models as initialization and extend them to video models by introducing new temporal modules [8, 3, 204, 183] or inflating them to video models [109]. Another direction is to directly pre-train a video model in a self-supervised manner [88, 44, 210, 149]. However, all these models are full finetuned on video data, which makes the training cost unaffordable to most researchers and practitioners. There are some recent works [119, 81, 173, 174] extending CLIP to perform action recognition, but they are multimodal methods which requires additional text branch. Our proposed AIM leverages existing pre-trained image models (no need for video model pre-training), only tunes a small number of model parameters (much more efficient than full finetuning), and achieves comparable or even better performance than previous state-of-the-arts.

2.6 Parameter-efficient Finetuning

Parameter-efficient finetuning techniques [68, 71, 92, 95, 59, 6, 146, 128] are first proposed in NLP since full finetuning the increasingly larger language models for various downstream tasks becomes less feasible. Their goal is to reduce the number of trainable parameters thus lowering the computation cost, while reaching or surpassing the performance of full finetuning. Recently, parameter-efficient transfer learning is also studied in computer vision [78, 4, 20, 80, 47]. All these methods focus on adapting models in the same domain (*e.g.*, image-to-image or video-to-video), while our method adapts an image model for video tasks. One concurrent work [102] also studies how to adapt image pre-trained models for video action recognition. However, there are several major differences. First, they add new trainable decoder branches, which consist of 3D convolutions and cross-frame attention, to the frozen image encoder. We simply reuse image pre-trained self-attention to perform temporal modeling, while enjoying better performance and less tunable parameters. Second, our method is shown to be compatible with different image models, while [102] only shows its effectiveness on CLIP image encoder.

CHAPTER 3: ADAPTIVE NEURAL NETWORK FOR IMAGE UNDERSTANDING

The work in this Chapter has been published in the following paper:

Taojiannan Yang, Sijie Zhu, Chen Chen, Shen Yan, Mi Zhang, Andrew Willis. "MutualNet: Adaptive ConvNet via Mutual Learning from Network Width and Resolution." European Conference on Computer Vision. 2020.

Standard 2D models are trained at a fixed width-spatial configuration (e.g., 1.0×-224 on ImageNet). However, the model does not generalize well to other configurations during inference as shown in Table 1.1. In our method, we randomly sample different width-spatial configurations during training, so the model can run effectively at various configurations during inference. Note that the computation cost of a vanilla 2D convolutional layer is given by

$$K \times K \times C_i \times C_o \times H \times W. \tag{3.1}$$

Here, K denotes the kernel size, and C_i and C_o respectively denote the input and output channels of this layer, while H and W respectively denote the height and width of the output feature map. For a smaller model configuration, e.g. 0.5×-160, the width is reduced by $\gamma_w = 0.5$ and the spatial resolution is reduced by $\gamma_s = 160/224 = 0.7$. The computation cost is reduced to

$$K \times K \times \gamma_w C_i \times \gamma_w C_o \times \gamma_s H \times \gamma_s W, \tag{3.2}$$

which is $\rho = \gamma_w^2 \gamma_s^2$ times that of the original in Eq. 3.1. The dynamic execution range of MutualNet

is determined by the range of γ_w and γ_s. The detailed settings of γ_w, γ_s and ρ will be discussed in the following sections.

This Chapter is organized as follows: in Section 3.1, we first show that distinct model configurations focus on different semantic information in an image. Then, we introduce the training process of our method by a concrete example. Section 3.3 explains the working mechanism of mutual learning from the perspective of model gradients. Section 3.4 introduces how to deploy the model and do inference at different resource budgets.

3.1 Knowledge in Different Model Configurations

We want to randomly sample different model configurations in each training iteration to allow them to learn from each other. However, is there any unique knowledge in different model configurations that is beneficial for transferring to others? The answer is yes. Fig. 3.1 shows the classification activation maps (CAM) [206] of two model configurations. The models are trained independently at the corresponding configuration. We can see that these two models focus on different semantic regions of the same object. The larger model configuration (i.e., 1.0×-224) tends to focus on fine details (e.g., face of the dog) while the smaller configuration learns the global structures (e.g., the whole body). This can be partially attributed to the downsampling of the input resolution, where

Table 3.1: Weakly-supervised localization accuracy of different model configurations on ImageNet validation set using CAM [206]. The backbone network is ResNet-50.

Model Config	All	Large	Small
1.0×-224	37.9%	48.3%	8.7%
1.0×-160	25.2%	28.3%	12.9%
0.75×-128	24.8%	31.2%	8.3%

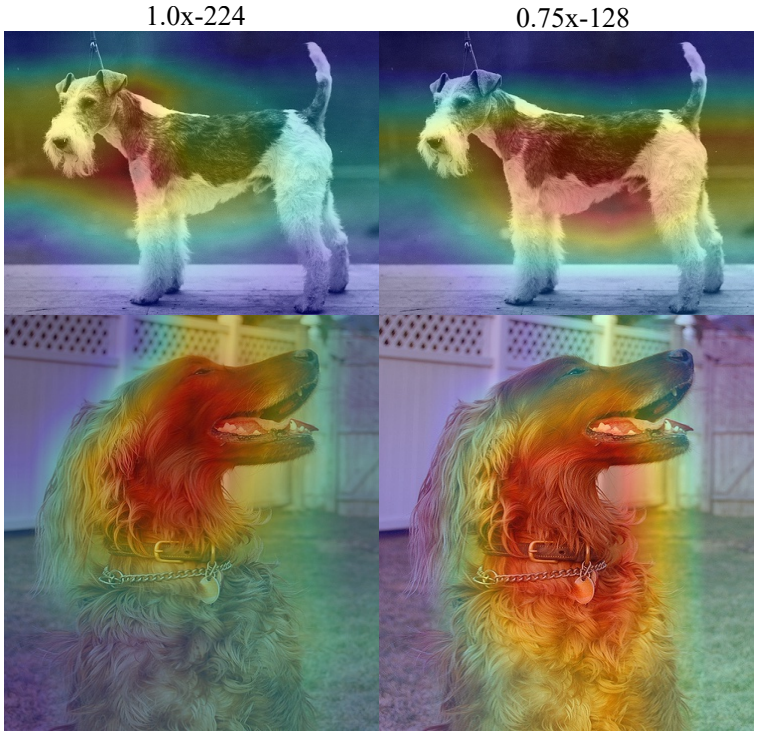


Figure 3.1: Class activation maps (CAM) of different model configurations (the network is ResNet-50 and is trained on ImageNet). Larger model configuration focuses more on details (e.g., face) while the smaller one focuses more on the contour (e.g., body).

some fine-grained information is lost but the object contour is enhanced. To further demonstrate that different model configurations have varied attention, we leverage their attention maps to conduct weakly-supervised object localization using CAM [206] on the ImageNet validation set and compare their localization accuracy on large and small objects. We define small objects as those with a ground truth bounding box smaller than 20% of the image size, and large objects as those with a ground truth bounding box larger than 50% of the image size. A prediction is considered correct if its IoU with the ground truth bounding box is larger than 0.5. The results are shown in Table 3.1. We can see that 1.0×-224 achieves the highest accuracy on all objects and large objects, while 1.0×-160 performs better on small objects. Also, 0.75×-128 has lower performance on small objects but higher performance on large objects compared to 1.0×-160. The results show

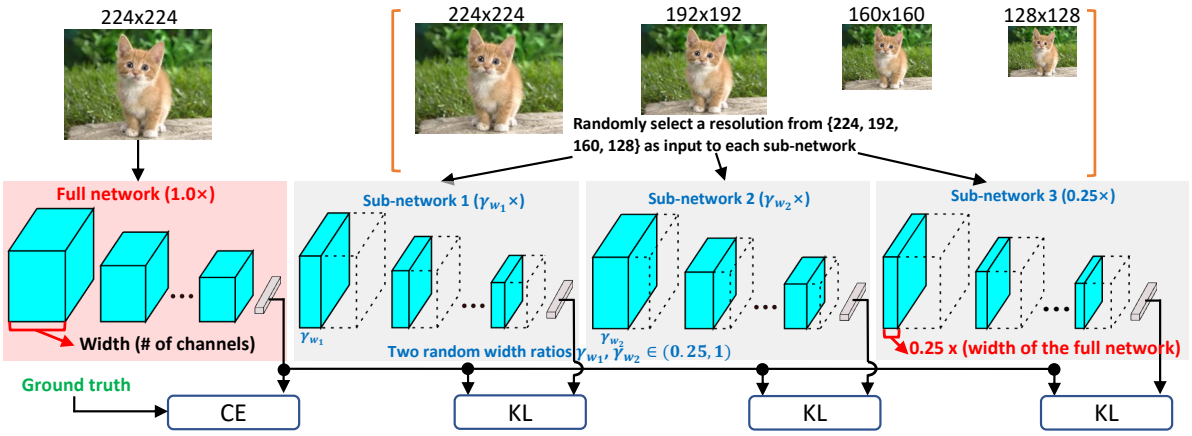


Figure 3.2: An example to illustrate the training process of MutualNet. The network width range is [0.25×, 1.0×], input resolution is chosen from {224, 192, 160, 128}. This can achieve a computation range of [13, 569] MFLOPs on MobileNet v1 backbone. We follow [189] to sample 4 networks, i.e., upper-bound full width network (1.0×), lower-bound width network (0.25×), and **two random width ratios** $\gamma_{w_1}, \gamma_{w_2} \in (0.25, 1)$. For the full-network, we constantly choose 224×224 resolution. For the other three sub-networks, we randomly select its input resolution. The full-network is optimized with the ground-truth label using Cross Entropy loss (CE). Sub-networks are supervised by the prediction of the full-network using Kullback–Leibler Divergence loss (KL). **Weights are shared among different networks to facilitate mutual learning**.

that different configurations focus on different semantic regions. Inspired by this observation, we leverage large configurations to supervise small ones during training. This further enhances the knowledge transfer among different configurations and helps the model to learn more diversified representations.

3.2 MutualNet Training

We present an example to illustrate our training process in Fig. 3.2. We set the adaptive width range as [0.25×, 1.0×], and the adaptive resolutions as {224, 192, 160, 128}. Note that one can adjust these settings according to the required dynamic resource budgets. The depth and resolution can even be larger than the default setting (1.0×-224) to scale up the model as we show in

Section 7.3. As shown in Fig. 3.2, we first follow [189] to sample four sub-networks, i.e., the smallest ($0.25\times$), the largest ($1.0\times$) and *two random width ratios* $\gamma_{w_1}, \gamma_{w_2} \in (0.25, 1)$. Then, unlike traditional ImageNet training with 224×224 input, we resize the input images to resolutions randomly chosen from {224, 196, 160, 128} and feed them into different sub-networks. We denote the weights of a sub-network as $W_{0:w}$, where $w \in (0, 1]$ is the width of the sub-network and $0 : w$ means the sub-network adopts the first $w \times 100\%$ weights of each layer of the full network. $I_{R=r}$ represents a $r \times r$ input image. Then $N(W_{0:w}, I_{R=r})$ represents the output of a sub-network with width w and input resolution $r \times r$. For the largest sub-network (i.e., the full-network in Fig. 3.2), we always train it with the highest resolution (224×224) and ground truth label y. The loss for the full network is

$$loss_{full} = CrossEntropy(N(W_{0:1}, I_{R=224}), y). \tag{3.3}$$

For the other sub-networks, we randomly pick an input resolution from {224, 196, 160, 128} and supervise it with the output of the full-network. As demonstrated in Section 3.1, this can transfer the unique knowledge in the full configuration to other configurations and benefit the overall performance. The loss for the i-th sub-network is

$$loss_{sub_i} = KLDiv(N(W_{0:w_i}, I_{R=r_i}), N(W_{0:1}, I_{R=224})), \tag{3.4}$$

where *KLDiv* is the Kullback-Leibler divergence which measures the distance between two distributions. The total loss is the summation of the full-network and sub-networks, i.e.,

$$loss = loss_{full} + \sum_{i=1}^{3} loss_{sub_i}. \tag{3.5}$$

The reason for training the full-network with the highest resolution is that the highest resolution contains more details. Also, the full-network has the strongest learning ability to capture the dis-

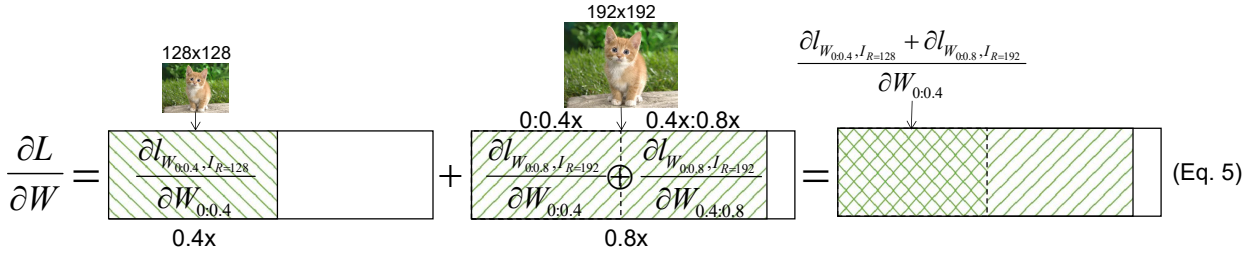


Figure 3.3: An illustration of the mutual learning scheme. It allows the sub-network to learn multi-scale representations, in terms of both network width and input image resolution.

criminatory information from the image data.

3.3 Gradient Analysis of Mutual Learning

To better understand why the proposed framework can mutually learn from different widths and resolutions, we perform a gradient analysis of the mutual learning process. For ease of demonstration, we only consider two network widths $0.4\times$ and $0.8\times$, and two resolutions 128 and 192 in this example. As shown in Fig. 3.3, sub-network $0.4\times$ selects input resolution 128, sub-network $0.8\times$ selects input resolution 192. Then we can define the gradients for sub-network $0.4\times$ and $0.8\times$ as $\frac{\partial l_{W_{0:0.4},I_R=128}}{\partial W_{0:0.4}}$ and $\frac{\partial l_{W_{0:0.8},I_R=192}}{\partial W_{0:0.8}}$, respectively. Since sub-network $0.8\times$ shares weights with $0.4\times$, we can decompose its gradient as

$$\frac{\partial l_{W_{0:0.8},I_R=192}}{\partial W_{0:0.8}} = \frac{\partial l_{W_{0:0.8},I_R=192}}{\partial W_{0:0.4}} \oplus \frac{\partial l_{W_{0:0.8},I_R=192}}{\partial W_{0.4:0.8}}, \tag{3.6}$$

where $\oplus$ is vector concatenation. Since the gradients of the two sub-networks are accumulated during training, the total gradients are computed as

$$\begin{aligned}\frac{\partial L}{\partial W} &= \frac{\partial l_{W_{0:0.4},I_R=128}}{\partial W_{0:0.4}} + \frac{\partial l_{W_{0:0.8},I_R=192}}{\partial W_{0:0.8}} \\ &= \frac{\partial l_{W_{0:0.4},I_R=128}}{\partial W_{0:0.4}} + \left(\frac{\partial l_{W_{0:0.8},I_R=192}}{\partial W_{0:0.4}} \oplus \frac{\partial l_{W_{0:0.8},I_R=192}}{\partial W_{0.4:0.8}} \right) \\ &= \frac{\partial l_{W_{0:0.4},I_R=128} + \partial l_{W_{0:0.8},I_R=192}}{\partial W_{0:0.4}} \oplus \frac{\partial l_{W_{0:0.8},I_R=192}}{\partial W_{0.4:0.8}}\end{aligned} \tag{3.7}$$

Therefore, the gradient for sub-network $0.4\times$ is $\frac{\partial l_{W_{0:0.4},I_R=128} + \partial l_{W_{0:0.8},I_R=192}}{\partial W_{0:0.4}}$, which consists of two parts. The first part is computed by itself ($0 : 0.4\times$) with 128×128 input resolution. The second part is computed by a larger sub-network $0.8\times$ (i.e., $0 : 0.4\times$ portion) with 192×192 input resolution. Thus the sub-network is able to capture multi-scale representations from different input scales and network scales. Due to the random sampling of network width, every sub-network is able to learn multi-scale representations in our framework. This allows the model to significantly outperform even independently-trained networks. Note that this is different from multi-scale data augmentation as explained in Section 7.3.

3.4 MutualNet Inference

After training, the model is able to run at various width-resolution configurations. To deploy the model, we need to find the best-performed model configuration under each particular resource constraint. For evaluation, after following the training example in Section 3.2, we first sample the network widths from $0.25\times$ to $1.0\times$ with a step-size of $0.05\times$. Then we sample the input resolutions from {224, 192, 160, 128}. We evaluate all these width-resolution configurations on a validation set which gives us a configuration-accuracy table. Similarly, we can evaluate the computational cost (e.g., FLOPs) of each model configuration which gives us a configuration-complexity table. Note that different model configurations may have the same computational cost (e.g., on

MobileNet v1, the computational cost of 0.6×-224 and 0.7×-192 are both ∼210 MFLOPs). So the final step is to find the best-performing model configuration for each resource budget, from which we can get the complexity-configuration query table. For real deployment, we only need to deploy the MutualNet model (which is of the same size as a regular model) and the query table. Then given a resource constraint, we can look up the query table and inference the model at the corresponding optimal configuration. Note that the feature statistics (mean and variance) are different across different model configurations, so we can not use one set of batch normalization (BN) statistics for all configurations. We follow [189] to perform BN statistics calibration for each model configuration. Before evaluation, we forward several batches of data to update the BN statistics for a specific configuration. **There is no re-training so that the whole process is fast and only needs to be done once**.

3.5 Experiments

We conduct extensive experiments to evaluate the effectiveness of MutualNet. We first present our results on ImageNet [32] classification to illustrate the effectiveness of MutualNet. Next, we conduct extensive ablation studies to analyze the mutual learning scheme. Finally, we apply MutualNet to transfer learning datasets and COCO [101] object detection and instance segmentation to demonstrate its robustness and generalization ability.

3.5.1 Evaluation on ImageNet Classification

We compare MutualNet with SlimmableNet (S-Net [191] and US-Net [189]) and independently-trained networks on the ImageNet dataset. We evaluate our framework on three popular network structures, MobileNetv1 [70], MobileNetv2 [136] and ResNet-50 [64].

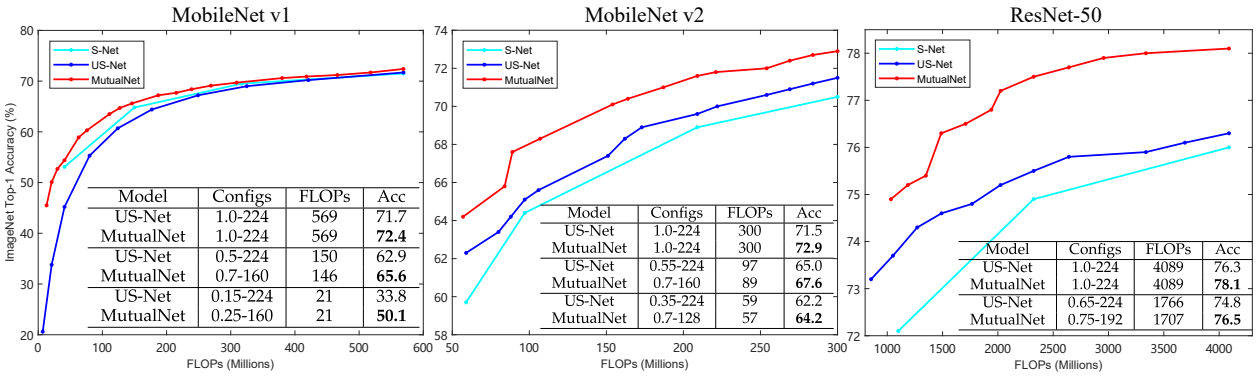


Figure 3.4: Comparisons of Accuracy-FLOPs curves of MutualNet, S-Net and US-Net. In the tables, we compare some points on the curves by their configurations, the corresponding FLOPs and accuracy. MutualNet consistently outperforms S-Net and US-Net at different model complexities.

Implementation Details. We follow the settings in SlimmableNet and make the comparison under the same dynamic FLOPs constraints: [13, 569] MFLOPs on MobileNetv1, [57, 300] MFLOPs on MobileNetv2 and [660, 4100] MFLOPs on ResNet-50. The input image resolution is randomly picked from {224, 192, 160, 128} unless specified. We use width scale [0.25, 1.0]$\times$ on MobileNetv1, [0.7, 1.0]$\times$ on MobileNetv2 and [0.7, 1.0]$\times$ on ResNet-50. The width lower bound is slightly higher than that in SlimmableNet because we perform multi-dimension trade-off during training. The other training settings are the same as SlimmableNet.

Comparison with SlimmableNet. The Accuracy-FLOPs curves are shown in Fig. 3.4. We can see that our method consistently outperforms S-Net and US-Net on MobileNetv1, MobileNetv2 and ResNet-50 backbones. Specifically, we achieve significant improvements under small computation costs. This is because our framework considers both network width and input resolution and can find a better balance between them. For example, on MobileNet v1 backbone, if the resource constraint is 150 MFLOPs, US-Net has to reduce the width to 0.5$\times$ given its constant input resolution 224, while MutualNet can meet this budget by a balanced configuration of (0.7$\times$ - 160), leading to a better accuracy (65.6% (Ours) vs. 62.9% (US-Net) as listed in the table of Fig. 3.4).

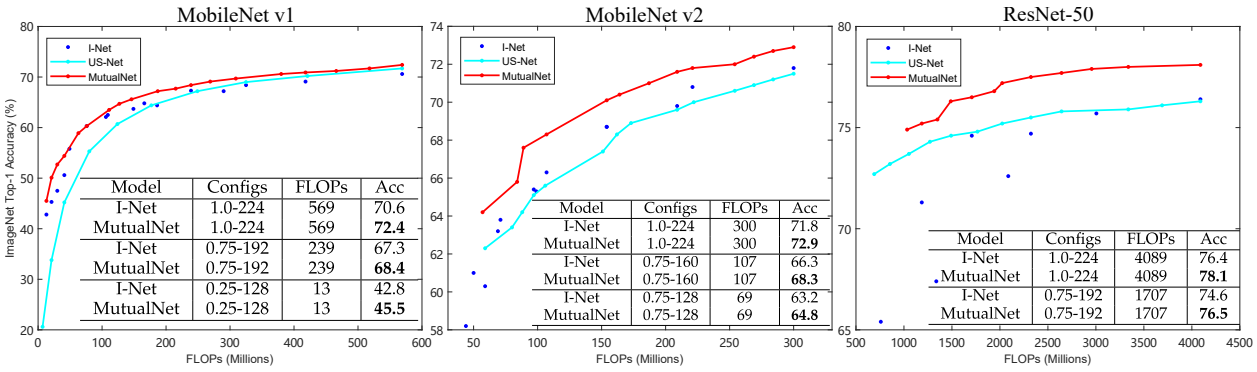


Figure 3.5: Comparisons of Accuracy-FLOPs curves of MutualNet, US-Net and I-Net. MutualNet consistently outperforms US-Net and I-Net at different model complexities.

Also, our framework is able to learn multi-scale representations as demonstrated in Section 3.3, which further boost the performance of each sub-network. We can see that even for the same configuration (e.g., 1.0×-224) our approach clearly outperforms US-Net, i.e., 72.4% (Ours) vs. 71.7% (US-Net) on MobileNet v1, 72.9% (Ours) vs. 71.5% (US-Net) on MobileNet v2, and 78.1% (Ours) vs. 76.3% (US-Net) on ResNet-50. (Fig. 3.4).

Comparison with Independently Trained Networks. We compare the performance of MutualNet and US-Net with independently-trained networks (**denoted by I-Net**) under different width-resolution configurations in Fig. 3.5. In I-Net, the resolutions are selected from {224, 192, 160, 128}. Width are selected from {1.0×, 0.75×, 0.5×, 0.25×} on MobileNet v1&v2 and {1.0× 0.75×} on ResNet-50. From Fig. 3.5 we can see that US-Net only achieves comparable (in many cases worse) performance compared to I-Net, while MutualNet consistently outperforms US-Net and I-Net on three backbones. Even at the same width-resolution configuration, which may not be the best configuration, MutualNet can achieve much better performance than I-Net. This demonstrates that MutualNet not only finds the better width-resolution balance but also learns stronger representations by the mutual learning scheme.

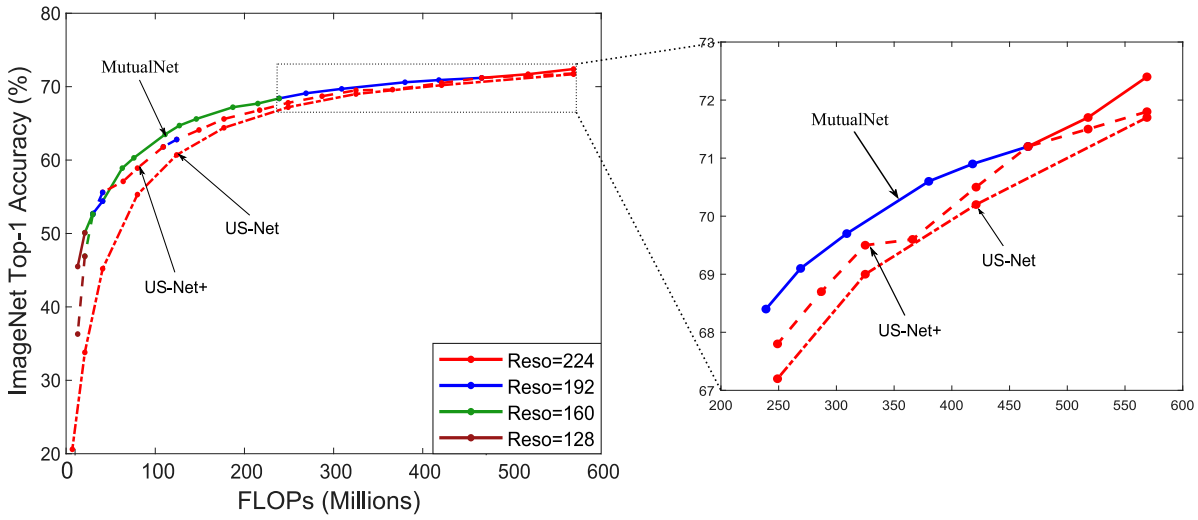


Figure 3.6: The width-resolution trade-offs at different resource constraints. The Accuracy-FLOPs curves are based on MobileNet v1 backbone. We highlight the selected resolution under different FLOPs with different colors. For example, the solid green line indicates that when the constraint range is [41, 215] MFLOPs, our method constantly selects input resolution 160 but reduces the width to meet the resource constraint. Best viewed in color.

Balanced Width-Resolution Configuration via Mutual Learning. One may apply different resolutions to US-Net *during inference* to yield improvement over the original US-Net. However, this way the optimal width-resolution balance can be achieved due to lack of width-resolution mutual learning. In one experiment, we evaluate US-Net at width scale $[0.25, 1.0]\times$ with input resolutions $\{224, 192, 160, 128\}$ and denote this improved model as **US-Net+**. In Fig. 3.6, we plot the Accuracy-FLOPs curves of our method and US-Net+ based on MobileNet v1 backbone, and highlight the selected input resolutions with different colors. As we decrease the FLOPs ($569 \rightarrow 468$ MFLOPs), MutualNet first reduces network width to meet the constraint while keeping the 224×224 resolution (red line in Fig. 3.6). After 468 MFLOPs, MutualNet selects lower input resolution (192) and then continues reducing the width to meet the constraint. On the other hand, US-Net+ cannot find such balance. It always slims the network width and uses the same (224) resolution as the FLOPs decreasing until it goes to really low. This is because US-Net+ does

Table 3.2: Comparison with EfficienNet to scale up MobileNetv1 by $\times 4$ on ImageNet. d: depth, w: width, r: resolution.

Model	Best Model	FLOPs	Top-1 Acc
EfficientNet [151]	$d = 1.4, w = 1.2, r = 1.3$	2.3B	75.6%
MutualNet	$w = 1.6, r = 1.3$	2.3B	**77.1%**

not incorporate input resolution into the learning framework. *Simply applying different resolutions during inference cannot achieve the optimal width-resolution balance.*

Comparison with EfficientNet. EfficientNet [151] acknowledges the importance of balancing among network width, depth and resolution. But they are considered as independent factors. The authors use grid search over these three dimensions and train each model configuration independently to find the optimal one under certain constraint, while MutualNet incorporates width and resolution in a unified training framework. To show the benefits of the mutual learning shceme, we compare MutualNet with the best model scaling that EfficientNet finds for MobileNet v1 at 2.3 BFLOPs (scale up baseline by $\times 4.0$). To cover this model scale we scale up MutualNet by using a width range of $[1.0\times, 2.0\times]$, and select resolutions from {224, 256, 288, 320}. This makes MutualNet executable in the range of [0.57, 4.5] BFLOPs. We pick the best performing width-resolution configuration at 2.3 BFLOPs. The results are compared in Table 3.2. Although EfficientNet claims to find the optimal scaling compound, its performance is much worse than MutualNet. This is because EfficientNet fails to leverage the information in other configurations, while MutualNet captures multi-scale representations for each model configuration thanks to the width-resolution mutual learning.

Comparison with Multi-scale Data Augmentation. In multi-scale data augmentation, the network may take images of different resolutions in different training iterations. But within each

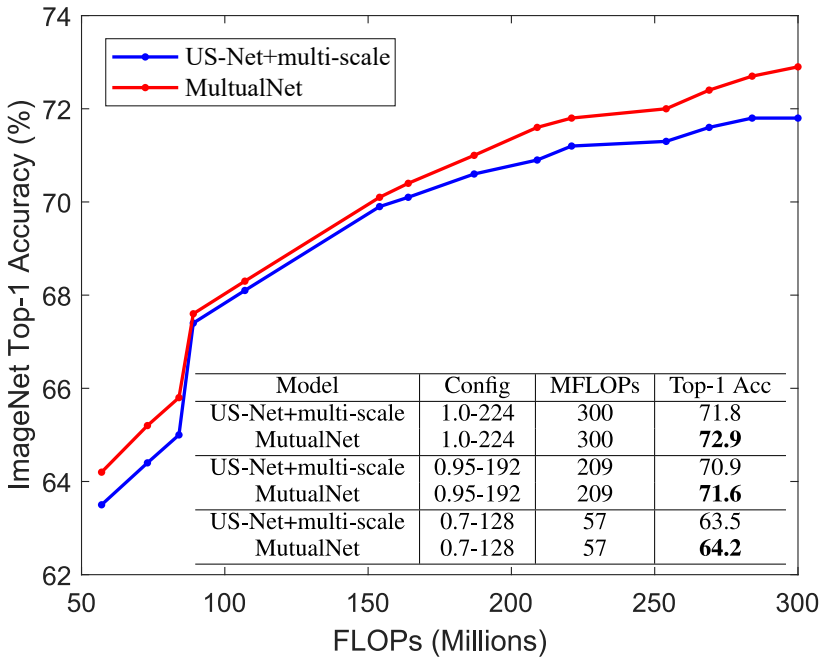
Table 3.3: Comparison between MutualNet and multi-scale data augmentation. MutualNet outperforms simply multi-scale data augmentation.

Model	ImageNet Top-1 Acc
MobileNet v2 (1.0× - 224) - Baseline	71.8%
Baseline + Multi-scale data augmentation	72.0%
MutualNet (MobileNet v2 backbone)	**72.9%**

iteration, the network weights are still optimized in the direction of the same resolution. In contrast, our method randomly samples several sub-networks which share weights with each other. Since sub-networks can select different image resolutions, the weights are optimized in the direction of mixed resolution in each iteration as illustrated in Fig. 3.3. This enables each sub-network to effectively learn multi-scale representations from both network width and resolution. To validate the superiority of our mutual learning scheme, we apply multi-scale data augmentation to I-Net and US-Net and explain the difference with MutualNet.

I-Net + Multi-scale data augmentation. We train MobileNetv2 (1.0× width) with multi-scale images. To have a fair comparison, input images are randomly sampled from scales {224, 192, 160, 128} and the other settings are the same as MutualNet. As shown in Table 3.3, multi-scale data augmentation only marginally improves the baseline (MobileNetv2) while MutualNet (MobileNetv2 backbone) clearly outperforms both of them by considerable margins.

US-Net + Multi-scale data augmentation. Different from our framework which feeds different scaled images to different sub-networks, in this experiment, we *randomly* choose a scale from {224, 192, 160, 128} and feed the *same* scaled image to all sub-networks in each iteration. That is, each sub-network takes the same image resolution. In this way, the weights are still optimized towards a single resolution direction in each iteration. For example, as illustrated in Fig. 3.3, the gradient of the sub-network 0.4× in MutualNet is $\frac{\partial l_{W_{0:0.4} \cdot I_{R=128}} + \partial l_{W_{0:0.8} \cdot I_{R=192}}}{\partial W_{0:0.4}}$, while in US-Net

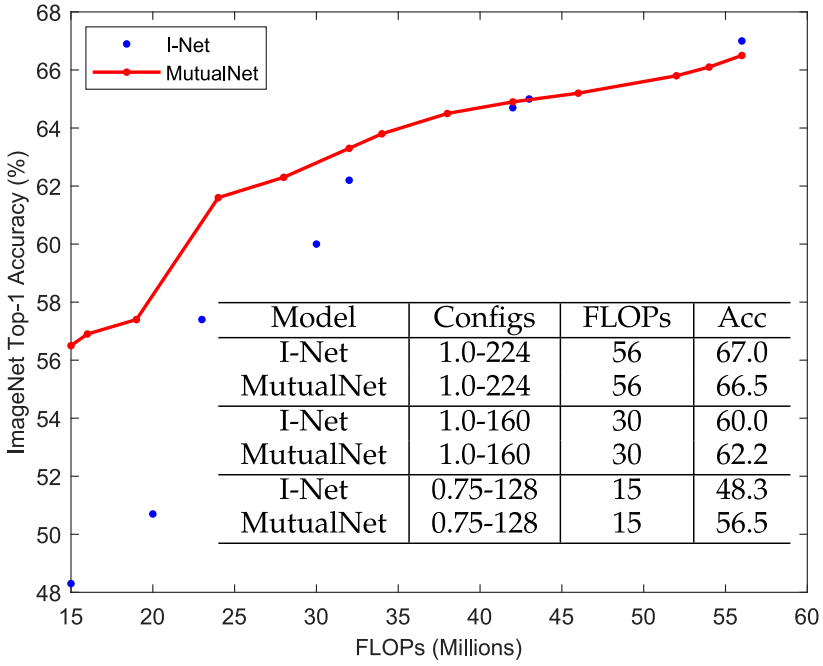


Model	Config	MFLOPs	Top-1 Acc
US-Net+multi-scale	1.0-224	300	71.8
MutualNet	1.0-224	300	**72.9**
US-Net+multi-scale	0.95-192	209	70.9
MutualNet	0.95-192	209	**71.6**
US-Net+multi-scale	0.7-128	57	63.5
MutualNet	0.7-128	57	**64.2**

Figure 3.7: MutualNet and US-Net + multi-scale data augmentation. MutualNet achieves better performance, which shows that the proposed mutual learning is different and more effective than simple multi-scale data augmentation.

+ multi-scale it would be $\frac{\partial l_{W_{0:0.4},I_{R=128}} + \partial l_{W_{0:0.8},I_{R=128}}}{\partial W_{0:0.4}}$. With more sub-networks and input scales involved, the difference between their gradient flows becomes more distinct. As shown in Fig. 3.7, our method clearly outperforms *US-Net + multi-scale data augmentation* over the entire FLOPs spectrum. This experiment is based on MobileNetv2 with the same settings as in Sec. 3.5.1. *These experiments demonstrate that the improvement comes from our mutual learning scheme rather than the multi-scale data augmentation.*

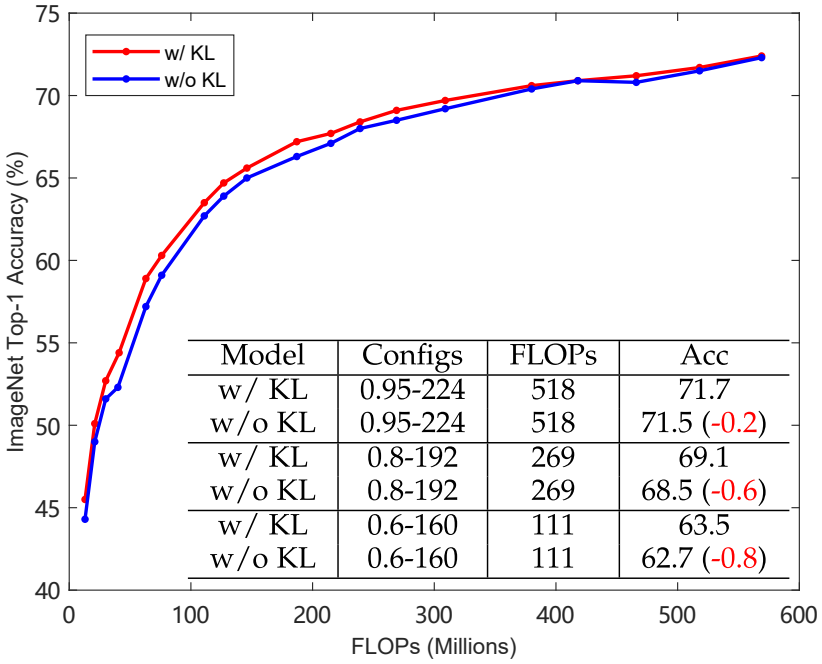
Combine with Dynamic Blocks. As reviewed in Section 2, there is a category of dynamic networks where the adaptive weights are determined by the input. We show that MutualNet can be directly combined with these networks by applying our method to MobileNetV3 [69], where the dynamic blocks are implemented by Squeeze and Excitation (SE) [72]. When sampling sub-networks, we multiply the width factor to both the backbone network and SE block. Then the



Model	Configs	FLOPs	Acc
I-Net	1.0-224	56	67.0
MutualNet	1.0-224	56	66.5
I-Net	1.0-160	30	60.0
MutualNet	1.0-160	30	62.2
I-Net	0.75-128	15	48.3
MutualNet	0.75-128	15	56.5

Figure 3.8: Comparison of independently-trained networks (I-Net) and MutualNet with the MobileNetV3 backbone. MutualNet can be applied to dynamic blocks such as the SE block.

full-network and sub-networks can be trained in the same way as other models. We compare MutualNet and independently trained MobileNetV3 (denoted by **I-Net**) at different configurations in Fig. 3.8. Note that the results of MobileNetV3 are reproduced by us since we could not strictly follow the settings in the original paper (the authors trained MobileNetV3 on 4×4 TPU Pod with a batch size of 4096). We train the network on an 8-GPU server for 150 epochs with a batch size of 1024. The initial learning rate is 0.4 with cosine decay schedule. Our reproduced performance is 0.4% lower than that in the original paper. MutualNet is trained on the same codebase with the same settings. As shown in Fig. 3.8, MutualNet does not outperforms independently-trained MobileNetV3 at large model configurations. We conjecture this is caused by the SE block in MobileNetV3. In MutualNet, sub-networks and the full-network share the same SE block, but their channel attention could be different (sub-networks do not have some channels). Fitting the channel attention to sub-networks may hurt the attention of the full-network, thus leading to decreased



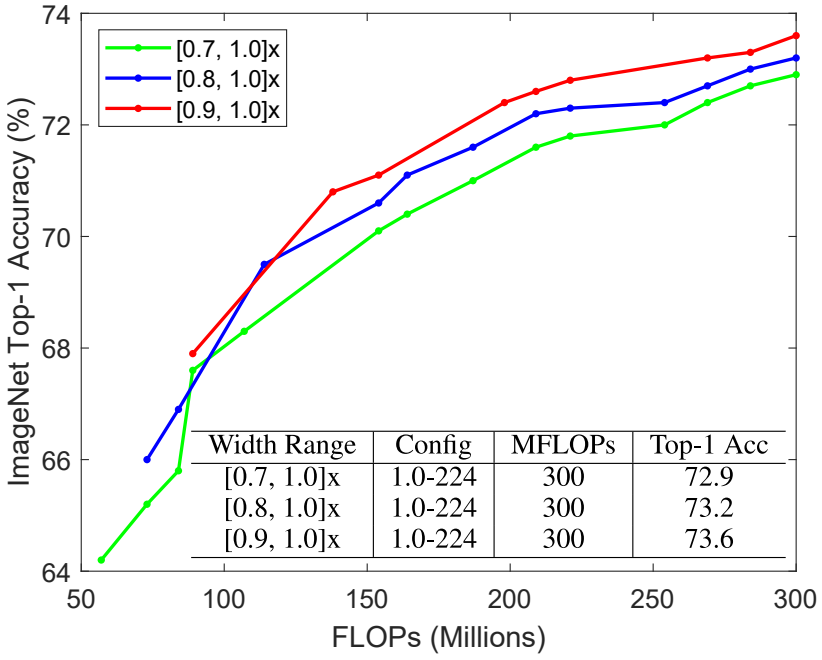
Model	Configs	FLOPs	Acc
w/ KL	0.95-224	518	71.7
w/o KL	0.95-224	518	71.5 (-0.2)
w/ KL	0.8-192	269	69.1
w/o KL	0.8-192	269	68.5 (-0.6)
w/ KL	0.6-160	111	63.5
w/o KL	0.6-160	111	62.7 (-0.8)

Figure 3.9: Contribution of the KLDiv loss to the overall performance. KLDiv loss contributes small improvements to the overall performance. Most improvements are coming from the proposed mutual learning paradigm.

performance. But at smaller configurations, MutualNet is significantly better. Although the overall improvement may not be as significant as on other backbones, MutualNet still has the advantage of covering a wide range of resource constraints by a single model, which makes it easier to deploy on resource-constrained devices. Still, we think that investigating the effective combination of MutualNet with other dynamic inference methods is an interesting problem. We leave further study on this for future work.

3.5.2 Ablation Study

Effects of KL Divergence. During training, we leverage the full-network to supervise sub-networks to enhance knowledge transfer. Here we study how much does the KL Divergence loss



Width Range	Config	MFLOPs	Top-1 Acc
[0.7, 1.0]x	1.0-224	300	72.9
[0.8, 1.0]x	1.0-224	300	73.2
[0.9, 1.0]x	1.0-224	300	73.6

Figure 3.10: Accuracy-FLOPs curves of different width lower bounds. Smaller lower bounds tend to yield worse performance but achieves large adaptive range.

contribute to the overall performance. As shown in Fig. 3.9, *w/ KL* is the original training process and *w/o KL* denotes that the sub-networks are supervised by the ground truth labels. The difference is very marginal where the largest gap is less than 1%. This demonstrates that the KL Divergence loss does benefit the performance, but the main contribution is coming from the mutual learning scheme as explained in Section 3.3.

Effects of Width Lower Bound. The dynamic constraint is affected by the width lower bound. To study its effects, we conduct experiments with three different width lower bounds (0.7×, 0.8×, 0.9×) on MobileNetv2. The results in Fig. 3.10 show that a higher lower bound gives better overall performance, but the dynamic range is narrower. One interesting observation is that the performance of the full-network (1.0×-224) is also largely improved as the width lower bound increases from 0.7× to 0.9×. This property is not observed in US-Net. We attribute this to the

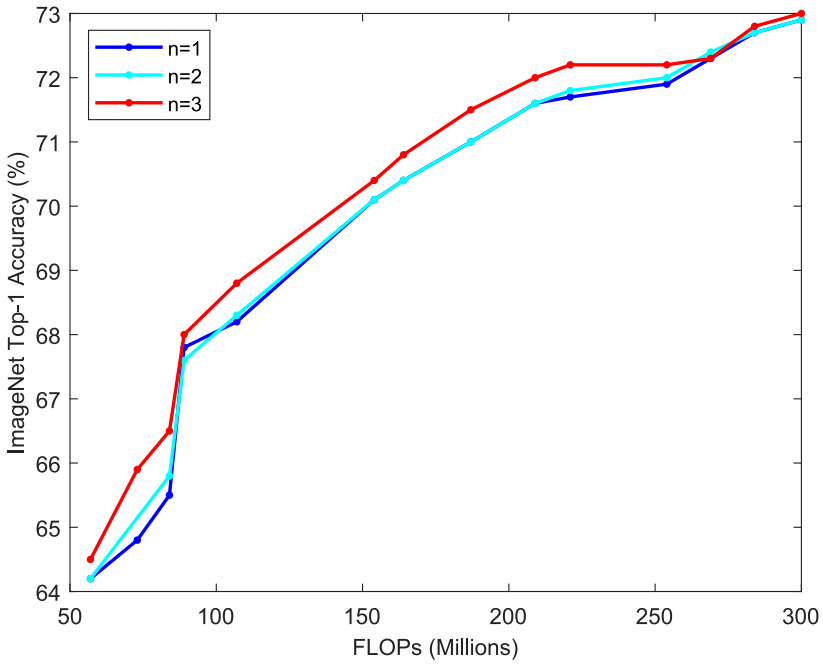


Figure 3.11: The effect of the number of randomly sampled sub-networks during training. The backbone network is MobileNetV2. $n = 1$ means one random width sub-network is sampled with the full-network and the smallest sub-network ($\gamma_w = 0.7$).

robust and well-generalized multi-scale representations which can be effectively re-used by the full-network, while in US-Net, the full-network cannot effectively benefit from sub-networks.

Effect of the number of sub-networks. Fig. 3.11 shows the performance with different numbers of random sub-networks. The backbone network is MobileNetV2. We can see that a larger number of sub-networks could slightly improve the performance over the whole FLOPs spectrum, but it will also increase the training cost. The results also show that even one sub-network can achieve respectable results.

Boosting Single Network Performance. As discussed above, the performance of the full-network is greatly improved as we increase the width lower bound. Therefore, we can apply MutualNet to improve the performance of a single full network **if dynamic budgets is not the concern**. We

Table 3.4: Comparisons of the Top-1 Accuracy (%) of MutualNet and state-of-the-art techniques for boosting a single network. The full network in MutualNet can even outperform state-of-the-art data augmentation and regularization methods.

Method	Cifar-10	Cifar-100	ImageNet
Baseline [196, 64]	96.1	81.2	76.5
Cutout [35]	96.9	81.6	77.1
SENet [72]	/	/	77.6
AutoAug [31]	**97.4**	82.9	77.6
ShakeDrop [182]	95.6	81.7	77.5
Mixup [201]	97.3	82.5	77.9
MutualNet	97.2	**83.8**	**78.6**

compare our method with the popular performance-boosting techniques (e.g., AutoAugmentation (AutoAug) [31], SENet [72] and Mixup [201] etc.) to show its superiority. We conduct experiments using WideResNet-28-10 [196] on Cifar-10 and Cifar-100 [86] and ResNet-50 [64] on ImageNet [32]. MutualNet adopts the width range [0.9, 1.0]$\times$ as it achieves the best-performed full-network in Fig. 3.10. The resolution is sampled from {32, 28, 24, 20} on Cifar-10 and Cifar-100 and {224, 192, 160, 128} on ImageNet. WideResNet is trained for 200 epochs following [196]. ResNet is trained for 120 epochs. The results are compared in Table 3.4. Surprisingly, MutualNet achieves substantial improvements over other techniques even though it is designed to achieve dynamic models. Note that MutualNet is model-agnostic and is as easy as regular training process, so it can take advantage of state-of-the-art network structures and data augmentation techniques.

3.5.3 Transfer Learning Evaluation

To evaluate the representations learned by our method, we further conduct experiments on three popular transfer learning datasets, Cifar-100 [86], Food-101 [9] and MIT-Indoor67 [130]. Cifar-100 is for superordinate-level object classification, Food-101 is for fine-grained classification and

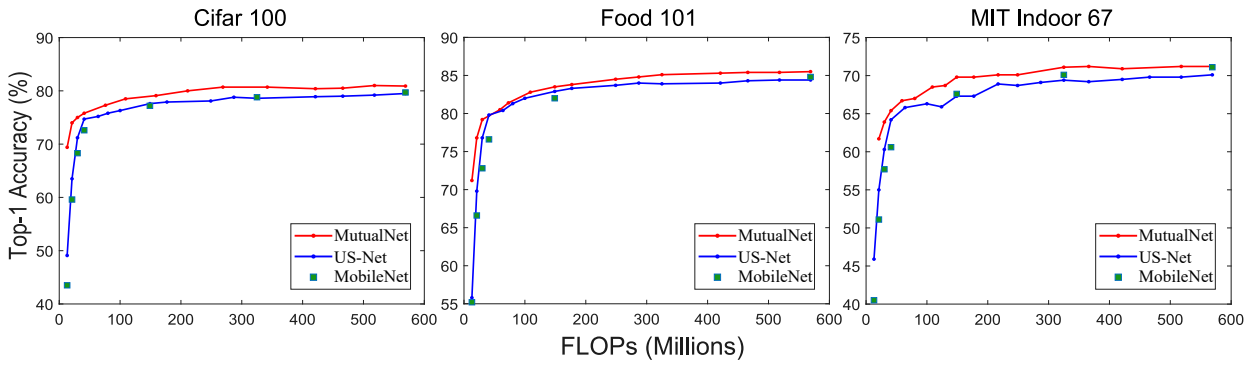
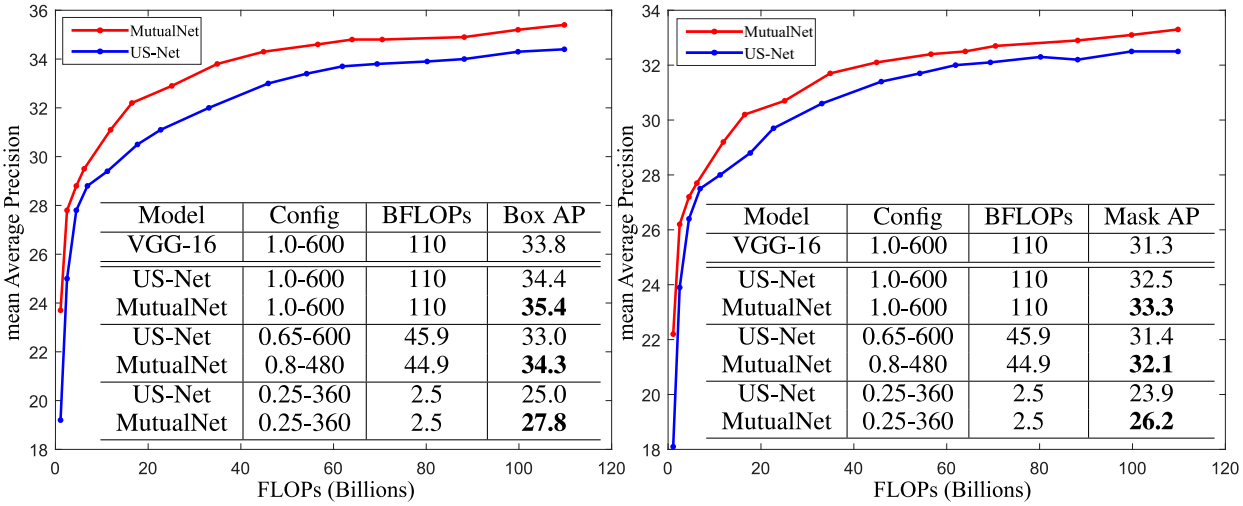


Figure 3.12: Accuracy-FLOPs curves of different methods on different transfer learning datasets. MobileNet is trained independently at different model configurations. MutualNet achieves consistently better performance on different datasets.

MIT-Indoor67 is for scene classification. Such a large variety of datasets can strongly demonstrate the robustness of the learned representations. We compare our approach with US-Net and MobileNetv1. We fine-tune ImageNet pre-trained models with a batch size of 256, initial learning rate of 0.1 with cosine decay schedule and a total of 100 epochs. Both MutualNet and US-Net are trained with width range [0.25, 1.0]$\times$ and tested with resolutions from {224, 192, 160, 128}. The results are shown in Fig. 3.12. Again, our MutualNet achieves consistently better performance compared to US-Net and MobileNet. This verifies that MutualNet is able to learn well-generalized representations.

3.5.4 Object Detection and Instance Segmentation

We also evaluate our method on COCO object detection and instance segmentation [101]. The experiments are based on Mask-RCNN-FPN [62, 100] and MMDetection [18] toolbox on VGG-16 [141] backbone. We first pre-train VGG-16 on ImageNet following US-Net and MutualNet respectively. Both methods are trained with width range [0.25, 1.0]$\times$. Then we fine-tune the pre-trained models on COCO. The feature pyramid network (FPN) neck and detection head are



Model	Config	BFLOPs	Box AP
VGG-16	1.0-600	110	33.8
US-Net	1.0-600	110	34.4
MutualNet	1.0-600	110	**35.4**
US-Net	0.65-600	45.9	33.0
MutualNet	0.8-480	44.9	**34.3**
US-Net	0.25-360	2.5	25.0
MutualNet	0.25-360	2.5	**27.8**

Model	Config	BFLOPs	Mask AP
VGG-16	1.0-600	110	31.3
US-Net	1.0-600	110	32.5
MutualNet	1.0-600	110	**33.3**
US-Net	0.65-600	45.9	31.4
MutualNet	0.8-480	44.9	**32.1**
US-Net	0.25-360	2.5	23.9
MutualNet	0.25-360	2.5	**26.2**

Figure 3.13: mAP-FLOPs curves of MutualNet and US-Net on object detection (left) and instance segmentation (right). The results are based on Mask-RCNN. All models follow the same training settings.

shared among different sub-networks. For simplicity, each sub-network is trained with the ground truth. The other training procedures are the same as training ImageNet classification. Following common settings in object detection, US-Net is trained with image resolution 1000×600. Our method randomly selects resolutions from $1000 \times \{600, 480, 360, 240\}$. All models are trained with $2\times$ schedule for better convergence and tested with different image resolutions. The mean Average Precision (AP at IoU=0.50:0.05:0.95) are presented in Fig. 3.13. These results reveal that our MutualNet significantly outperforms US-Net under all resource constraints. Specifically, for the full network ($1.0\times$-600), MutualNet significantly outperforms both US-Net and independent network. This again validates the effectiveness of our width-resolution mutual learning scheme. Fig. 3.14 provides some visual examples which reveal that MutualNet is more robust to small-scale and large-scale objects than US-Net.

Figure 3.14: Visualization examples of MutualNet and US-Net on object detection and instance segmentation. Detection and segmentation results are demonstrated by bounding boxes and masks respectively. To facilitate comparison, we use **yellow boxes** to highlight the objects that MutualNet detects but US-Net fails. [Best viewed with zoom-in.]

3.6 Summary

In summary, we highlight the importance of simultaneously considering network width and input resolution for efficient network design. A new framework namely MutualNet is proposed to mutually learn from network width and input resolution for adaptive accuracy-efficiency trade-offs. Extensive experiments have shown that it significantly improves inference performance per FLOP on various datasets and tasks. The mutual learning scheme is also an effective training strategy for boosting single network performance. The generality of the proposed framework allows it to translate well to generic problem domains.

CHAPTER 4: ADAPTIVE NEURAL NETWORK FOR VIDEO UNDERSTANDING

The work in this Chapter has been published in the following paper:

Taojiannan Yang, Sijie Zhu, Matias Mendieta, Pu Wang, Ravikumar Balakrishnan, Minwoo Lee, Tao Han, Mubarak Shah, Chen Chen. "MutualNet: Adaptive ConvNet via Mutual Learning from Different Model Configurations." IEEE Transaction on Pattern Analysis and Machine Intelligence. 2021.

In Chapter 3, we demonstrate the effectiveness of MutualNet in learning adaptive neural networks for various image understanding tasks. In this chapter, we generalize the idea to video understanding. We are the first to achieve adaptive spatiotemporal (3D) neural networks for video understanding. A popular way to model spatiotemporal information in videos is to extend 2D convolutions to 3D convolutions. The computation cost of a vanilla 3D convolutional layer is given by

$$K \times K \times K \times C_i \times C_o \times H \times W \times T. \tag{4.1}$$

Here, K denotes the kernel size, and C_i and C_o respectively are the input and output channels of this layer. H, W, T denote the spatial-temporal size of the output feature map. Similar to 2D networks, we sample model configurations by network width, spatial and temporal resolution. Following the notations in Chapter 3, we denote a model configuration by width-spatial-temporal. The scaling coefficients γ_w and γ_r are the same as defined before. γ_t is the temporal resolution coefficient where

$\gamma_t \in [0, 1]$. The computational cost is reduced to

$$K \times K \times K \times \gamma_w C_i \times \gamma_w C_o \times \gamma_s H \times \gamma_s W \times \gamma_t T. \tag{4.2}$$

The computational cost is now reduced by $\rho = \gamma_w^2 \gamma_s^2 \gamma_t$ times. For single-branch structures [157, 15, 43], which are extended from 2D networks, our method can be easily applied by randomly sampling both spatial and temporal dimensions during training. For two-branch structures such as SlowFast [45], where spatial and temporal dimensions are processed asymmetrically, we conduct asymmetric sampling.

In the following sections, we first show that the temporal dimension can bring additional knowledge to be transferred among different model configurations. Then we explain MutualNet structure on one-branch and two-branch video models.

4.1 Knowledge in Different 3D Model Configurations

Temporal modeling is essential in 3D networks. Following 2D MutualNet, we first demonstrate that different model configurations will focus on different spatial-temporal semantic regions. Fig. 4.1 shows the spatial and temporal distributions of network activation following CAM [206]. Higher value means more contribution to the final logit value. Although both models generate the prediction as “headbutting", their decisions are based on different areas of different frames. The input of the left model has 8 frames, and the 2nd and 8th frames contribute the most to the final prediction. While the right model only has 4 input frames, where those two key frames in the left model are not sampled. So it has to learn other semantic information, forcing a change in both temporal and spatial activation distributions. For example, the activation value of the 5th frame exceeds that of the 3rd frame in the right model, which is the opposite case in the left model. The

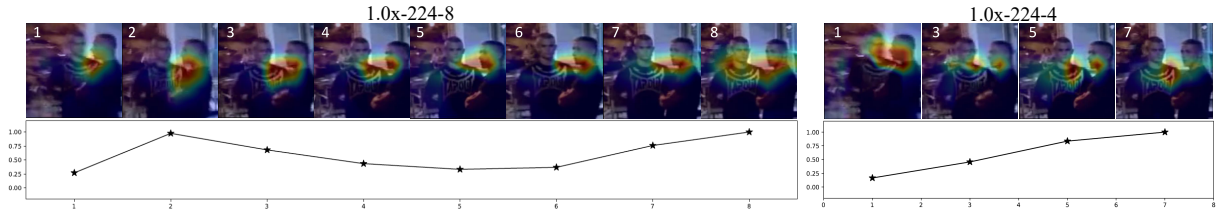


Figure 4.1: Class activation map along spatial and temporal dimensions of two network configurations. X-axis is the frame index number and y-axis is the normalized activation value. The action is “headbutting" from the Kinetics-400 dataset.

spatial attention areas are also unique in the two models. In the first frame, the attention is on the shoulder in the left model, but shifts to the face in the right model indicating that a varied set of visual cues is captured.

4.2 3D MutualNet Training

4.2.1 One-branch Structure

In one-branch structures, the 3D convolutions are directly extended from 2D convolutions so that we can apply the mutual learning strategy in the same way as 2D networks. The left half of Fig. 4.2 shows how mutual training works in single pathway structures. In each training iteration, we sample two sub-networks (by the width factor γ_w) in addition to the full-network. Sub-networks share the parameters with the full-network in the same way as 2D networks. The full-network is fed with the highest spatial-temporal resolution inputs, while sub-networks are fed with randomly downsampled inputs. Similar to 2D MutualNet training, the full-network is supervised by the ground-truth label while sub-networks are supervised by the full-network to facilitate knowledge transfer. The total loss is the summation of the full-network’s loss and sub-networks’ losses.

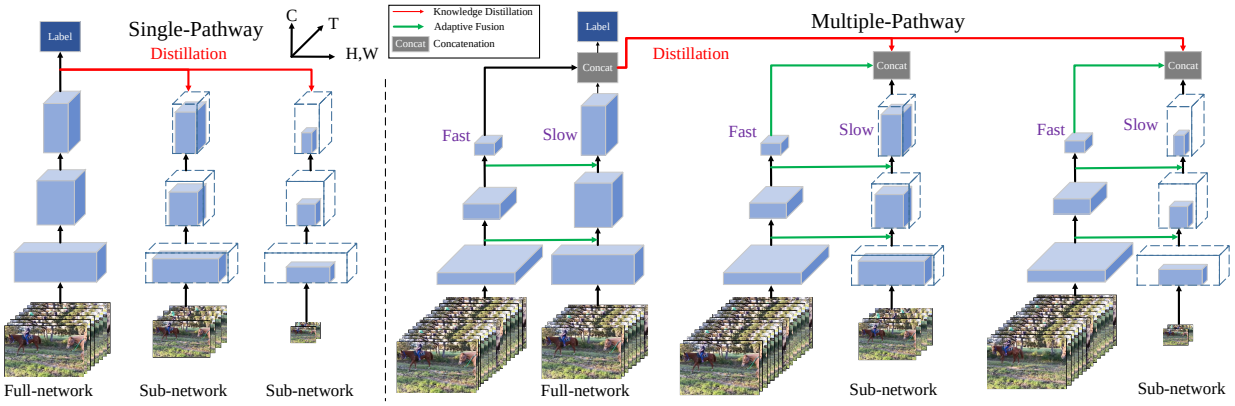


Figure 4.2: An overview of 3D MutualNet training. Left is for single-pathway structures, which is similar to 2D MutualNet training. Right is for multiple-pathway structures, where only the Slow branch is downsampled. The two branches are fused by the proposed Adaptive Fusion block.

4.2.2 Two-branch Structure

Recently, SlowFast [45] proposes that the temporal dimension should not be processed symmetrically to the spatial dimension, as slow and fast motions contain different information for identifying an action class. SlowFast shows that a lightweight fast pathway, which aims to capture fine motion information, is a good complement to the slow pathway, which mainly captures spatial semantics. This inspires us to leverage multiple-pathway trade-offs in 3D MutualNet. The structure is shown in the right half of Fig. 4.2. Since the fast pathway is lightweight (about 10% of the overall computation), reducing its spatial-temporal resolution or network width is not beneficial for the overall computation-accuracy trade-off. In two-branch structures, we keep the respective $\gamma_w, \gamma_s, \gamma_t = 1$ for the Fast pathway so that it can provide complementary information for the Slow path with its own $\gamma_w, \gamma_s, \gamma_t \leq 1$. *Note that this complementary information is not only on temporal resolution but also on spatial resolution.*

Adaptive Fusion. Given fixed temporal resolutions for two pathways, the fusion is conducted by lateral connections with time-strided convolution in SlowFast [45]. *However, since all the three*

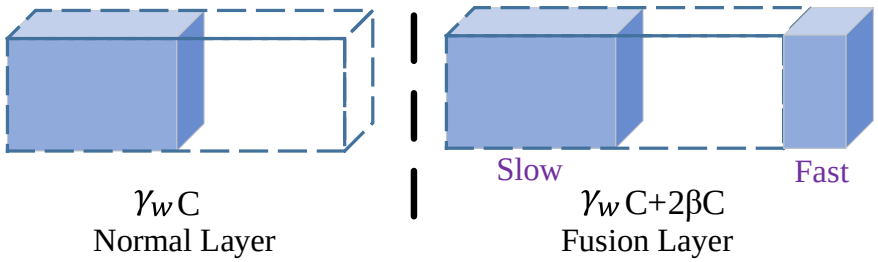


Figure 4.3: An illustration of adaptive fusion on network channels.

dimensions (γ_w, γ_s, γ_t) of the Slow pathway can change during training, directly applying the time-strided convolution does not work for our framework. Therefore, we design an adaptive fusion block for multiple-pathway 3D MutualNet.

Following SlowFast [45], we denote the feature shape of a standard Slow pathway as $\{T, S^2, C\}$, where S is the spatial resolution and C is the number of channels. Then the feature shape of adaptive Slow pathway is $\{\gamma_t T, (\gamma_s S)^2, \gamma_w C\}$. The feature shape of Fast pathway remains $\{\alpha T, S^2, \beta C\}$ as in [45] (α, β are hyperparameters defined in [45]. $\alpha = 8, \beta = 1/8$ for SlowFast 4×16). Following the settings in [45], we first perform a 3D convolution of 5×1^2 kernel with $2\beta C$ output channels and a stride of α. The output feature shape of this convolution layer is $\{T, S^2, 2\beta C\}$. To fuse it with the adaptive Slow pathway (whose shape is $\{\gamma_t T, (\gamma_s S)^2, \gamma_w C\}$), we perform a spatial interpolation and temporal down-sampling to make the output shape $\{\gamma_t T, (\gamma_s S)^2, 2\beta C\}$. Then the final feature shape after the fusion is $\{\gamma_t T, (\gamma_s S)^2, (\gamma_w + 2\beta)C\}$. As shown in Fig. 4.3, normal convolution layers of adaptive Slow pathway have $\gamma_w C$ channels indexing from the left, while the first convolution layer after each fusion has $\gamma_w C + 2\beta C$ input channels. The last $2\beta C$ channels are always kept for the output of Fast pathway, while the first $\gamma_w C$ channels from the left can vary for each iteration. This operation enforces an exact channel-wise correspondence between the fusion features and the parameters in convolution layers.

Table 4.1: Training cost of 2D MutualNet and independent 2D models of different scales. The backbone is MobileNet v1. * indicates expected values.

MobileNet v1	Independent						MutualNet
Scale	$\times 0.1$	$\times 0.3$	$\times 0.5$	$\times 0.7$	$\times 0.9$	$\times 1.0$	$\times 0.02 \sim \times 1.0$
MFLOPs	57	171	284	398	512	569	910*
Total	1991						910*
Mins/epoch	1*	3*	5*	7*	9*	10	20
Total	35*						20

Table 4.2: Training cost of 3D MutualNet and independent 3D models of different scales. The backbone is Slow-8$\times$8. * indicates expected values.

Slow-8$\times$8	Independent						MutualNet
Scale	$\times 0.1$	$\times 0.3$	$\times 0.5$	$\times 0.7$	$\times 0.9$	$\times 1.0$	$\times 0.06 \sim \times 1.0$
GFLOPs	5.5	16.4	27.3	38.2	49.1	54.5	74.8*
Total	191						74.8*
Mins/epoch	5.8*	17.4*	29.0*	40.6*	52.2*	58	69
Total	203*						69

4.3 3D Model Inference

Similar to 2D networks, we need to find the best-performing configuration at each resource constraint. We evaluate different width-spatial-temporal configurations on a validation set. Then the complexity-configuration table can be obtained by following the steps in Section 3.4. For real deployment, the model and the table need to be maintained, and therefore the memory consumption is essentially the same as a single model. The model can be adjusted according to the table to meet different resource budgets.

4.4 Complexity of MutualNet Training

Since we additionally sample sub-networks during training, MutualNet consumes more computational cost than training a single model. However, we show that the training cost is several times less than training many independent models. In Table 4.1, we measure the theoretical complexity (i.e., FLOPs) and practical wall-clock time of training MutualNet and independent models. The network backbone is MobileNetv1 [70]. In MutualNet, the width range is [0.25, 1.0]$\times$ and the resolution range is {224, 192, 160, 128}. This achieves a dynamic constraint of [13, 569] MFLOPs, which corresponds to a model scale from $\times$0.02 to $\times$1.0. Note that we use 1.0$\times$ to denote the network width coefficient while $\times$1.0 to denote the overall model scale. For independent training, we train a set of models where the smallest one is $\times$0.1 and the stepsize is $\times$0.2. The FLOPs of MutualNet training is estimated by taking the expectation of Eq. 3.2. The wall-clock time of the full model ($\times$1.0) and MutualNet is measured on an 8$\times$2080TI GPU sever with a batch-size of 1024. Other models' training time is estimated by the model FLOPs because the practical time depends on the manner the model is scaled down. However, it should be higher than the estimated values because the data loading/processing time does not decrease if the model is scaled down. Table 4.1 shows that the training cost of MutualNet is around 2 times of the full-model, but it is much smaller than independently training each model.

We also evaluate the training cost on 3D networks (Slow-8$\times$8 [45]) in Table 4.2. The training time is measured on an 8$\times$2080TI GPU sever with a batch-size of 64. In MutualNet, the width range is [0.63, 1.0]$\times$, spatial resolution is {142, 178, 224} and temporal resolution is {3, 5, 8}. This achieves a model scale from $\times$0.06 to $\times$1.0. We also evaluate a group of independent models from $\times$0.1 to $\times$1.0. We can see that although the theoretical complexity of MutualNet is about 1.4 times of the full-model, the wall-clock time is only slightly higher. This is because the data loading/processing is time-consuming in 3D networks, the additional cost introduced by sub-networks

is therefore not that significant. *In summary, MutualNet saves time compared to independently training several models, and it only needs to deploy one model to meet dynamic resource constraints during inference.*

4.5 Experiments

To the best of our knowledge, we are the first to achieve adaptive 3D networks. So we only compare our method with independently-trained networks. We conduct experiments based on Slow/SlowFast [45] and X3D [43] backbones, which are state-of-the-art 3D network structures. Following previous works [45, 43], we evaluate the method on the following three video datasets.

Kinetics-400 [82] is a large scale action classification dataset with $\sim$240k training videos and 20k validation videos trimmed as 10s clips. However, since some of the YouTube video links have expired, we can not download the full dataset. Our Kinectic-400 only has 237,644 out of 246,535 training videos and 19,761 validation videos. The training set is about 4% less than that in SlowFast [45] and some of the videos have a duration less than 10s. This leads to an accuracy drop of 0.6% on Slow-8$\times$8, 1.2% on SlowFast-4$\times$16 and 0.93% on X3D-M as we reproduce the results with the officially released codes [40]. **Charades** [139] is a multi-label action classification dataset with longer activity duration. The average activity duration is $\sim$30 seconds. The dataset is composed of $\sim$9.8k training videos and 1.8k validation videos in 157 classes. The evaluation metric is mean Average Precision (mAP). **AVA** [54] is a video dataset for spatio-temporal localization of human actions. It consists of 211k training and 57k validation video segments. We follow previous works [45] to report the mean Average Precision (mAP) on 60 classes using an IoU threshold of 0.5.

Implementation Details. For single-pathway structures, we adopt Slow 8$\times$8 and X3D-M as our backbone. For multiple-pathway structures we use SlowFast 4$\times$16 due to the limitation of GPU

memory. For Slow and SlowFast, the width factor γ_w is uniformly sampled from $[0.63, 1.0]\times$. The spatial resolution factor is $\gamma_s \in \{0.63, 0.80, 1.0\}$ (corresponding to $\{142, 178, 224\}$) and the temporal resolution factor is $\gamma_t \in \{0.4, 0.63, 1.0\}$ (corresponding to $\{3, 5, 8\}$). For X3D-M, the width factor γ_w is uniformly sampled from $[0.63, 1.0]\times$. The spatial resolution factor is $\gamma_s \in \{0.63, 0.71, 0.86, 1.0\}$ (corresponding to $\{142, 160, 192, 224\}$) and the temporal resolution factor is $\gamma_t \in \{0.4, 0.6, 0.8, 1.0\}$ (corresponding to $\{6, 9, 12, 16\}$). Other training settings are the same as the official codes [40].

4.5.1 Main Results

Evaluation on Kinetics-400. In Fig. 4.4, we report the results of MutualNet on different backbones along with its separated trained counterparts. The original results (reported in paper) are denoted as "-P" and our reproduced results using official code [40] are denoted as "-R". We report both results to have a fair comparison since we can not reproduce the original results due to lack of data. For results of MutualNet, we use different line colors to show different dimensions for accuracy-efficiency trade-off. Red means the spatial resolution is reduced to meet the dynamic resource budget in this range. Similarly, green stands for temporal resolution and blue stands for network width. **Black** indicates multiple dimensions are involved for one trade-off step.

As shown in Fig. 4.4, MutualNet consistently outperforms its separately trained counterparts on three network backbones. It achieves significant improvements over our reproduced results and shows clear advantages over the reported results in the paper. The improvement is even more significant (3.5% on Slow backbone and 1.6% on SlowFast backbone) for small resource budgets. This is because our method allows the model to find a better width-spatial-temporal trade-off at each resource budget. And the mutual learning scheme can transfer the knowledge in large configurations to small models to further improve its performance. On X3D-M backbone, our method

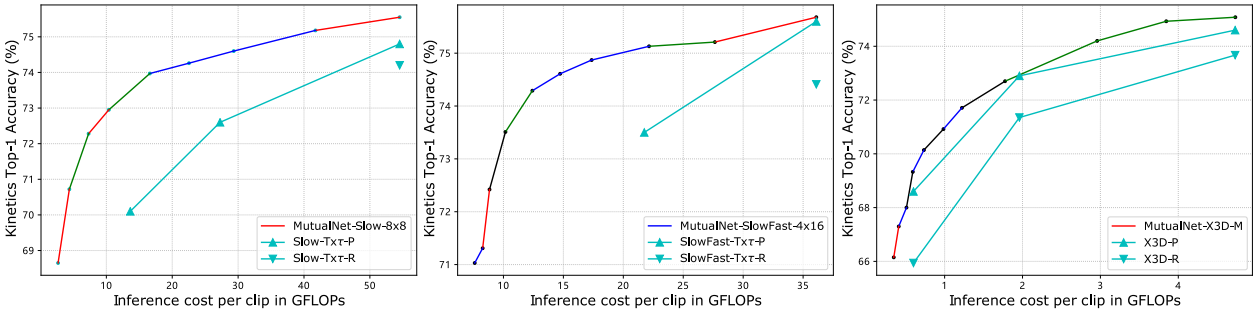


Figure 4.4: Comparison of MutualNet and its independently trained counterparts under different computational constraints for video action recognition. Both the results in the corresponding papers and our reproduced results are reported for a fair comparison. Our reproduced results are lower because of lack of training data.

achieves consistent improvements under different budgets. Note that X3D finds the best-performed model configuration by a search process. It trains many model configurations independently and choose the best one, while MutualNet train all configurations jointly which saves training time and improves the overall performance.

Comparison with state-of-the-art. Based on X3D-M backbone, we compare MutualNet with state-of-the-art methods [158, 99, 45, 43] for action recognition in Fig. 4.5. The results are based on 10-view testing. Note that **the x-axis is in log-scale for better visualization.** We can see that based on the state-of-the-art structure (X3D), MutualNet substantially outperforms previous works (including X3D). This reveals MutualNet is a general training framework and can benefit from improved model structures.

4.5.2 Ablation Study

Contribution of Temporal Dimension. To investigate the effect of the temporal dimension, we train MutualNet-Slow-8×8 without a temporal trade-off. To reduce the computation by 16× (around [0.06,1]), the lower bounds of $\gamma_s, \gamma_t, \gamma_w$ are $0.63, 0.4, 0.63$. However, if we keep $\gamma_t = 1$,

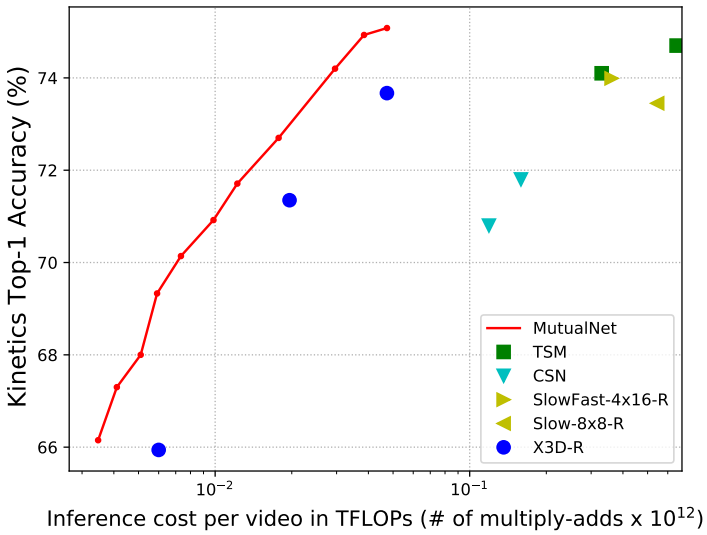


Figure 4.5: Comparison of MutualNet-X3D-M with state-of-the-art 3D networks. MutualNet outperforms other video models at different model complexities. X-axis is in log-scale.

Table 4.3: The contribution of temporal dimension in mutual training. w/o T indicates remove temporal dimension. Including temporal dimension in 3D MutualNet achieves better performance.

	Model	$S^2 \times T, \gamma_w$	top-1	GFLOPs×views	Param
×1	Slow-8×8-P[45]	$256^2 \times 8, 1.0$	74.8	54.5×30	32.5M
	Slow-8×8-R	$256^2 \times 8, 1.0$	74.2	54.5×30	32.5M
	MutualNet-Slow-8×8 w/o T	$256^2 \times 8, 1.0$	75.1	54.5×30	32.5M
	MutualNet-Slow-8×8	$256^2 \times 8, 1.0$	**75.6**	**54.5×30**	**32.5M**
×0.25	Slow-2×32-P [45]	$256^2 \times 2, 1.0$	70.1	13.6×30	32.5M
	MutualNet-Slow-8×8 w/o T	$224^2 \times 8, 0.6$	73.3	15.4×30	**11.8M**
	MutualNet-Slow-8×8	$224^2 \times 5, 0.73$	**73.6**	**14.1×30**	17.5M
×0.06	A3D-Slow-8×8 w/o T	$112^2 \times 8, 0.5$	67.8	2.8×30	**8.3M**
	MutualNet-Slow-8×8	$142^2 \times 3, 0.63$	**68.7**	**2.7×30**	13.0M

then the lower bounds of γ_s, γ_w have to be $0.5, 0.5$ to provide the same coverage range. Table 4.3 shows that removing the temporal dimension in 3D MutualNet leads to a lower accuracy for both the full-network ($\times 1$) and sub-networks ($\times 0.25$, $\times 0.06$).

Table 4.4: Comparison between 3D MutualNet and multi-scale training on Slow-8×8 backbone. MutualNet significantly outperforms multi-scale training

Model	top-1	top-5
Slow-8×8-P [45]	74.8	91.6
Slow-8×8-R	74.2	91.3
Slow-8×8-multi-scale	73.4	90.8
MutualNet-Slow-8×8 ×1	**75.6**	**91.8**

Effectiveness of Mutual Learning. As explained in Chapter 3, our proposed mutual learning paradigm is different from multi-scale training. To show the effectiveness of our mutual learning paradigm, we train a Slow-8×8 with randomly sampled spatial-temporal resolutions. The candidate resolutions are the same as those of MutualNet-Slow-8×8. As shown in Table 4.4, simply applying spatial-temporal multi-resolution training does not improve the performance, because the network is partly trained with low resolution inputs while the testing resolution is always the highest. On the contrary, our mutual training always feeds the full-network with the highest spatial-temporal resolution, but allows sub-networks to learn multi-resolution representations. This procedure will not hurt the performance of the full-network.

4.5.3 Transfer Learning Evaluation

Evaluation on Chrades. We finetune the models trained on Kinetics-400 on Charades. For SlowFast models, we use the pre-trained models reproduced by us for a fair comparison. For MutualNet models, we do not perform adaptive training during finetuning. That means both SlowFast models and MutualNet models follow the same finetuning process on Charades. The only difference is the pre-trained models. We follow the training settings in the released codes [40]. Since we train the model on 4 GPUs, we reduce the batch-size and base learning rate by half following the

Table 4.5: Comparison of different models on Charades. MutualNet achieves better performance than independently-trained models.

Model	Pretrain	mAP	GFLOPs×views
CoViAR, R-50 [172]	ImageNet	21.9	N/A
Asyn-TF, VGG16 [138]	ImageNet	22.4	N/A
MultiScale TRN [205]	ImageNet	25.2	N/A
Nonlocal, R-101 [166]	ImageNet+Kinetics	37.5	544×30
Slow-8×8	Kinetics	34.7	54.5×30
MutualNet-Slow-8×8	Kinetics	**35.6**	54.5×30

linear scaling rule [52]. All other settings remain unchanged. As can be seen in Table 4.5, MutualNet model outperforms its counterpart (Slow-8×8) by 0.9% without increasing the computational cost. Note that the only difference lies in the pre-trained model, so the improvement demonstrate that our method helps the network learn effective and well-generalized representations which are transferable across different datasets.

Evaluation on AVA Detection. Similar to the experiments in Charades, we follow the same training settings as the released SlowFast codes [40]. The detector is similar to Faster R-CNN [133] with minimal modifications adopted for video. The region proposals are pre-computed by an off-the-shelf person detector. Experiments are conducted on AVA v2.1. All models are trained on a 4-GPU machine for 20 epochs with a batch-size of 32. The base learning rate is 0.05 with linear warm-up for the first 5 epochs. The learning rate is reduced by a factor of 10 at the 10th and 15th epochs. Both SlowFast pre-trained models and MutualNet pre-trained models are finetuned following the standard training procedure; the only difference is the pre-trained models. As shown in Table 4.6, MutualNet pre-trained model also outperforms SlowFast and previous methods. Note that only the pre-trained weights are different in the experiments, so the improvements are not marginal and clearly demonstrate the effectiveness of the learned representations.

Table 4.6: Comparison of different models on AVA v2.1. MutualNet achieves better performance than independently-trained models.

Model	Flow	Pretrain	mAP
I3D [15]		Kinetics-400	14.5
I3D [15]	✓	Kinetics-400	15.6
ACRN, S3D [145]	✓	Kinetics-400	17.4
ATR, R-50+NL [79]		Kinetics-400	20.0
Slow-8×8		Kinetics-400	20.2
MutualNet-Slow-8×8		Kinetics-400	**20.6**

4.6 Summary

In this chapter, we present how to apply MutualNet to train adaptive 3D networks. To the best of our knowledge, we are the first to achieve adaptive 3D network. 3D MutualNet considers network width and input spatiotemporal resolution and randomly samples different spatial-temporal-width configurations in network training. A spatial-temporal distillation scheme is developed to facilitate knowledge transfer among different configurations. The training paradigm is generic and applicable to any 3D networks. Using the same backbone, 3D MutualNet outperforms the state-of-the-art SlowFast [45] networks under various computation constraints on Kinetics. Extensive evaluations also validate the effectiveness of the learned representations of 3D MutualNet for cross dataset and task transfer.

CHAPTER 5: IMPROVED REPRESENTATION LEARNING BY SELF-REGULARIZATION

The work in this Chapter has been published in the following paper:

Taojiannan Yang, Sijie Zhu and Chen Chen. "GradAug: A New Regularization Method for Deep Neural Networks." Neural Information Processing Systems. 2020.

Learning good representations from data are fundamental for downstream visual tasks. In this work, we define good representations as those that can generalize well to various samples. Motivated by this, we propose a new representation learning method. The idea is that when a random transformation (e.g., random rotation, random scale, random crop, etc.) is applied to the input image, a well-generalized network should still recognize the transformed image as the same object. Moreover, when we sample a sub-network from the full network, the sub-network should also make the same prediction as the full network. To this end, during training, we sample several sub-networks from the full network and feed them with differently augmented images and regularize their predictions to be consistent. We show that this self-regularization paradigm will introduce meaning disturbance to the gradients of full network, which is termed as Gradient Augmentation (GradAug). In contrast, other popular regularization methods only add random noises to the original gradients.

This chapter is organized as follows: Section 5.1.1 introduces the details of our proposed representation learning method GradAug. Section 5.1.2 provides an in-depth analysis on how GradAug and other regularization methods influence the gradients of the full network during training. In Section 5.2, we conduct comprehensive experimental evaluation and ablation study to demonstrate

the effectiveness of the proposed method. Section 5.3 provides a summary for this chapter.

5.1 Methodology

5.1.1 GradAug

When applying some random transformations to an image, human can still recognize it as the same object. We expect deep neural networks to have the same generalization ability. GradAug aims to regularize sub-networks with differently transformed training samples. There are various of methods to generate sub-networks during training. Previous works [142, 75, 89] usually stochastically drop some neurons, layers or paths. In GradAug, we expect the final full-network to take advantage of the learned representations of the sub-networks. Therefore, we sample sub-networks in a more structured manner, that is by the network width. We define θ as the model parameter. Without loss of generality, we use convolutional layers for illustration, then $\theta \in \mathbb{R}^{c_1 \times c_2 \times k \times k}$, where c_1 and c_2 are number of input and output channels, k is the convolution kernel size. We define the width of a sub-network as $w \in [\alpha, 1.0]$, where α is the width lower bound. The weights of the sub-network is θ_w. Different from random sampling, we always sample the first $w \times 100\%$ channels of the full-network and the sub-network weights are $\theta_w \in \mathbb{R}^{wc_1 \times wc_2 \times k \times k}$. In this way, a larger sub-network always share the representations of a smaller sub-network in a weights-sharing training fashion, so it can leverage the representations learned in smaller sub-networks. Iteratively, sub-networks can construct a full-network with diversified representations. Figure 5.1 shows the class activation maps (CAM) [206] of the sub-network and full-network. The full-network pays attention to several regions of the object because it can leverage the representation of the sub-network. For example, when the sub-network ($w = 0.9$) focuses on one dog in the image, the full-network shares this attention and uses the other network part to capture the information of another dog. Therefore, the full-network learns richer semantic information in the image, while the baseline model only

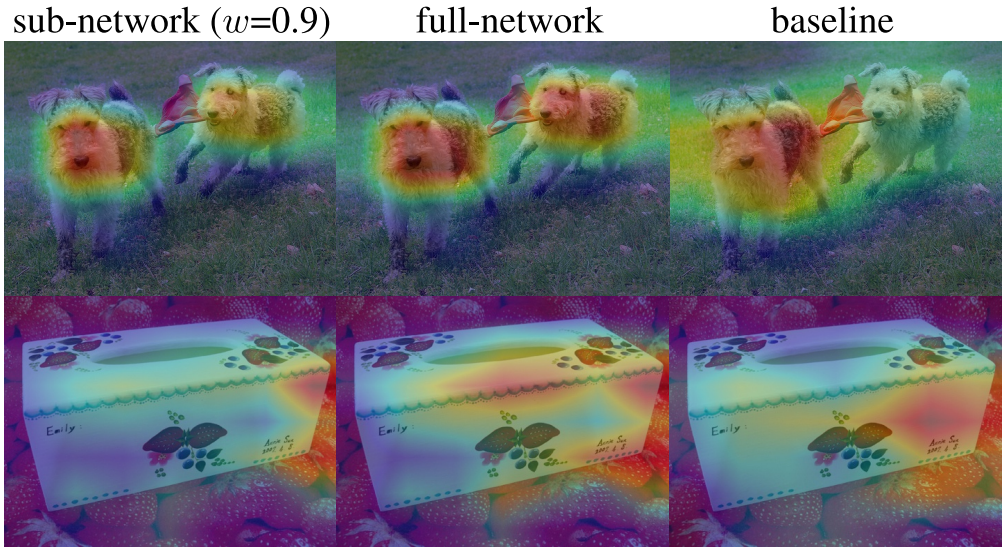


Figure 5.1: Class activation maps (CAM) of the network trained by GradAug and the baseline. The full-network shares the attention of the sub-network and focuses on multiple semantic regions.

models a single region and does not fully comprehend the salient information of the image. To make the method simple and generic, we choose among the most commonly used transformations such as random scales, random rotations, random crops, etc. In the experiments, we show that a simple random scale transformation can already achieve state-of-the-art performance on image classification, and it can be directly applied to other applications. Moreover, we can use more powerful augmentations such as CutMix for further enhanced performance.

Training procedure. The training procedure of GradAug is very similar to the regular network training. In each training iteration, we train the full-network with the original images, which is the same as the regular training process. Then we additionally sample n sub-networks and train them with randomly transformed images. Finally, we accumulate the losses of full-network and sub-networks to update the model weights. This naive training approach achieves good training accuracy but the testing accuracy is very low. This is caused by the batch normalization (BN) [76] layers. The BN layer will collect a moving average of training batches' means and variances

Algorithm 1 Gradient Augmentation (GradAug)

Input: Network Net. Training image img. Random transformation T. Number of sub-networks n. Sub-network width lower bound α.
▷ Train full-network.
Forward pass, $output_f = Net(img)$
Compute loss, $loss_f = criterion(output, target)$
▷ Regularize sub-networks.
for i in $range(n)$ **do**
  Sample a sub-network, $subnet_i = Sample(Net, \alpha)$
  Fix BN layer's mean and variance, $subnet_i.track\_running\_stats = False$
  Forward pass with transformed images, $output_i = subnet_i(T^i(img))$
  Compute loss with soft labels, $loss_i = criterion(output_i, output_f)$
end for
Compute total loss, $L = loss_f + \sum_{i=1}^{n} loss_i$
Compute gradients and do backward pass

during training. The collected mean and variance will be used during inference. However, the batch mean and variance in the sub-networks can be very different from those in the full-network because the training samples are randomly transformed. This will cause the final BN mean and variance to be inappropriate for the full-network during inference. But in the training phase, BN uses the mean and variance of the current batch, so the training behaves normally. To obtain the correct BN statistics for the full-network, *we do not update BN mean and variance when training the sub-networks*. Only the full-network is allowed to collect these statistics. However, the weights in BN layer are still updated by sub-networks because they can be shared with full-network. To further improve the performance, we also leverage two training tricks in [189]. First, we use the output of the full-network as soft labels to train the sub-networks. Second, we always sample the smallest sub-network (i.e., $w = \alpha$) during training if $n > 1$. The Pytorch-style pseudo-code of GradAug is presented in Algorithm 1.

5.1.2 Analysis of Gradient Property

We provide an in-depth analysis of GradAug from the perspective of gradient flow. For simplicity, we consider a fully connected network with 1-D training samples. We define the network as N. The parameter of one layer in the full-network is $\theta \in \mathbb{R}^{c_1 \times c_2}$. The parameter of sub-networks is θ_w as explained in Section 5.1.1. $x \in \mathbb{R}^d$ is the training sample and y is its label. The output of the network is denoted as $N(\theta, x)$, and the training loss is $l(N(\theta, x), y)$ where l is the loss function, which is often the cross entropy in image classification. The loss and gradients in a standard training process are computed as

$$L_{std} = l(N(\theta, x), y), \quad g_{std} = \frac{\partial L_{std}}{\partial \theta}, \tag{5.1}$$

where $g_{std} \in \mathbb{R}^{c_1 \times c_2}$. Structure regularization methods [142, 75, 89] randomly drop some connections in the network, and their loss and gradients can be computed as

$$L_{sr} = l(N(\theta_{rand}, x), y), \quad g_{sr} = \frac{\partial L_{sr}}{\partial \theta_{rand}}. \tag{5.2}$$

We can view g_{sr} has the same shape as g_{std} where the gradients of disabled connections are 0. Therefore, we can rewrite g_{sr} as

$$g_{sr} = g_{std} + g_{noise}, \tag{5.3}$$

where $g_{noise} \in \mathbb{R}^{c_1 \times c_2}$ is a random matrix which introduces some random disturbances to the gradients. In contrast, GradAug applies more meaningful disturbances to the gradients. Let T be the random transformation operation (e.g., random scale, random rotation, etc.) and T^i be the

transformation to sub-network i ($i = [1, ..., n]$). The loss and gradients are computed as:

$$
\begin{aligned}
L_{GA} &= l(N(\theta, x), y) + \sum_{i=1}^{n} l(N(\theta_{w_i}, T^i(x)), N(\theta, x)) \\
g_{GA} &= \frac{\partial l(N(\theta, x), y)}{\partial \theta} + \sum_{i=1}^{n} \frac{\partial l(N(\theta_{w_i}, T^i(x)), N(\theta, x))}{\partial \theta_{w_i}} = g_{std} + g'.
\end{aligned}
\tag{5.4}
$$

g_{GA} has a similar form with g_{sr}. The first term is the same as the gradients in standard training. *But the second term g' is derived by the sub-networks with transformed training samples. Since sub-networks are part of the full-network, we call this term "self-guided".* It reinforces good descent directions, leading to improved performance and faster convergence. g' can be viewed as an augmentation to the raw gradients g_{std}. It allows different parts of the network to learn diverse representations.

The gradients of data-level regularization methods are similar to g_{std}, with the difference only in the training sample. The gradients are

$$
g_{dr} = \frac{\partial l(N(\theta, f(x)), y)}{\partial \theta}, \tag{5.5}
$$

where f is the augmentation method such as CutMix. GradAug can also leverage these augmentations by applying them to the original samples and then following random transformations. The gradients become

$$
g_{GA} = \frac{\partial l(N(\theta, f(x)), y)}{\partial \theta} + \sum_{i=1}^{n} \frac{\partial l(N(\theta_{w_i}, T^i(f(x))), N(\theta, f(x)))}{\partial \theta_{w_i}} = g_{dr} + g'. \tag{5.6}
$$

g' is still an augmentation to g_{dr}. Data augmentation can also be combined with other structure regularization methods. However, similar to the derivations in Eq. 5.2 and Eq. 5.3, such combination strategy introduces random noises to g_{dr}, which is not as effective as GradAug as shown in Table 5.3.

5.2 Experiments

We first evaluate the effectiveness of GradAug on image classification. Next, we show the generalization ability of GradAug on object detection and instance segmentation. Finally, we demonstrate that GradAug can improve the model's robustness to image distortions and adversarial attacks. We also show GradAug is effective in low data settings and can be extended to semi-supervised learning.

5.2.1 ImageNet Classification

Implementation details. ImageNet [32] dataset contains 1.2 million training images and 50,000 validation images in 1000 categories. We follow the same data augmentations in [195] to have a fair comparison. On ResNet-50, we train the model for 120 epochs with a batch size of 512. The initial learning rate is 0.2 with cosine decay schedule. We sample $n = 3$ sub-networks in each training iteration and the width lower bound is $\alpha = 0.9$. For simplicity, we only use random scale transformation for sub-networks. That is the input images are randomly resized to one of $\{224 \times 224, 192 \times 192, 160 \times 160, 128 \times 128\}$. Note that we report the final-epoch accuracy rather than the highest accuracy in the whole training process as is reported in CutMix[195].

We evaluate GradAug and several popular regularization methods on the widely used ResNet-50 [64]. The results are shown in Table 5.1. GradAug achieves a new state-of-the-art performance of 78.79% based on ResNet-50. Specifically, GradAug significantly outperforms the structure regularization methods by more than 1 point. As illustrated in Eq. 5.3 and Eq. 5.4, GradAug has a similar form with structure regularization. The difference is that GradAug introduces self-guided disturbances to augment the raw gradients. The large improvement over the structure regularization methods clearly validates the effectiveness of our proposed method.

As shown in Eq. 5.6, GradAug can be seamlessly combined with data augmentation. We combine GradAug with CutMix (p=0.5) and denote this method as GradAug†. We compare GradAug† with bag of tricks [65] at the bottom of Table 5.1. It is evident that GradAug† outperforms bag of tricks both in model complexity and accuracy. Note that bag of tricks includes a host of advanced techniques such as model tweaks, training refinements, label smoothing, knowledge distillation, Mixup augmentation, etc., while GradAug is as easy as regular model training.

Due to the sub-networks in GradAug training, one natural question arises: *Would the training cost of GradAug increase significantly?* As stated in [195], typical regularization methods [195, 200, 49] require more training epochs to converge, while GradAug converges with less epochs. Thus the total training time is comparable. The memory cost is also comparable because sub-networks do forward and back-propagation one by one, and only their gradients are accumulated to update the weights. Table 5.2 shows the comparison on ImageNet. The training cost is measured on an 8$\times$ 1080Ti GPU server with a batch size of 512. We can see that the training time of GradAug is comparable with state-of-the-art regularization methods such as CutMix.

5.2.2 CIFAR Classification

Implementation details. We also evaluate GradAug on CIFAR-100 dataset [86]. The dataset has 50,000 images for training and 10,000 images for testing in 100 categories. We choose WideResNet [197] and PyramidNet [56] structures as they achieve state-of-the-art performance on CIFAR dataset. We follow the training setting in [197, 56] in our experiments. For WideResNet, we train the model for 200 epochs with a batch size of 128. The initial learning rate is 0.1 with cosine decay schedule. Weight decay is 0.0005. PyramidNet is trained for 300 epochs with a batch size of 64. The initial learning rate is 0.25 and decays by a factor of 0.1 at 150 and 225 epochs. Weight decay is 0.0001. We use random scale transformation where input images are resized to one of

Table 5.1: ImageNet classification accuracy of different techniques on ResNet-50 backbone.

Model	FLOPs	Accuracy	
		Top-1 (%)	Top-5 (%)
ResNet-50 [64]	4.1 G	76.32	92.95
ResNet-50 + Cutout [35]	4.1 G	77.07	93.34
ResNet-50 + Dropblock [49]	4.1 G	78.13	94.02
ResNet-50 + Mixup [200]	4.1 G	77.9	93.9
ResNet-50 + CutMix [195]	4.1 G	78.60	94.08
ResNet-50 + StochDepth [75]	4.1 G	77.53	93.73
ResNet-50 + Droppath [89]	4.1 G	77.10	93.50
ResNet-50 + ShakeDrop [182]	4.1 G	77.5	-
ResNet-50 + GradAug (Ours)	4.1 G	**78.79**	**94.38**
ResNet-50 + bag of tricks [65]	4.3 G	79.29	94.63
ResNet-50 + GradAug† (Ours)	**4.1 G**	**79.67**	**94.93**

$\{32 \times 32, 28 \times 28, 24 \times 24\}$. The number of sub-networks is $n = 3$ and the width lower bound is $\alpha = 0.8$.

The results are compared in Table 5.3. GradAug is comparable with the state-of-the-art CutMix, and it clearly outperforms the best structure regularization method ShakeDrop, which validate the effectiveness of the self-guided augmentation to the raw gradients. We further illustrate this by comparing GradAug† with CutMix + ShakeDrop. On WideResNet, ShakeDrop severely degrades the Top-1 accuracy of CutMix by 2.44%, while GradAug consistently improves CutMix by more than 1 point. The reason is that ShakeDrop introduces random noises to the training process, which is unstable and ineffective in some cases. However, GradAug is a self-guided augmentation to the gradients, which makes it compatible with various structures and data augmentations.

Table 5.2: Training cost of state-of-the-art regularization methods on ImageNet. Our method requires less number of epochs to converge. Therefore, the total training cost is comparable with other methods.

ResNet-50	#Epochs	Mem (MB)	Mins/epoch	Total hours	Top-1 Acc (%)
Baseline [200]	90	6973	22	**33**	76.5
Baseline [200]	200	6973	22	73	76.4
Mixup [200]	90	6973	23	35	76.7
Mixup [200]	200	6973	23	77	77.9
Dropblock [49]	270	6973	23	104	78.1
CutMix [195]	300	6973	23	115	78.6
GradAug	120	7145	61	122	**78.8**
GradAug	200	7145	61	203	**78.8**

Table 5.3: CIFAR-100 classification accuracy of different techniques on WideResNet and PyramidNet.

Model	WideResNet-28-10		PyramidNet-200 ($\tilde{\alpha} = 240$)	
	Top-1 Acc (%)	Top-5 Acc (%)	Top-1 Acc (%)	Top-5 Acc (%)
Baseline [64]	81.53	95.59	83.49	94.31
+ Mixup [200]	82.5	-	84.37	96.01
+ CutMix [195]	**84.08**	**96.28**	84.83	86.73
+ ShakeDrop [182]	81.65	96.19	84.57	97.08
+ GradAug (Ours)	83.98	96.17	**84.98**	**97.08**
+ CutMix + ShakeDrop	81.64	96.46	85.93	**97.63**
+ GradAug† (Ours)	**85.25**	**96.85**	**86.24**	97.33

5.2.3 Ablation Study

We study the contribution of random width sampling and random transformation to the performance, respectively. We also show the impact of the number of sub-networks n and the width

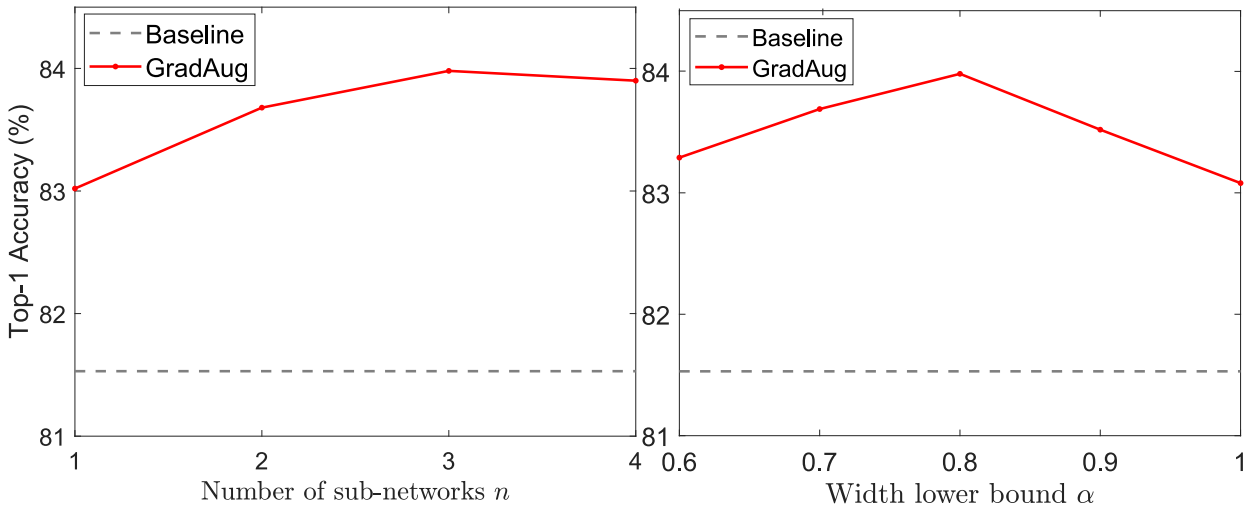


Figure 5.2: Effect of number of sub-networks and width lower bound. Sampling more number of sub-networks tend to yield better performance, but it plateaus after 3 sub-networks.

lower bound α. The experiments are conducted on CIFAR-100 based on the WideResNet-28-10 backbone.

Random width sampling and random transformation. We study the effect of one component by abandoning the other one. First, we do not randomly sample sub-networks. Then GradAug becomes multi-scale training in our experiments. In each iteration, we feed different scaled images to the network. Second, we do not conduct random scale transformation. In each iteration, we sample 3 sub-networks and feed them with the original images. The results are shown in Table 5.4. Random scale and random width sampling only achieve marginal improvements over the baseline, but GradAug remarkably enhances the baseline (+2.43%). This reaffirms the effectiveness of our method, which unifies data augmentation and structure regularization in the same framework for better performance.

Number of sub-networks and width lower bound. There are two hyperparameters in GradAug, the number of sub-networks n and sub-network width lower bound α. We first explore the effect

Table 5.4: Contribution of random width sampling and random scale on CIFAR-100. RandomScale and RandWidth can both slightly improve the model performance, while GradAug achieves significant improvements.

WideResNet-28-10	Top-1 Acc	Top-5 Acc
Baseilne	81.53	95.59
RandScale	82.27	96.16
RandWidth	81.74	95.56
GradAug	83.98	96.17

of n. Other settings are the same as Section 5.2.2. The results are shown in Figure 5.2. A larger n tends to achieve higher performance since it involves more self-guided gradient augmentations. The accuracy plateaus when $n \geq 3$. Note that even one sub-network can significantly improve the baseline. Then we investigate the impact of width lower bound α by fixing other settings. As shown in Figure 5.2, $\alpha = 0.8$ achieves the best accuracy, but all the values clearly outperform the baseline. GradAug is not sensitive to these hyperparameters. Empirically, we can set $n \geq 3$ and $\alpha \in [0.7, 0.9]$.

Effect of different transformations. As shown in experiments above, GradAug is very effective when leveraging random scale transformation and CutMix. Here we further explore other transformations, including random rotation transformation and the combination of random scale and rotation transformations. We conduct the experiments on WideResNet-28-10 and ResNet-50 following the settings above. For random rotation, we randomly rotate the images by a degree of $\{0^\circ, 90^\circ, 180^\circ, 270^\circ\}$. For the combination, the input images are first randomly rotated and then randomly resized. The results are shown in Table 5.5. It is clear that both transformations (random scale and random rotation) and their combination achieve significant improvements over the baseline. This validates our idea of regularizing sub-networks by different transformed images.

Table 5.5: Top-1 Accuracy (%) of WideResNet-28-10 on CIFAR-100 and ResNet-50 on ImageNet using different image transformations.

Model	WideResNet	ResNet
Baseline	81.53	76.32
RandScale	83.98	78.79
RandRot	83.36	77.62
Scale&Rot	84.21	78.66

Table 5.6: Sampling sub-networks by random depth in GradAug. We list the top-1 accuracy reported in the paper and our re-implementation. Random depth sampling in GradAug still significantly improves the performance.

ResNet-110	CIFAR-10		CIFAR-100	
	Reported	Reimpl.	Reported	Reimpl.
Baseline	93.59	93.49	72.24	72.21
StochDepth [75]	94.75	94.29	75.02	75.20
GradAug	-	**94.85**	-	**77.01**

Generating sub-networks by stochastic depth. In the experiments above, we generate sub-networks by cutting the network width. Similarly, we can generate sub-network by shrinking the network depth. We follow StochDepth [75] to randomly drop some layers during training. The training settings are the same as [75] and we use random scale transformation to regularize sub-networks. As shown in Table 5.6, GradAug significantly outperforms the baseline and StochDepth. This demonstrates that GradAug can be generalized to depth-shortened sub-networks and again verifies the effectiveness of our idea.

Table 5.7: COCO object detection and instance segmentation based on Mask-RCNN-FPN. GradAug shows better performance than state-of-the-art regularization methods.

Model	ImageNet Cls Acc (%)	Det mAP	Seg mAP
ResNet-50 (Baseline)	76.3 (+0.0)	36.5 (+0.0)	33.3 (+0.0)
Mixup-pretrained	77.9 (+1.6)	35.9 (-0.6)	32.7 (-0.6)
CutMix-pretrained	78.6 (+2.3)	36.7 (+0.2)	33.4 (+0.1)
GradAug-pretrained	**78.8** (**+2.5**)	**37.7** (**+1.2**)	**34.5** (**+1.2**)
GradAug	**78.8** (**+2.5**)	**38.2** (**+1.7**)	**35.4** (**+2.1**)

5.2.4 Object Detection and Instance Segmentation

To evaluate the generalization ability of the learned representations by GradAug, we finetune its ImageNet pretrained model for COCO [101] object detection and instance segmentation. The experiments are based on Mask-RCNN-FPN [63, 100] framework and MMDetection toolbox [17] on ResNet-50 backbone. Mixup and CutMix, two most effective methods in image classification, are employed for comparison. As explained in Section 1.2, Mixup and CutMix are mixed sample data augmentation methods, which can not be applied to object detection and segmentation. Therefore, we compare these methods by directly finetuning their ImageNet pretrained models on COCO dataset. All models are trained with $1\times$ schedule on COCO dataset. The image resolution is 1000×600. The mean Average Precision (AP at IoU=0.50:0.05:0.95) is reported in Table 5.7. We can see that although Mixup and CutMix achieve large improvements on ImageNet classification, the learned representations can barely benefit object detection and segmentation. In contrast, GradAug-pretrained model considerably improves the performance of Mask-RCNN. This validates that GradAug enables the model to learn well-generalized representations which transfer well to other tasks.

Figure 5.3: Object detection and instance segmentation visual examples. GradAug is more robust to small scale and large scale objects.

Moreover, the training procedure of GradAug can be directly applied to the detection framework. The result (last line of Table 5.7) shows that it further boosts the performance as compared with GradAug-pretrained and can significantly improve the baseline by +1.7 det mAP and +2.1 seg mAP. We also show some visual results of object detection and instance segmentation by using the baseline model MaskRCNN-R50 and the model trained by GradAug. As depicted in Figure 5.3, GradAug-trained model can effectively detect small scale and large scale objects. It is also robust to occlusions. This indicates that GradAug helps the model learn well-generalized representations.

5.2.5 Model Robustness

Deep neural networks are easily fooled by unrecognizable changes on input images. Developing robust machine learning models is pivotal for safety-critical applications. In this section, we evaluate the model robustness to two kinds of permutations, image corruptions and adversarial attacks.

Image corruption. ImageNet-C dataset [66] is created by introducing a set of 75 common visual corruptions to ImageNet classification. ImageNet-C has 15 types of corruptions drawn from four categories (noise, blur, weather and digital). Each type of corruption has 5 levels of severity.

Table 5.8: Corruption error of ResNet-50 trained by different methods. **The lower the better**.

Model	Clean Err	Noise			Blur				Weather				Digital				mCE
		Gauss	Shot	Impulse	Defocus	Glass	Motion	Zoom	Snow	Frost	Fog	Bright	Contrast	Elastic	Pixel	JPEG	
ResNet-50	23.7	72	75	76	77	91	82	81	78	76	65	59	65	89	72	75	75.5
+ Cutout	22.9	72	74	77	77	91	80	80	77	77	65	58	64	89	75	76	75.5
+ Mixup	22.1	68	72	72	**75**	**88**	**75**	**74**	70	70	55	55	61	**85**	65	72	70.5
+ CutMix	21.4	72	74	76	77	91	78	78	77	75	62	56	65	87	77	74	74.6
+ GradAug	21.2	72	72	79	78	90	80	80	73	72	61	55	64	87	**64**	71	73.2
+ GradAug†	**20.4**	71	73	78	76	91	78	77	72	71	61	**53**	63	86	76	**69**	73.0
+ GradAug*	21.9	**62**	**65**	**63**	77	90	79	75	**64**	**57**	**50**	54	**52**	87	77	75	**68.5**

Corruptions are applied to validation set only. Models trained on clean ImageNet should be tested on the corrupted validation set without retraining. We follow the evaluation metrics in [66] to test ResNet-50 trained by different regularization methods. The mean corruption error (mCE) is reported in Table 5.8. Mixup has lower mCE than other methods. We conjecture the reason is that Mixup proportionally combines two samples, which is in a similar manner to the generation of corrupted images. GradAug outperforms the second best competing method CutMix by 1.4%. Note that GradAug can also be combined with Mixup and we denote it as GradAug*. The results in Table 5.8 reveal that GradAug* further improves Mixup and achieves the lowest mCE. This demonstrates that GradAug is capable of leveraging the advantages of different augmentations.

Adversarial attack. We also evaluate model robustness to adversarial samples. Different from image corruption, adversarial attack uses a small distortion which is carefully crafted to confuse a classifier. We use Fast Gradient Sign Method (FGSM) [51] to generate adversarial distortions and conduct white-box attack to ResNet-50 trained by different methods. The classification accuracy on adversarially attacked ImageNet validation set is reported in Table 5.9. Note that here Mixup is not as robust as to image corruptions, which validates our aforementioned conjecture in the image corruption experiment. GradAug and CutMix are comparable and both significantly outperform other methods. GradAug† further gains improvements over GradAug and CutMix, manifesting superiority of our self-guided gradient augmentation.

Table 5.9: ImageNet Top-1 accuracy after FGSM adversarial attack. ε is the attack severity. GradAug shows better robustness than other regularization methods

Model	$\varepsilon = 0.05$	$\varepsilon = 0.10$	$\varepsilon = 0.15$	$\varepsilon = 0.20$	$\varepsilon = 0.25$
ResNet-50	27.90	22.65	19.50	17.04	15.09
+ Cutout	27.22	21.55	17.51	14.68	12.37
+ Mixup	30.76	25.59	21.63	18.44	16.19
+ CutMix	37.73	33.42	29.69	26.29	23.26
+ GradAug	36.51	31.44	27.70	24.93	22.33
+ GradAug†	**40.26**	**35.18**	**31.36**	**28.04**	**25.12**

Table 5.10: Top-1 accuracy on CIFAR-10 and STL-10 with limited labels. GradAug shows larger advantages over other method when labeled data is limited.

Model	CIFAR-10			STL-10
	250	1000	4000	1000
WideResNet-28-2	45.23±1.01	64.72±1.18	80.17±0.68	67.62±1.06
+ CutMix (p=0.5)	43.45±1.98	63.21±0.73	80.28±0.26	67.91±1.15
+ CutMix (p=0.1)	43.98±1.15	64.60±0.86	82.14±0.65	69.34±0.70
+ ShakeDrop	42.01±1.94	63.11±1.22	79.62±0.77	66.51±0.99
+ GradAug	**50.11±1.21**	**70.39±0.82**	**83.69±0.51**	**70.42±0.81**
+ GradAug-semi	**52.95±2.15**	**71.74±0.77**	84.11±0.25	**70.86±0.71**
Mean Teacher [154]	48.41±1.01	65.57±0.83	**84.13±0.28**	-

5.2.6 Low Data Setting

Deep neural network models suffer from more severe over-fitting when there is only limited amount of training data. Thus we expect regularization methods to show its superiority in low data setting. However, we find that state-of-the-art methods are not as effective as supposed. For a fair comparison, we follow the same hyperparameter settings in [120]. The backbone network is WideResNet-

28-2. We first evaluate different methods on CIFAR-10 with 250, 1000 and 4000 labels. Training images are sampled uniformly from 10 categories. We run each model on 5 random data splits and report the mean and standard deviation in Table 5.10. We observe that CutMix (p=0.5) and ShakeDrop even degrade the baseline model performance, especially when labels are very limited. CutMix mixes images and their labels, which introduces strong noises to the data and ground truth labels. This is effective when there is enough clean labels to learn a good baseline. But when the baseline is weak, this disturbance is too severe. We reduce the impact of CutMix by setting p=0.1, where CutMix is barely used during training. CutMix still harms the baseline when there are only 250 labels, but it becomes beneficial when there are 4000 labels. ShakeDrop has a similar trend with CutMix since it introduces noises to the structure. In contrast, GradAug significantly and consistently enhances the baseline in all cases because it generates self-guided augmentations to the baseline rather than noises. Moreover, GradAug can be easily extended to semi-supervised learning (SSL). We can leverage the full-network to generate labels for unlabeled data and use them to train the sub-networks. Our GradAug-semi can further improve the performance over GradAug. It even achieves comparable performance with Mean Teacher [154], which is a popular SSL algorithm. We also evaluate the methods on STL-10 dataset [30]. The dataset is designed to test SSL algorithms, where the unlabeled data are sampled from a different distribution than labeled data. Similarly, CutMix and ShakeDrop are not effective while GradAug and GradAug-semi achieve clear improvements.

5.3 Summary

In summary, we propose GradAug which introduces self-guided augmentations to the network gradients during training. The method is easy to implement while being effective. It achieves a new state-of-the-art accuracy on ImageNet classification. The generalization ability is verified on

COCO object detection and instance segmentation. GradAug is also robust to image corruption and adversarial attack. We further reveal that current state-of-the-art methods do not perform well in low data setting, while GradAug consistently enhances the baseline in all cases.

CHAPTER 6: EFFICIENT NEURAL ARCHITECTURE SEARCH VIA SHORT TRAINING

The work in this Chapter has been published in the following paper:

Taojiannan Yang, Linjie Yang, Xiaojie Jin and Chen Chen.. "Revisiting Training-free NAS Metrics: An Efficient Training-based Method." Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. 2023.

Recent neural architecture search (NAS) works proposed training-free metrics to rank networks which largely reduced the search cost in NAS. However, after an in-depth evaluation of these training-free metrics, we find that: (1) the number of parameters (#Param), which is the most straightforward training-free metric, is overlooked in previous works but is surprisingly effective, (2) recent training-free metrics largely rely on the #Param information to rank networks. Our experiments show that the performance of recent training-free metrics drops dramatically when the #Param information is not available. Motivated by these observations, we argue that metrics less correlated with the #Param are desired to provide additional information for NAS. We propose a light-weight training-based metric which has a weak correlation with the #Param while achieving better performance than training-free metrics at a lower search cost.

This chapter is organized as follows: In Section 6.1, we first revisit several existing training-free metrics and #Param. We demonstrate that #Param is an effective search metric on NAS-Bench-101 and NAS-Bench-201, and that existing training-free metrics rely on #Param to achieve high performance. Then we introduce our light-weight training-based metric and short-training strategy in Section 6.2.1 and Section 6.2.2, respectively. Section 6.5 provides a summary for this chapter.

6.1 Revisiting Training-free Metrics

The number of **linear regions (LR)** is used in [113, 22] to rank networks at initialization. Linear region is a well-studied theoretical criteria [180, 117] to indicate the learning capacity of a network. It is defined as how many regions a network could split the input space into. A larger number of linear regions indicates that the network has higher performance. The number of LR is estimated differently in TE-NAS [22] and NASWOT [113]. TE-NAS calculates LR by forwarding a batch of samples to the network and count how many samples have different activation patterns, while NASWOT feeds a batch of samples to the network and compute the Hamming distance between different activation patterns. The Hamming distance between these activation patterns is used to define a kernel matrix $\mathbf{K}$. The ranking metric is defined as the determinant of $\mathbf{K}$. To distinguish these two metrics, we denote the LR estimated by TE-NAS and NASWOT as LR1 and LR2, respectively.

TE-NAS further leverages the **neural tangent kernel (NTK)** to score networks. [90, 177] point out that the network's convergence speed is determined by the condition number of NTK. Intuitively, a faster convergence speed indicates that the network has a higher performance. So the condition number of NTK can be used to rank networks. Note that in [22], NTK is negtively correlated with the accuracy while in this paper we use negative NTK to make it positive.

These theoretical indicators describe a network's property from different perspectives. However, the most naive indicator to describe a network would be the **number of parameters (#Param)**. Intuitively, a larger model tends to have better performance. This makes us wonder whether the number of parameters is a good training-free metric? The answer is yes. In Table 6.1, we show the comparison of #Param and training-free metrics on NAS-Bench-101 [188] and NAS-Bench-201 [37]. We evaluate these metrics based on random search. Specifically, we randomly sample 100 networks from the search space and use the metrics to select the best one. We run each experiment 5

Table 6.1: Comparison of #Param and training-free metrics on NAS-Bench-101 and NAS-Bench-201. Each experiment is repeated 5 times and mean accuracy and standard deviation are reported. #Param is comparable with training-free metrics.

Metrics	NAS-Bench-101	NAS-Bench-201		
	CIFAR-10	CIFAR-10	CIFAR-100	ImageNet16-120
#Param	92.6(1.3)	**93.2(0.5)**	**70.1(0.8)**	41.6(4.1)
LR1	91.6(0.9)	92.3(1.1)	66.2(5.0)	43.1(2.5)
NTK	91.2(0.9)	91.9(1.7)	66.6(4.3)	41.4(4.9)
LR2	**92.8(1.2)**	92.6(0.9)	69.3(1.4)	**43.3(2.9)**

Table 6.2: Correlation (Kendall's Tau) of different training-free metrics with the number of parameters (#Param). Training-free metrics have a high correlation with #Param.

Correlation with #Param	LR1	NTK	LR2
NAS-Bench-101	0.46	0.36	0.62
NAS-Bench-201	0.39	0.30	0.56

times and report mean accuracy and standard deviation. Surprisingly, the results show that #Param achieves comparable performance with other training-free metrics on different datasets.

The good performance of #Param further motivates us to investigate whether these training-free metrics are correlated with #Param. We compute the Kendall rank correlation coefficient (Kendall's Tau) [83] between different training-free metrics and #Param on NAS-Bench-101 (10000 networks) and NAS-Bench-201 (15625 networks) in Table 6.2. As a reference, the correlation between LR1 and LR2 is 0.56 on NAS-Bench-201. Note that they are the same metric just estimated differently, thus a correlation of 0.56 is high. The results show that all these training-free metrics have high correlations with #Param, especially the two linear region metrics. This is intuitively plausible because the number of linear regions is upper bounded by $2^{\#activations}$, while the number of activation units is highly correlated with the number of parameters. These results imply that the ranking ability of these training-free metrics may mainly come from the high correlation with

#Param. In Section 6.3, we validate this conjecture by evaluating training-free metrics on networks of the same number of parameters. Their performance drops dramatically in this situation.

What are the drawbacks of metrics having high correlation with #Param? Firstly, these training-free metrics claim to rank networks by estimating the model's capacity and convergence, but their functionality is in fact similar to #Param while being unnecessarily complicated. Secondly, #Param is not always a good metric. In the scenarios where the #Param is not helpful (*e.g.*, MLP vs. CNN, Residual vs. Plain structure, networks with similar #Param as in Section 6.3), the performance of such metrics will drop dramatically.

Motivated by these observations, we explore a new type of metric in this work, which is weakly correlated with the number of parameters while providing additional information on estimating the performance of the neural networks. Our proposed metric is introduced in the following sections.

6.2 Short-training NAS

6.2.1 Angle Metric

Since existing training-free metrics all have a high correlation with the number of parameters based on the observations in Section 6.1, we shift our attention to the training dynamics. **Angle metric** is a training dynamic which is first proposed in [13] to indicate the generalization ability of a network and later used in [73, 202] as a metric to rank candidate networks in NAS. Considering all the weights of a network as a one-dimensional vector, angle metric is defined as the angle between the weight vectors before and after training. Specifically, let $\boldsymbol{W}_0$ denote the weights of a network

Table 6.3: Comparison of Kendall's Tau of θ_{feat} and θ_{pred} on 50 random networks with different sizes (different #Param) or the same size (same #Param), respectively. θ_{pred} is less affected when #Param information is not available.

Sampled Networks	θ_{feat}	θ_{pred}
diff. #Param	0.37	-0.50
same #Param	-0.09	-0.25

$\boldsymbol{N}$ at initialization, and $\boldsymbol{W}_t$ denote the weights after training. Then the angle metric is defined as

$$\theta(\boldsymbol{N}) = \arccos\left(\frac{\boldsymbol{W_0} \cdot \boldsymbol{W_t}}{\|\boldsymbol{W_0}\|_2 \|\boldsymbol{W_t}\|_2}\right), \tag{6.1}$$

where $\boldsymbol{W_0} \cdot \boldsymbol{W_t}$ is the inner product of $\boldsymbol{W}_0$ and $\boldsymbol{W}_t$. [202] shows that the angle metric is positively correlated with a network's final performance.

However, we find that the angle metric behaves differently at different network stages. Specifically, the angle metric computed with the weights from the feature extraction layers is positively correlated with the network's final accuracy, while the angle metric computed with the weights of the prediction layer (the final fully-connected layer) is negatively correlated with the performance. In most NAS search spaces [188, 37, 106], the feature extraction stage is mainly constructed by a stack of network modules. We denote the angle metric of the feature extraction stage θ_{feat} and the angle metric of the prediction layer θ_{pred} for brevity.

In Tab .6.3, we demonstrate the impact of model parameters on above two variants of angle metrics through two kinds of network settings. We randomly sample 50 networks with different sizes (setting 1) and the same size (setting 2) from NAS-Bench-201, and fully train them on CIFAR-10. Then we compute the Kendall's Tau of θ_{feat} and θ_{pred} for these two scenarios. In setting 1, it shows that θ_{feat} is positively correlated with the accuracy, which is consistent with [73, 202], but θ_{pred} is negatively correlated and has a higher correlation than θ_{feat}. However, in setting 2,

the Kendall's Tau of θ_{feat} degrades dramatically to around 0, which means θ_{feat} fails to rank the networks without the #Param information. But the Kendall's Tau of θ_{pred} degenerates less and is still able to rank the networks of the same number of parameters. Therefore, θ_{pred} is a metric with weak dependency on the number of parameters.

6.2.2 Short-training Scheme

In Section 6.2.1, we show θ_{pred} is a good metric at ranking networks even without the #Param information. However, fully training all candidate networks is too expensive in NAS. To alleviate this problem, we propose an extremely light-weight short-training scheme by using a small proxy dataset from the original target dataset. Specifically, we first randomly sample a sub-set of classes from the target dataset. Then for each sampled class, we randomly sample a small amount of images, generating a highly condensed proxy dataset. We train networks on the proxy dataset for a limited number of iterations. This training procedure is thousands times faster than fully training a network. We find our θ_{pred} metric is effective under such a compact setting in different search spaces and datasets.

Besides θ_{pred}, we also use another training dynamic, the training loss, as an additional metric to evaluate networks. Note that training loss comes for free in our method. In Section 6.3, we show that training loss also has weak correlation with the number of parameters. Combining training loss with θ_{pred} gives richer information on model performance without increasing the computational cost.

Since the scales of θ_{pred} and training loss are different, directly adding their values will cause one dominating the other. To avoid this problem, we first use these two metrics to rank networks respectively. Then we add their ranking index as the final ranking index of each network. Note that both θ_{pred} and training loss are negatively correlated with the model accuracy. For clarity, we take

Algorithm 2 ST-NAS

Input: Number of candidate networks N. Search space $\mathscr{S}$. Target dataset $\mathscr{D}$. Training iterations m.
Output: Model with the highest rank.
▷ Initialization
θ_{pred} = zeros(N), $loss$ = zeros(N)
sampler = RandomSampler()
Sample proxy dataset $\tilde{\mathscr{D}}$ from $\mathscr{D}$
▷ Evaluate candidate networks
for i in $0, 1, ..., N-1$ **do**
  network = sampler($\mathscr{S}$)
  W_0 = network.fc.weights
  Train the network for m iterations with $\tilde{D}$.
  $loss$[i] = - compute_loss(network, $\tilde{\mathscr{D}}$)
  W_t = network.fc.weights
  θ_{pred}[i] = - compute_angle_metric(W_0, W_t)
end for
▷ Combine two metrics
$R_{\theta_{pred}}$ = get_rankings(θ_{pred})
R_{loss} = get_rankings($loss$)
$R = R_{\theta_{pred}} + R_{loss}$
max_idx = model index with the highest rank in R
return: $\mathscr{S}$[max_idx]

the negative value of the two metrics to make them positive in the following experiments. Since the proposed metric employs a short period of training, we name our NAS method combined with this metric as Short-Training NAS (ST-NAS). A pipeline of ST-NAS based on random search is shown in Algorithm 2.

6.3 Empirical Study

As discussed in Section 6.1, recent training-free metrics are highly correlated with the number of parameters, which implies their effectiveness comes from the high correlation with number of parameters. To further validate our claim, we thoroughly evaluate different training-free metrics and our metric on curated search spaces with the same number of parameters. This prevents metrics from leveraging the parameter information to evaluate networks. In the following sections, *Angle*

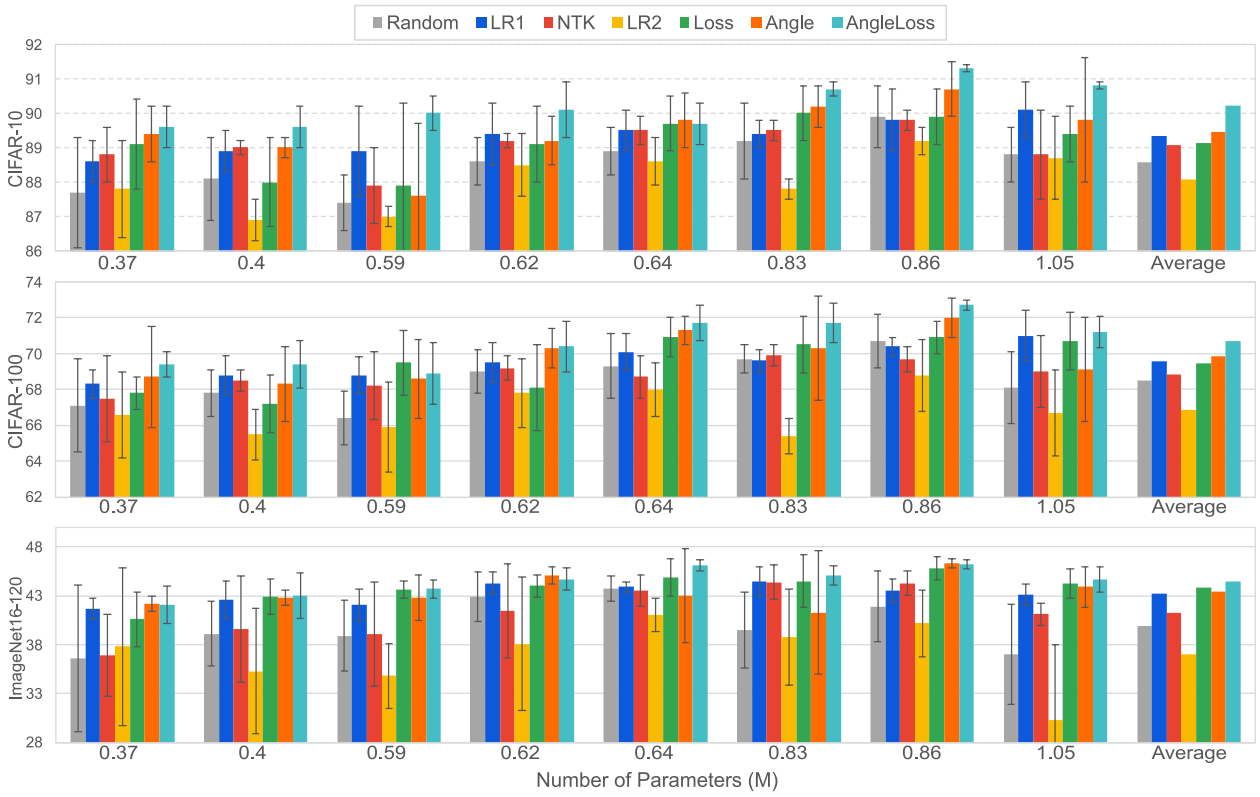


Figure 6.1: Test accuracy (%) of different metrics when evaluated on networks of the same number of parameters. X-axis is the number of parameters (M) in each network group. Each experiment is repeated 5 times and the mean accuracy and standard deviation are reported.

denotes searching with θ_{pred}, *Loss* denotes searching with training loss and *AngleLoss* denotes searching with the combination of the two metrics.

We craft several search spaces based on NAS-Bench-201 [37]. NAS-Bench-201 defines a cell-based search space. Each cell is represented as a densely-connected directed acyclic graph (DAG). Each cell has 4 nodes and 6 edges, where each edge represents an operation. There are 5 candidate operations, including *zeroize, skip-connect,* 1×1 *conv,* 3×3 *conv, and* 3×3 *avg pooling*. Different models may have the same number of parameters but with different structures and performances. We choose 8 groups of models, and models in the same group has the same number of parameters, i.e. $\{0.37, 0.40, 0.59, 0.62, 0.64, 0.83, 0.86, 1.05\}$ M, respectively. The number of networks in each

Table 6.4: Kendall's Tau between our metrics and #Param. Our proposed metrics have less correlation with #Param compared to training-free metrics.

Metrics	Angle	Loss	AngleLoss
Correlation	0.20	-0.11	0.07

group is {1602, 540, 1602, 810, 180, 540, 180, 135}, respectively. We evaluate the effectiveness of different metrics on each of these network groups. We compute the training-free metrics using the settings in the original papers [22, 113]. Our training scheme is detailed in Section 6.2.2. We randomly sample 10 classes and 10 images from each class. The network is trained for 50 iterations with a fixed learning rate of 0.2. Other settings follow those in NAS-Bench-201 [37]. Note this is the default setting throughout our experiments if not specified.

We compare the performance of previous training-free metrics and our metrics using random search. We randomly sample 100 networks from each network group and select the best-performing network per the metric. We also add a baseline which randomly selects a network from candidate networks. Each experiment is repeated 5 times and the mean accuracy and standard deviation are reported. As shown in Figure 6.1, LR2, which has the highest correlation with #Param in Table 6.2 and the best performance in Table 6.1, performs the worst in this scenario. It is even worse than the random baseline. Our *AngleLoss* metric consistently outperforms training-free metrics on all the network groups on three datasets. In most cases, *AngleLoss* is higher than training-free metrics by more than 1%. We also show our metrics' Kendall's Tau with #Param in Table 6.4. As can be seen, the correlations are much lower than that of the training-free metrics in Table 6.2. Above experiment evidences that training-free metrics largely rely on the parameter information to rank networks, and that our metric is advantageous by having weak correlation with the number of parameters, providing additional useful information to estimate a network's performance.

6.4 Experiments

In Section 6.4.1, we first show the comparisons of training-free metrics and our metric on NAS-Bench-101 and NAS-Bench-201. We apply metrics to both random search method and pruning-based search method. Then we compare our metric with other methods on DARTS search space in Section 6.4.2. Finally, we conduct ablation studies to show the impact of short-training hyperparameters.

6.4.1 Results on NAS-Bench-101/201

Random Search. We first evaluate different metrics based on random search. We randomly sample 100 networks from the search space and use different metrics to select the best one. We follow the default settings in [113, 22] to compute training-free metrics LR1, LR2, and NTK. Our training settings are the same as in Section 6.3. We run each experiment 5 times and report the mean accuracy and standard deviation. The search cost is measured on a single GTX-1080Ti GPU.

The results are shown in Table 6.5. We add #Param as a baseline metric in Table 6.5. It is shown that #Param performs well on both NAS-Bench-101 and NAS-Bench-201. It is even slightly better than training-free metrics on CIFAR-10 and CIFAR-100. Note that #Param is very easy to compute, with a search cost of only 3 seconds on 100 networks. The linear region based metrics (LR1 and LR2) are better and more stable than NTK. The performance of NTK is low and has a very large variance. Although both LR1 and LR2 are based on linear regions, LR2 is slightly better and more stable. Note the effectiveness of training-free metrics could be attributed to their high correlation with #Param.

Surprisingly, our metric *AngleLoss* does not perform well on the overall search space of NAS-Bench-201, although we have demonstrated in Section 6.3 that it is significantly better than other

Table 6.5: Comparison of the test accuracy of different metrics on NAS-Bench-101 and NAS-Bench-201 based on random search ($N = 100$). Each experiment is repeated 5 times to compute its mean and standard deviation.

Metrics	Search Cost (s)	NAS-Bench-101	NAS-Bench-201		
		CIFAR-10	CIFAR-10	CIFAR-100	ImageNet16-120
#Param	3	92.58(1.26)	93.21(0.49)	70.15(0.83)	41.58(4.07)
LR1	60	91.98(1.31)	92.30(1.07)	66.23(4.96)	43.12(2.52)
LR1+#Param	60	92.52(1.37)	92.96(0.55)	69.83(0.43)	43.71(2.20)
NTK	181	91.23(1.11)	91.94(1.70)	66.63(4.29)	41.38(4.88)
NTK+#Param	181	91.48(1.52)	93.12(0.48)	69.82(0.73)	42.39(1.61)
LR2	48	91.95(1.16)	92.65(0.93)	69.28(1.40)	43.33(2.91)
LR2+#Param	48	92.58(1.39)	93.33(0.13)	70.10(1.22)	42.83(1.49)
AngleLoss	437	92.86(0.77)	84.65(5.88)	58.06(0.40)	28.08(0.31)
AngleLoss+#Param	437	**93.60(0.46)**	**93.46(0.59)**	**70.58(0.82)**	**43.74(1.48)**
AngleLoss+LR2	462	93.47(0.47)	93.08(0.66)	69.62(0.59)	43.43(1.62)

training-free metrics in different network groups. By visualizing the searched network structures, we find that our *Angle* metric could collapse to some trivial structures, where most of the connections are *zeroize, skip-connect* or *avg_pooling*. Our conjecture is that in these trivial structures, the feature extraction layers are not learning anything meaningful, and the prediction layer is optimized towards random directions in each training iteration. So the weight vector of the prediction layer almost does not change after training, which means *Angle* metric will give a high score to these structures. However, this problem could be easily resolved if we combine our metric with #Param to avoid the structures with a small number of parameters. It can also be avoided when we use a pruning-based search method. In Table 6.5, we see that our metric is significantly boosted by around 10% when combined with #Param, and it achieves higher performance than other training-free metrics. On NAS-Bench-101, we don't have the collapse problem because there are fewer trivial structures. We achieve significantly better performance than training-free metrics.

We also combine training-free metrics with #Param. It shows that these training-free metrics can also slightly benefit from #Param, but the improvement is marginal. Taking #Param as the base-

Table 6.6: Comparison of the test accuracy on NAS-Bench-201 based on pruning-based search in [22]. † indicates the results are reproduced by us using the official released codes [1]. The search cost of our method and TE-NAS is measured on 1080Ti GPU while LGA is measured on Tesla A40 GPU. The **best** and second best results are bold and underlined, respectively.

Method	Search Cost (s)	CIFAR-10	CIFAR-100	ImageNet16-120
RSPS [94]	8007	87.66(1.69)	58.33(4.34)	31.14(3.88)
DARTS (1st) [106]	10889	54.30(0.00)	15.61(0.00)	16.32(0.00)
GDAS [38]	28925	93.61(0.09)	70.70(0.30)	41.84(0.90)
LGA [116]	5400	**93.94(N/A)**	**72.42(N/A)**	**45.17(N/A)**
TE-NAS [22]	1558	93.90(0.47)	71.24(0.56)	42.38(0.46)
TE-NAS† [22]	682	93.20(0.29)	70.44(1.34)	42.34(0.63)
AngleLoss	**508**	93.16(0.37)	70.48(1.04)	43.04(1.82)
AngleLoss+#Param	**508**	93.36(0.26)	70.87(0.41)	43.77(1.33)

line, combined with training-free metrics will even degrade its performance on NAS-Bench-201 CIFAR-10 and CIFAR-100. However, our metric achieves consistent improvements upon #Param on three datasets. We also show that when combined with LR2, AngleLoss+LR2 improves upon LR2 on all datasets. These experiments demonstrate that our metric provides orthogonal information to #Param and training-free metrics. They can be combined together to achieve better performance.

Pruning-based Search. We also apply our metric to pruning-based search used in TE-NAS [22]. All the settings are the same as in Section 6.3, except that we train the supernet for 100 iterations because it takes longer for the supernet to converge. Each experiment is repeated 5 times and the mean and standard deviation are reported.

We compare our method with TE-NAS in Table 6.6. The performances of some other NAS methods are cited from [37] for reference. We report two results for TE-NAS, one is reported in the original paper [22] and the other is reproduced by us using the official codes [1] since we cannot reproduce the results in the original paper using the default setting. The reproduced performance is

Table 6.7: Comparison with state-of-the-art on DARTS CIFAR-10. The **best** and second best results are bold and underlined, respectively.

Method	Search Cost (GPU days)	Params (M)	Top-1 Acc (%)	Search Method
NASNet-A [212]	2000	3.3	97.35	RL
ENAS [124]	0.5	4.6	97.11	RL
AmoebaNet-A [132]	3150	3.2	96.66	evolution
Random baseline [106]	4	3.2	96.71	random
DARTS (1st) [106]	0.4	3.3	97.00	gradient
DARTS (2nd) [106]	1.0	3.3	97.24	gradient
GDAS [38]	0.17	2.5	97.18	gradient
P-DARTS [24]	0.3	3.4	**97.50**	gradient
PC-DARTS [181]	0.1	3.6	97.43	gradient
SDARTS-ADV [23]	1.3	3.3	97.39	gradient
TE-NAS [22]	**0.05**	3.8	97.37	training-free
AngleLoss	0.09	3.2	97.37	short-training
AngleLoss+#Param	0.09	3.2	97.44	short-training

lower while the search cost is also cheaper (we evaluate it on a 1080Ti GPU, which is the same as in TE-NAS). In Table 6.6, we can see that our short-training method is even faster than TE-NAS. This is because TE-NAS needs to compute two metrics (LR1 and NTK), and for each metric it repeats 3 times and takes the average value to have a better and stable performance. However, we only compute our metric once with an extremely short training scheme.

Under the pruning-based search, our metric does not show the collapse problem as in random search. This is because pruning-based method starts from a supernet, which is definitely non-trivial. With a limited number of pruning steps, the network almost never reach a trivial structure with large numbers of empty operations. As shown in Table 6.6, the original results of TE-NAS are better than ours on CIFAR-10 and CIFAR-100, but the search cost is $3\times$ of ours. Our performance is comparable with the reproduced results of TE-NAS at a lower search cost. On ImageNet16-120, our metric is better than TE-NAS in both cases. We also combine our metric with #Param

with negligible additional search cost. It further improves our performance on all three datasets by $0.2\% - 0.7\%$.

Trivial Structures in Random Search. As shown in the Random Search based experiments above, our metric (AngleLoss) does not perform well on the overall search space. The reason is that the *Angle* metric may give a very high score to trivial structures where most of the connections are *zeroize, skip-connect* or *avg_pool*. Our conjecture is that in these trivial structures, the feature extraction layers do not learn anything meaningful, and the prediction layer is optimized towards random directions in each training iteration. So the weight vector of the prediction layer almost does not change after training, which means *Angle* metric will give a high score to these structures. Here we show an example of the trivial structure and how it is resolved when *Angle* metric is combined with #Param. Figure 6.2 shows the rank index and accuracy of different structures on NAS-Bench-201 CIFAR-100. We randomly select 100 networks and rank them using *Angle* metric. The orange dot in Figure 6.2 has the highest score. However, its accuracy is very low. We visualize its cell structure on the right. We find that the orange dot is a trivial structure which is only composed of *zeroize, skip-connect* and *avg_pool*. As explained above, *Angle* metric may give a high score to such structures, therefore the performance on the overall search space is not good and has a very large variance. However, such trivial structures can be easily removed when *Angle* metric is combined with #Param. In Fig. 6.3, we show the rank index of the combined metric on the same set of structures. The trivial structure (orange dot) is now ranked around the middle of all networks because it has a very low rank index in terms of #Param. As a result, #Param could help remove the trivial solution improperly discovered by *Angle* metric.

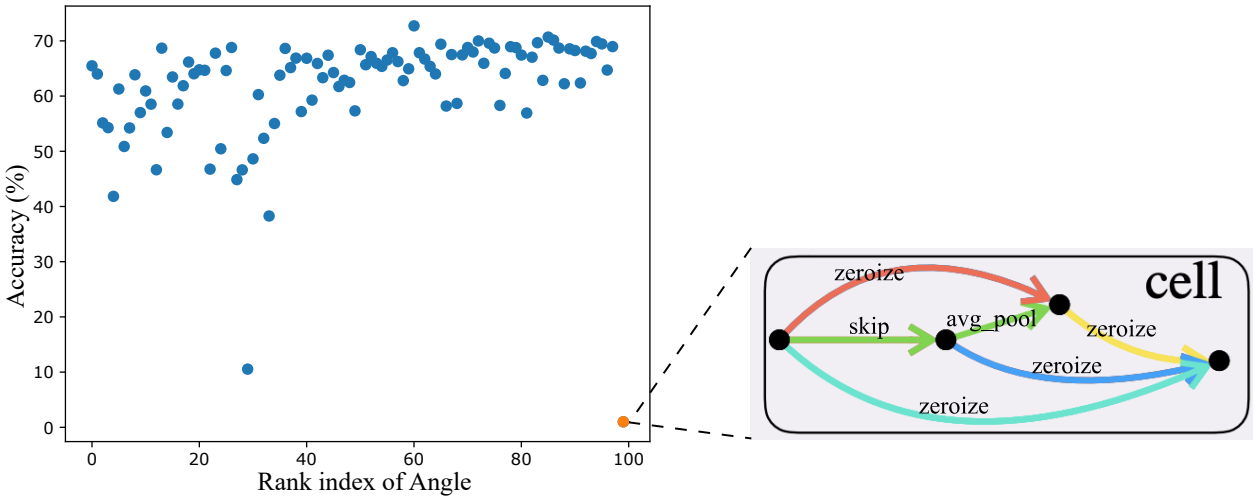


Figure 6.2: Rank index of different structures using *Angle* metric. The orange dot has the highest score. Its structure is visualized on the right. This is based on NAS-Bench-201 CIFAR-100 dataset and random search.

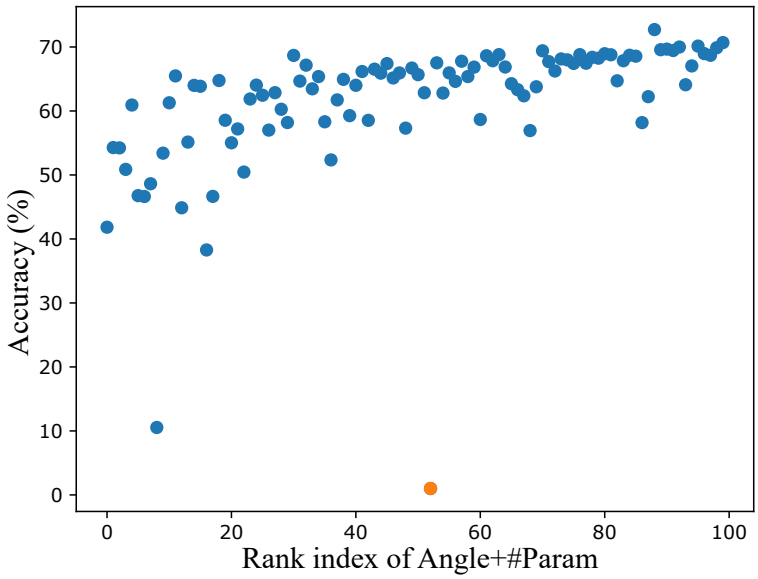


Figure 6.3: Rank index of different structures using *Angle+#Param*. The orange dot is the one that has the highest score in Figure 6.2. Adding #Param help alleviate the effect of trivial structures.

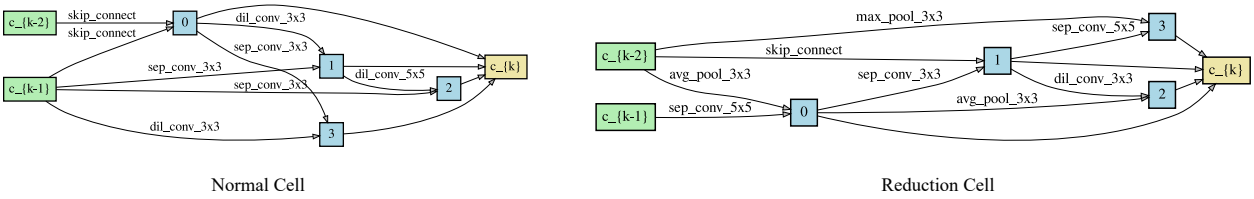


Figure 6.4: Normal and Reduction cells discovered by *AngleLoss* metric on DARTS CIFAR-10.

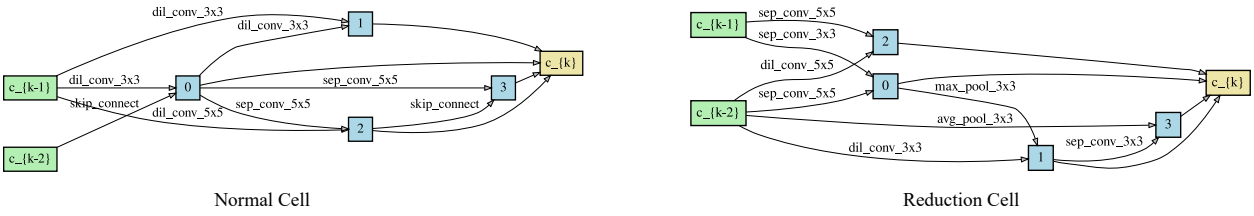


Figure 6.5: Normal and Reduction cells discovered by *AngleLoss+#Param* metric on DARTS CIFAR-10.

6.4.2 Results on DARTS Search Space

We apply our metric to the pruning-based search method used in TE-NAS [22] for the following experiments.

Results on CIFAR-10. We first compare our metric with other methods on CIFAR-10 dataset. As shown in Table 6.7, our metric completes the search process in 0.09 days (*i.e.*, 2.16 hours) on a single 1080Ti GPU. Different from the results on NAS-Bench-201, our search cost is higher than TE-NAS in this case. This is because TE-NAS uses a smaller batch-size to compute NTK on DARTS CIFAR-10, resulting in less computation. Nevertheless, our search cost is still much lower than other NAS methods. Our metric also achieves comparable performance with TE-NAS, but the searched network size is much smaller. When combined with #Param, our metric again achieves a lower test error of 2.56%, which is competitive with state-of-the-art methods. The searched neural network structures by our method are visualized in Figure 6.4 and 6.5.

Table 6.8: Comparison with state-of-the-art NAS methods on DARTS search space ImageNet-1K dataset. Our method outperforms state-of-the-art NAS methods and uses even less search cost than training-free metrics.

Method	Search Cost (GPU days)	Params (M)	Top-1 (%)	Top-5 (%)	Search Method	Search Dataset
NASNet-A [212]	2000	5.3	74.0	91.6	RL	
AmoebaNet-C [132]	3150	6.4	75.7	92.4	evolution	
DARTS (2nd) [106]	4.0	4.7	73.3	91.3	gradient	
GDAS [38]	0.21	5.3	74.0	91.5	gradient	
P-DARTS [24]	0.3	4.9	75.6	92.6	gradient	CIFAR-10
PC-DARTS [181]	0.1	5.3	74.9	92.2	gradient	
TE-NAS [22]	0.05	6.3	73.8	91.7	training-free	
AngleLoss	0.09	4.7	75.3	92.5	short-training	
AngleLoss+#Param	0.09	4.7	74.8	92.3	short-training	
ProxylessNAS [104]	8.3	7.1	75.1	92.5	gradient	
PC-DARTS [181]	3.8	5.3	74.8	92.7	gradient	
TE-NAS [22]	0.17	5.4	75.5	92.5	training-free	ImageNet-1K
AngleLoss	**0.11**	4.8	74.5	91.9	short-training	
AngleLoss+#Param	**0.11**	5.9	**75.9**	**92.9**	short-training	

Table 6.9: Ablation study of different training hyper-parameters on NAS-Bench-201 CIFAR-100.

(a) Number of training iterations.

#Iters	10	25	50	75
Cost (s)	99	230	437	673
Acc (%)	70.22(1.08)	70.33(0.91)	70.58(0.82)	70.37(0.57)

(b) Number of sampled classes.

#Classes	5	10	20
Cost (s)	332	437	641
Acc (%)	70.02(0.74)	70.58(0.82)	70.30(0.74)

(c) Network initialization.

Init.	Kaiming_uniform	Kaiming_normal	Xavier_uniform
Cost (s)	437	437	437
Acc (%)	70.58(0.82)	70.40(0.70)	70.25(1.00)

(d) Number of sampled images.

#Images	5	10	20
Cost (s)	347	437	627
Acc (%)	70.26(1.08)	70.58(0.82)	70.28(0.97)

Results on ImageNet-1K. We compare our metric with state-of-the-art NAS methods on ImageNet-1K [32] in Table 6.8. Our short-training setting is the same as in CIFAR-10. For evaluation, we follow [22] to stack the network with 14 cells and the initial number of channel is 48. In the top half of Table 6.8, the networks are searched on CIFAR-10 and then evaluated on ImageNet-1K. We can see that our metric is competitive with state-of-the-art NAS methods with a much lower

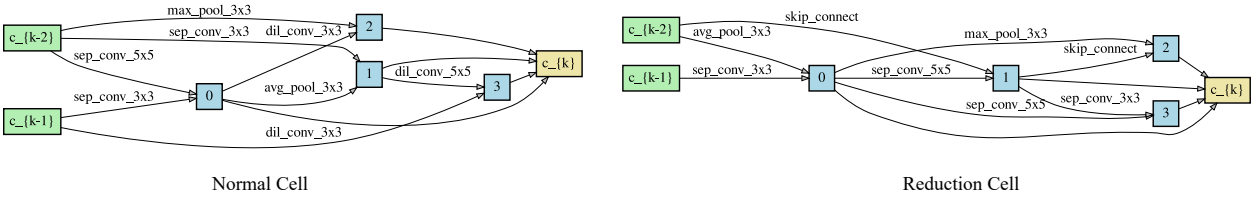


Figure 6.6: Normal and Reduction cells discovered by *AngleLoss* metric on DARTS ImageNet.

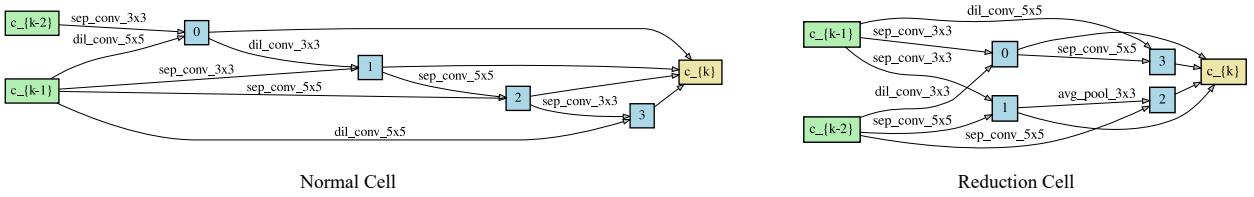


Figure 6.7: Normal and Reduction cells discovered by *AngleLoss+#Param* metric on DARTS ImageNet.

search cost. Compared to TE-NAS, our performance is significantly better and the network size is much smaller. The bottom half of Table 6.8 shows the results with different methods searched directly on ImageNet-1K. Pruning-based search with our metric completes in only 0.11 GPU days (*i.e.*, 2.64 GPU hours), which is even faster than TE-NAS. Our metric is more than 30$\times$ faster than the other NAS methods. The performance of our metric alone is slightly lower than other methods but with a smaller model size. When combined with #Param, the performance of our metric is largely improved and reaches a competitive top-1/top-5 error rate of 24.1%/7.1%, outperforming listed differentiable and training-free methods. Note that our search cost is also significantly lower than other methods. We also visualize the searched architectures in Figure 6.6 and 6.7.

6.4.3 Ablation Study

Here we study the impact of different hyper-parameters in our short-training scheme, including number of training iterations, sampled classes, images per class and weight initialization methods.

We conduct experiments on CIFAR-100. The results of different settings are shown in Table 6.9. We use the random search method in Table 6.4.1 as the baseline. We can see that longer training iterations tend to achieve better performance. This is because longer training iterations allow the network to converge better, which yields more informative angle metric and training loss. But even only 10 training iterations can achieve a decent performance. Increasing the number of classes does not always improve the performance. We speculate that although more classes could provide more information about the target dataset, it also makes the proxy dataset harder, which makes the network harder to converge in the limited iterations and yields less informative angle metric and training loss. Similarly, increasing the number of images does not guarantee better performance either. To achieve the optimal accuracy-efficiency trade-off, one may need to tune the training hyper-parameters. But the performance is not very sensitive to the hyper-parameters and it is feasible to tune hyper-parameters because our method is highly efficient.

6.5 Summary

In summary, We conduct a systematic study to explore the relationship between recent training-free metrics and #Param. Our empirical study shows that recent training-free metrics works similarly to #Param while being unnecessarily complicated. Motivated by this discovery, we propose a light-weight training-based metric which provides orthogonal information than #Param on estimating model performance. Our method achieves competitive performance with state-of-the-art NAS methods, while being even faster than training-free metrics. On the search spaces where the #Param information is not useful, the performance of training-free metrics drops dramatically while our method significantly outperforms them on different datasets. We hope our work could inspire future works to design new metrics which provide more parameter-independent information on estimating the network's performance.

CHAPTER 7: ADAPTING FROZEN IMAGE MODELS FOR EFFICIENT VIDEO UNDERSTANDING

The work in this Chapter has been published in the following paper:

Taojiannan Yang, Yi Zhu, Yusheng Xie, Aston Zhang, Chen Chen and Mu Li.. "AIM: Adapting Image Models for Efficient Video Action Recognition." International Conference on Learning Representations. 2023.

Recent vision transformer based video models mostly follow the "*image pre-training then finetuning*" paradigm and have achieved great success on multiple video benchmarks. However, full finetuning such a video model could be computationally expensive and unnecessary, given the pre-trained image transformer models have demonstrated exceptional transferability. In this chapter, we propose a novel method to Adapt pre-trained Image Models (AIM) for efficient video understanding. By freezing the pre-trained image model and adding a few lightweight Adapters, we introduce spatial adaptation, temporal adaptation and joint adaptation to gradually equip an image model with spatiotemporal reasoning capability. We show that our proposed AIM can achieve competitive or even better performance than prior arts with substantially fewer tunable parameters on four video action recognition benchmarks. Thanks to its simplicity, our method is also generally applicable to different image pre-trained models, which has the potential to leverage more powerful image foundation models in the future.

This chapter is organized as follows: we first briefly describe ViT and video baselines in Section 7.1. Then we introduce spatial adaptation in Section 7.2.1, temporal adaptation in Section 7.2.2 and joint adaptation Section 7.2.3, to show how we adapt a pre-trained image model for effective

video modeling step-by-step. Section 7.3 demonstrates the effectiveness of the proposed method by comprehensive experiments on different datasets. Section 7.4 provides ablation studies and discussions about the proposed method. Section 7.5 summarizes this chapter.

7.1 Preliminary

After the seminal work of Vision Transformer (ViT) [39], transformer-based models have been widely adopted in various computer vision tasks, including video action recognition. In this work, we focus on adapting pre-trained image transformer models and compare them to full finetuned video transformer models, unless otherwise stated.

More specifically, ViT handles an image as a sequence of small patches. Given input image $\boldsymbol{x} \in \mathbb{R}^{H\times W\times C}$, ViT first splits the image to N non-overlapping patches and maps each patch to a D-dim patch embedding via a trainable linear projection [126, 127]. Here, (H,W) is the image resolution and C is the number of channels. Patch embeddings $\boldsymbol{x}_p \in \mathbb{R}^{N\times D}$, where $N = HW/P^2$ and P denotes the patch size. Then a learnable [class] token is prepended to $\boldsymbol{x}_p$ as $\boldsymbol{x}_0 = [\boldsymbol{x}_{class}; \boldsymbol{x}_p] \in \mathbb{R}^{(N+1)\times D}$. To encode positional information, positional embeddings $\boldsymbol{E}_{pos} \in \mathbb{R}^{(N+1)\times D}$ are added to $\boldsymbol{x}_0$ as $\boldsymbol{z}_0 = \boldsymbol{x}_0 + \boldsymbol{E}_{pos}$, where $\boldsymbol{z}_0$ is the final input being fed to a sequence of transformer blocks. Each transformer block is composed of a multiheaded self-attention (MSA) and a MLP layer, together with Layernorm (LN) and skip connections, see Figure 7.1(b). The computation of a standard transformer block can be written as

$$\boldsymbol{z}'_l = \boldsymbol{z}_{l-1} + \mathrm{MSA}(\mathrm{LN}(\boldsymbol{z}_{l-1})), \tag{7.1}$$

$$\boldsymbol{z}_l = \boldsymbol{z}'_l + \mathrm{MLP}(\mathrm{LN}(\boldsymbol{z}'_l)), \tag{7.2}$$

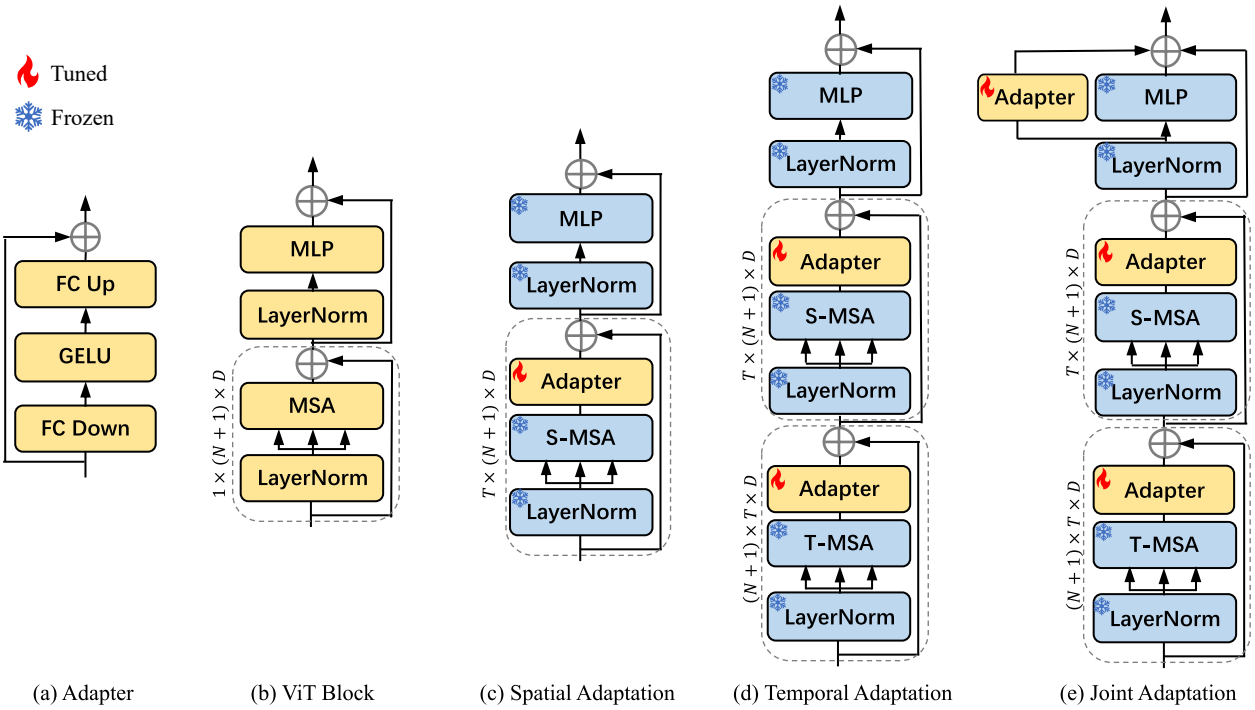
where z_{l-1} and z_l denotes the input and output of the l-th transformer block, respectively. Finally, the learned [class] token x_{class} from the last transformer block is used as global visual representation and fed into a classification head to make the prediction.

Space-only and space-time models for video. A video is a stack of frames with temporal structure. Hence, video understanding requires the model to learn both good appearance representations in each frame (spatial modeling) and also infer the temporal information across frames (temporal modeling). In order to leverage an image transformer model for video tasks, one key thing is how to perform temporal modeling. A simple baseline, termed space-only model, process each video frame independently by an image model. Given $x \in \mathbb{R}^{T\times H\times W\times C}$, where T is the number of frames, space-only model will get T [class] tokens where each [class] token stands for the representation of each frame. These T [class] tokens will be averaged as a way of temporal modeling for final prediction. In order to enhance the capability of temporal modeling, recent works [8, 3, 204] introduce space-time model by adding new temporal modules to image models. These models are now the top performers on most video action recognition benchmarks, however, their training costs are prohibitively high due to full finetuning. Given the increasingly larger and more powerful pre-trained image models, in this work, we study how to efficiently adapt them for video action recognition.

7.2 Methodology

7.2.1 Spatial Adaptation

Since image pre-trained models have been trained on large-scale datasets and demonstrated strong transferability to downstream tasks, we believe they could achieve good spatial modeling in video action recognition with minimal finetuning.



(a) Adapter (b) ViT Block (c) Spatial Adaptation (d) Temporal Adaptation (e) Joint Adaptation

Figure 7.1: We show how we adapt a standard ViT block (b) for video action recognition, by gradually adding spatial adaptation (c), temporal adaptation (d) and joint adaptation (e). Note that S-MSA and T-MSA share weights but are applied to different input dimensions. During training, only newly added Adapters are updated while all the other layers are frozen.

Inspired by efficient finetuning techniques [68, 92, 95, 6] in NLP, we adopt Adapter [68] due to its simplicity. As shown in Figure 7.1(a), Adapter is a bottleneck architecture which consists of two fully connected (FC) layers and an activation layer in the middle. The first FC layer projects the input to a lower dimension and the second FC layer projects it back to the original dimension. To adapt the pre-trained spatial features to target video data, we add an Adapter after the self-attention layer as shown in Figure 7.1(c), which we term as spatial adaptation. During training, all the other layers of the transformer model are frozen while only the Adapters are updated. In Table 7.1, we show that our spatial adaptation strategy achieves comparable performance with the full finetuned space-only baseline. This indicates that spatial adaptation helps the frozen image model to learn good spatial representations from video data. However, there is still a large gap

between the performance of spatial adaptation and a fully finetuned video model because spatial adaptation alone lacks the ability to learn temporal information in videos.

7.2.2 Temporal Adaptation

To capture temporal information more effectively, previous methods usually incorporate new temporal modules to pre-trained image models because it is commonly believed that image models cannot infer temporal structured information in videos. However, adding new temporal modules, either through temporal attention [8, 204] or temporal encoder/decoder [3, 102], will introduce sizable number of extra tunable parameters. In addition, these new modules require full finetuning, which is inefficient.

To address this problem, we present a new strategy: *reuse the pre-trained self-attention layer in the image model to do temporal modeling*. More specifically, we denote the original self-attention layer as S-MSA for spatial modeling, and the reused self-atentnion layer as T-MSA for temporal modeling. As shown in Figure 7.1(d), we put T-MSA in front of S-MSA. Now given the video patch embedding $\boldsymbol{z} \in \mathbb{R}^{T\times(N+1)\times D}$, we first reshape it into $\boldsymbol{z}^T \in \mathbb{R}^{(N+1)\times T\times D}$, where $N = HW/P^2$ is the number of spatial patches and T is the number of frames. Then we feed $\boldsymbol{z}^T$ into the T-MSA where it tries to learn the relationship among the T frames. Note that T-MSA and S-MSA are the same layers (i.e., pre-trained MSA in the image model) and kept frozen during model tuning, but just applied to different input dimensions. This explicit operation helps our model with enhanced temporal modeling, while keeping the number of parameters fixed. In the end, similar to spatial adaptation, we add another Adapter after the reused temporal attention layer to adapt its features on video data, which we term as temporal adaptation. The structure of the Adapter is the same as in spatial adaptation but without the skip connection. The reason is that we want to initialize the adapted model to be close to the original model [68], thus we need to initialize the adapter to zero

and remove the skip connection here to detach the effect of temporal adaptation at the beginning of training. As seen in Table 7.1, temporal adaptation helps to close the gap to full finetuned video models while only introducing another lightweight Adapter into the transformer block.

7.2.3 Joint Adaptation

Spatial and temporal adaptation are performed sequentially to different input dimensions with their individual purposes. It would be desirable to jointly tune the representations for spatiotemporal reasoning. To this end, we further introduce an Adapter in parallel to the MLP layer, which we term as joint adaptation. This Adapter has the same structure as the one in temporal adaptation.

The final structure of a transformer block in our proposed AIM is presented in Figure 7.1(e). The computation of the adapted block can be written as

$$\boldsymbol{z}_l^T = \boldsymbol{z}_{l-1} + \text{Adapter}(\text{T-MSA}(\text{LN}(\boldsymbol{z}_{l-1}))), \tag{7.3}$$

$$\boldsymbol{z}_l^S = \boldsymbol{z}_l^T + \text{Adapter}(\text{S-MSA}(\text{LN}(\boldsymbol{z}_l^T))), \tag{7.4}$$

$$\boldsymbol{z}_l = \boldsymbol{z}_l^S + \text{MLP}(\text{LN}(\boldsymbol{z}_l^S)) + s \cdot \text{Adapter}(\text{LN}(\boldsymbol{z}_l^S)), \tag{7.5}$$

where $\boldsymbol{z}_l^T$, $\boldsymbol{z}_l^S$, $\boldsymbol{z}_l$ denotes the temporal adapted, spatial adapted, and jointly adapted output in the l-th transformer block, respectively. Here, s is a scaling factor to control the weight of the output from Adapter. For the final prediction, we simply take the average of the [class] tokens of each input frame and feed it to the classification head.

7.3 Experiments

Datasets. We evaluate the proposed method on four widely adopted video action recognition benchmarks, Kinetics-400 (K400) [82], Kinetics-700 (K700) [14], Something-something-v2 (SSv2) [53] and Diving-48 [96]. K400 contains around 240K training videos and 20K validation videos in 400 human action classes. The videos are all trimmed to around 10 seconds. K700 is an extended version of K400 which contains around 530K training videos and 34K validation videos in 700 classes. SSv2 contains 168.9K training videos and 24.7K validation videos in 174 classes. SSv2 is more challenging because it requires stronger temporal modeling [209, 137]. Diving-48 contains 15.9K training videos and 2K validation videos in 48 fine-grained diving actions. It is designed to be unbiased towards static representations, which means a model cannot simply rely on the objects or background to determine the action.

7.3.1 Effectiveness of Components

To demonstrate the effectiveness of our proposed components in Section 7.2, we compare our method to three baselines. The first baseline is a frozen space-only model. Recall in Section 7.1, space-only model processes input frames independently and performs temporal average pooling in the end. We freeze the image backbone and only tune the classification head, which is also known as linear probing [61]. The second baseline is a full finetuned space-only model. It should be able to learn spatial information from video data, but still has difficulties in capturing temporal information. The third baseline is a full finetuned space-time video model, which should serve as oracle. Here we choose TimeSformer [8] because we are based on the same ViT-B backbone and share a similar structure (*i.e.*, divided space-time attention).

In the experiments, we use the ViT-B/16 pre-trained on IN-21K as image backbone, and we com-

Table 7.1: Effectiveness of proposed components. We compare to three baselines on Something-something-v2 dataset. Spatial adaptation, temporal adaptation and joint adaptation gradually add spatiotemporal reasoning to the frozen image model. Views = #frames $\times$ #temporal $\times$ #spatial.

Methods	Pretrain	Param (M)	Tunable Param (M)	Top-1	Top-5	Views
Frozen space-only	IN-21K	86	0.1	15.1	36.9	8×1×3
Finetuned space-only	IN-21K	86	86	36.2	68.1	8×1×3
Finetuned space-time [8]	IN-21K	121	121	59.5	85.6	8×1×3
Frozen space-only + spatial adaptation	IN-21K	89	3.7	36.7	68.3	8×1×3
+ temporal adaptation	IN-21K	97	10.8	61.2	87.7	8×1×3
+ joint adaptation (AIM)	IN-21K	100	14.3	**62.0**	87.9	8×1×3
AIM	CLIP	100	14.3	**66.4**	90.5	8×1×3

pare the proposed method with the baselines on SSv2 [53] where temporal modeling is critical. The results for three baselines are shown in Table 7.1 top. We can see that the frozen space-only model only needs to tune 0.1M parameters, but it also performs much worse than the full finetuned video model (15.1% vs 59.5%). Full finetuning the space-only model allows it to learn improved spatial representations from video data and largely improves the performance (15.1% $\rightarrow$ 36.2%). However, it also significantly increases the number of tunable parameters and still has a large gap from the full finetuned video model due to lack of temporal modeling. The third baseline, full finetuned video model, achieves the highest accuracy due to its strong spatiotemporal reasoning capability, but the number of tunable parameters increases again to 121M.

Our goal is to add a few tunable parameters to the frozen space-only model and close the gap to full finetuned video model. As shown in Table 7.1 bottom, after spatial adaptation, the frozen space-only model achieves comparable performance with the full finetuned space-only model (36.7% vs 36.2%), with significantly less number of tunable parameters (3.7M vs 86M). This means spatial adaptation is able to help frozen image models to achieve good spatial modeling on video data. In addition, adding temporal adaptation further boosts the performance to 61.2%, which is even

higher than the full finetuned video model. This indicates that our temporal adaptation introduces strong temporal modeling to the space-only model. Finally, joint adaptation is incorporated to tune the features for improved spatiotemporal reasoning, which is our method AIM. We not only close the gap to full finetuned space-time video model but obtain higher accuracy (62% vs 59.5%) with fewer number of tunable parameters (14.3M vs 86M). These results successfully validate the effectiveness of our proposed adaptation strategies.

Furthermore, our method could easily take advantage of stronger pre-trained image models and adapt them for video action recognition. For example, simply switch the ViT-B/16 pre-trained on IN-21K to CLIP pre-trained, we obtain another accuracy boost (62.0% $\rightarrow$ 66.4%)

7.3.2 Comparisons to the State of the Art

In this section, we compare the proposed method with state-of-the-art video models on four video action recognition benchmarks. For all the experiments, we use the ViT models pre-trained by CLIP [131]. We mostly follow the training settings in [109].

7.3.2.1 Results on Kinetics-400 and Kinetics-700

Table 7.2 presents the comparisons with state-of-the-art video models on K400 dataset. First, we can see that with ViT-B/16 backbone, our method only needs to tune 11M parameters for competitive performance, which is much smaller than previous video models. Taking input of 8 frames as an example, AIM ViT-B/16 achieves 83.9% top-1 accuracy while only requiring 606 GFLOPs. When using 16 input frames, our method even outperforms MTV-L [183], which requires more than 10$\times$ computations (1214 vs 18050 GFLOPs). When switching to larger backbone ViT-L/14, we achieve the highest accuracy 87.5% on K400 dataset, with 38M tunable parameters.

Table 7.2: Comparison to state-of-the-art on Kinetics-400. Views = #frames × #temporal × #spatial. AIM outperforms state-of-the-art video models while tuning much less number of parameters.

Methods	Pretrain	GFLOPs	Param (M)	Tunable Param (M)	Top-1	Top-5	Views
MViT-B [41]	-	4095	37	37	81.2	95.1	64×3×3
UniFormer-B [93]	IN-1K	3108	50	50	83.0	95.4	32×4×3
TimeSformer-L [8]	IN-21K	7140	121	121	80.7	94.7	64×1×3
ViViT-L/16×2 FE [3]	IN-21K	3980	311	311	80.6	92.7	32×1×1
VideoSwin-L [109]	IN-21K	7248	197	197	83.1	95.9	32×4×3
MViTv2-L (312 ↑) [97]	IN-21K	42420	218	218	86.1	97.0	32×3×5
MTV-L [183]	JFT	18050	876	876	84.3	96.3	32×4×3
TokenLearner-L/10 [135]	JFT	48912	450	450	85.4	96.3	64×4×3
PromptCLIP A7 [81]	CLIP	-	-	-	76.8	93.5	16×5×1
ActionCLIP [162]	CLIP	16890	142	142	83.8	97.1	32×10×3
X-CLIP-L/14 [119]	CLIP	7890	420	420	87.1	97.6	8×4×3
EVL ViT-L/14 [102]	CLIP	8088	368	59	87.3	-	32×3×1
AIM ViT-B/16	CLIP	606	97	11	83.9	96.3	8×3×1
AIM ViT-B/16	CLIP	1214	97	11	84.5	96.6	16×3×1
AIM ViT-B/16	CLIP	2428	97	11	84.7	96.7	32×3×1
AIM ViT-L/14	CLIP	2802	341	38	86.8	97.2	8×3×1
AIM ViT-L/14	CLIP	5604	341	38	87.3	97.6	16×3×1
AIM ViT-L/14	CLIP	11208	341	38	**87.5**	**97.7**	32×3×1

Note that several works also leverage CLIP pre-trained models to do video action recognition. However, ActionCLIP [162] and X-CLIP [119] are multimodal methods which require additional text branch and tune the whole model end-to-end. PromptCLIP [81] applies prompt tuning [92] to CLIP and adds several temporal blocks for temporal modeling. EVL [102] introduces a new decoder branch to learn temporal information. However, AIM simply re-uses image pre-trained self-attention for temporal modeling. This makes AIM much simpler than previous methods, yet achieving better performance at much less tunable parameters. The simplicity also makes AIM much easier to adapt to different model architectures (single modal or multi-modal models). But previous methods such as ActionCLIP/X-CLIP/PromptCLIP cannot leverage pure image backbone because they need an additional text branch.

Table 7.3: Comparison to state-of-the-art on Something-Something-v2. K400†/K600† indicates the model is pre-trained on both IN-21K and K400/K600.

Methods	Pretrain	GFLOPs	Param (M)	Tunable Param (M)	Top-1	Top-5	Views
TimeSformer-L [8]	IN-21K	7140	121	121	62.4	-	64×1×3
MTV-B [183]	IN-21K	4790	310	310	67.6	90.4	32×4×3
MViT-B [41]	K400	510	37	37	67.1	90.8	32×1×3
MViTv2-B [97]	K400	675	51	51	70.5	92.7	40×1×3
ViViT-L/16×2 [3]	K400†	11892	311	311	65.4	89.8	16×4×3
VideoSwin-B [109]	K400†	963	89	89	69.6	92.7	32×1×1
Omnivore [50]	K400†	-	-	-	71.4	93.5	32×1×3
MViTv2-L (312 ↑) [97]	K400†	8484	213	213	**73.3**	**94.1**	32×1×3
UniFomer-B [93]	K600†	777	50	50	71.2	92.8	32×1×3
CoVeR [199]	JFT-3B	-	-	-	70.9	-	-
EVL ViT-B/16 [102]	CLIP	2047	182	86	62.4	-	32×1×3
EVL ViT-L/14 [102]	CLIP	9641	484	175	66.7	-	32×1×3
AIM ViT-B/16	CLIP	624	100	14	66.4	90.5	8×1×3
AIM ViT-B/16	CLIP	1248	100	14	68.1	91.8	16×1×3
AIM ViT-B/16	CLIP	2496	100	14	69.1	92.2	32×1×3
AIM ViT-L/14	CLIP	2877	354	50	67.6	91.6	8×1×3
AIM ViT-L/14	CLIP	5754	354	50	69.4	92.3	16×1×3
AIM ViT-L/14	CLIP	11508	354	50	70.6	92.7	32×1×3

Furthermore, we evaluate our method on K700 dataset in Table7.4. We can see that AIM ViT-B/16 with 11M tunable parameters is able to outperform MTV-L (875M) and MViTv2-B (51M). And AIM ViT-L/14 (38M) achieves comparable performance with MaskFeat (218M) [169]. Note that MaskFeat uses larger input resolution (312 vs 224) and more input frames (40 vs 32) than us. This again justifies the effectiveness of our efficient adaptation pipeline.

7.3.2.2 Results on Something-Something-v2

Table 7.3 presents the performance comparisons on SSv2. Based on CLIP ViT-L/14, our method achieves competitive or better performance than most prior arts. In terms of fair comparison to EVL, which also uses CLIP pre-trained image encoder, we achieve significantly higher accuracy (70.6% $>$ 66.7%), while introducing less tunable parameters (50M $<$ 175M). Note that to introduce temporal modeling into image model, EVL adds 12 layers of decoder blocks, while our method simply reuse image pre-trained self-attention layers to achieve stronger temporal modeling .

However, our method falls behind some full finetuned video models [50, 97, 93]. One reason is that SSv2 is a “temporal-heavy" dataset [137], which requires model to really understand the temporal evolution within a video. In order to obtain high accuracy, most previous video models are first pre-trained on some video datasets (such as K400/K600) to learn good spatiotemporal representations, then finetuned on SSv2. But our method still starts from the image pre-trained model. Another reason is that simply reusing the image pre-trained self-attention for temporal modeling may not be able to fully capture the complicated temporal information in SSv2 videos. This suggests that we need to conduct more temporal adaptation for these challenging “temporal-heavy" datasets.

7.3.2.3 Results on Diving-48

A diving class in Diving-48 [96] is defined by the combination of takeoff, movements in flight and entry, thus it requires the model to differentiate such fine-grained actions. As shown in Table 7.5, our method with 11M tunable parameters outperforms all prior methods. AIM ViT-L/14 further improves the top-1 accuracy to 90.6%. Comparing to ORViT [67], despite they leverage additional object tracking model, our method still outperforms it with much less tunable parameters. This

Table 7.4: Comparison to state-of-the-art on Kinetics-700. AIM achieves competitive performance while using less tunable parameters and smaller input scale.

Method	Pretrain	Tunable Param (M)	Top-1
VidTR-L [204]	IN-21K	91	70.2
MTV-L [183]	IN-21K	876	75.2
MViTv2-B [97]	-	51	76.6
MViTv2-L ($40 \times 312 \uparrow$) [97]	IN-21K	218	79.4
MaskFeat ($40 \times 312 \uparrow$) [169]	K700	218	**80.4**
AIM ViT-B/16	CLIP	11	76.9
AIM ViT-L/14	CLIP	38	**80.4**

Table 7.5: Comparison to state-of-the-art on Diving-48. AIM outperforms previous best methods while using less tunable parameters.

Method	Pretrain	Tunable Param (M)	Top-1
TimeSformer-L [8]	IN-21K	121	81.0
VideoSwin-B [109]	IN-21K	88	81.9
BEVT [163]	K400†	88	86.7
SIFAR-B-14 [42]	IN-21K	87	87.3
ORViT [67]	IN-21K	160	88.0
AIM ViT-B/16	CLIP	11	88.9
AIM ViT-L/14	CLIP	38	**90.6**

suggests that efficient finetuning can handle fine-grained action recognition.

7.4 Discussion

Different Pre-trained Models. Here we demonstrate the effectiveness of AIM on different pre-trained models. In Table 7.6, we first show AIM based on ViT-B backbone. We compare AIM to TimeSformer because we use the same backbone (ViT-B) and have a similar structure (i.e., both using divided space-time attention). As can be seen, AIM achieves better performance than

Table 7.6: Performance of using different pre-trained image models on K400. AIM achieves competitive or better performance with the corresponding fully finetuned videos on different backbones. AIM also largely saves training memory footprint and time cost.

Model	Backbone	Pretrain	Tunable Param (M)	Mem (G)	Time (H)	Top-1
TimeSformer	ViT-B	IN-21K	121	10	20	78.5
AIM	ViT-B	IN-21K	11	7	15	78.8
TimeSformer	ViT-B	CLIP	121	10	20	82.0
AIM	ViT-B	CLIP	11	7	15	83.9
VideoSwin-B	Swin-B	IN-21K	88	18	64	82.7
AIM	Swin-B	IN-21K	9.2	9	37	82.1

Table 7.7: Effect of position of Adapters. Skip means adding Adapters every two blocks.

Position	Tunable Param (M)	Top-1
Bottom 6	5.6	80.7
Top 6	5.6	83.3
Skip	5.6	83.2
All	11	**83.9**

full finetuned TimeSformer under both IN-21K and CLIP pre-trained weights. Then we apply AIM to Swin-B backbone and compare it to VideoSwin when we both use Swin-B and IN-21K pre-training. Similarly, AIM achieves comparable performance with full finetuned VideoSwin.

Data Efficiency. One advantage of our efficient tuning paradigm is that we can keep the well pre-trained image representations intact. In the scenario where downtream data is insufficient, our method will be less prone to over-fitting compared to full finetuning. In Figure 7.2, we compare AIM with full finetuned TimeSformer under different amounts of training data on K400. For fair comparison, both AIM and TimeSformer use CLIP pre-trained ViT-B/16 as backbone. We can observe that under all scenarios, our method AIM outperforms full finetuned TimeSformer. In

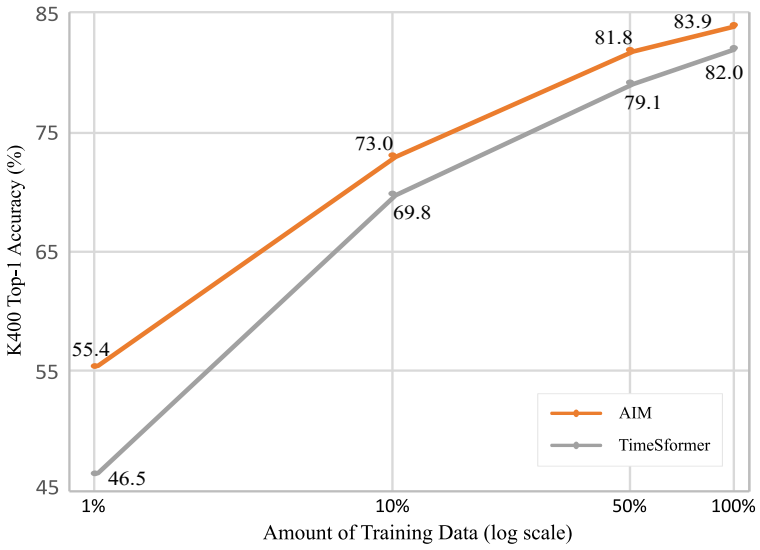


Figure 7.2: Data efficiency comparison. AIM outperforms full finetuned TimeSformer under all scenarios, especially in low data regime.

Table 7.8: Effect of bottleneck ratio of Adapters.

Ratio	Tunable Param (M)	Top-1
0.0625	3	83.3
0.125	5.6	83.4
0.25	11	**83.9**
0.5	21	83.8

particular, when the amount of data becomes less, the advantage of AIM becomes larger. For example, when there is only 1% of training data, we outperform TimeSformer by a significant margin of 8.9%.

Training Cost. Table 7.6 also shows the training time (hours) and memory cost (GB) of our method and full finetuning on different backbones. All metrics are measured on 8 Tesla V100 GPUs. Compared to TimeSformer, we reduce the memory cost by 30% and training time by 25%. Compared to VideoSwin, we reduce the memory cost by 50% and training time by 42%.

Position of Adapters. By default, we add Adapters to every ViT block (12 blocks in total). Here we study the effect of adding Adapters in different layers. We add Adapters to the bottom 6 blocks (close to the input), top 6 blocks (close to the output) and one every two blocks. All these variants have the same number of tunable parameters. As can be seen in Table 7.7, adding Adapters to the bottom 6 blocks yields much worse performance than others. We hypothesize that the shallow layers learn generic representations which do not need much adaptation, while deeper layers learn task-specific features like temporal information thus feature adaptation is important. Adding Adapters to the top 6 blocks achieves comparable performance with adding to all blocks while saving half of the parameters. This could serve as a good candidate when training resources are more limited.

Bottleneck Ratio of Adapters. By tuning the bottleneck ratio of Adapters, we can easily control the number of tunable parameters. Here we study how the bottleneck ratio of Adapters affects the final performance. The results in Table 7.8 reveal that a larger bottleneck ratio tends to achieve better performance, but it will also introduce more tunable parameters. The performance plateaus after bottleneck ratio goes beyond 0.25. Note that a small ratio of 0.0625 could still achieve 83.3% top-1 accuracy on K400, which is competitive among state-of-the-art video models in Table 7.2 while introducing only 3M tunable parameters.

Per-class Analysis. In Table 7.3, we show that AIM still falls behind some SoTA full finetuned video models on the "temporal-heavy" Something-Something-v2 (SSv2) dataset. We conjecture one reason is that simply reusing the image pre-trained self-attention for temporal modeling may not be able to fully capture the complicated temporal information in some nuanced action classes in SSv2. To provide further insights, we compute the per-class accuracy differences of AIM and TimeSformer on SSv2 and show the top-5 and bottom-5 classes in Figure 7.3. We can see that the classes where AIM performs better are normal action classes with decent motion. The classes where AIM performs worse are those with minor differences (*e.g.*, "Pulling something from left to

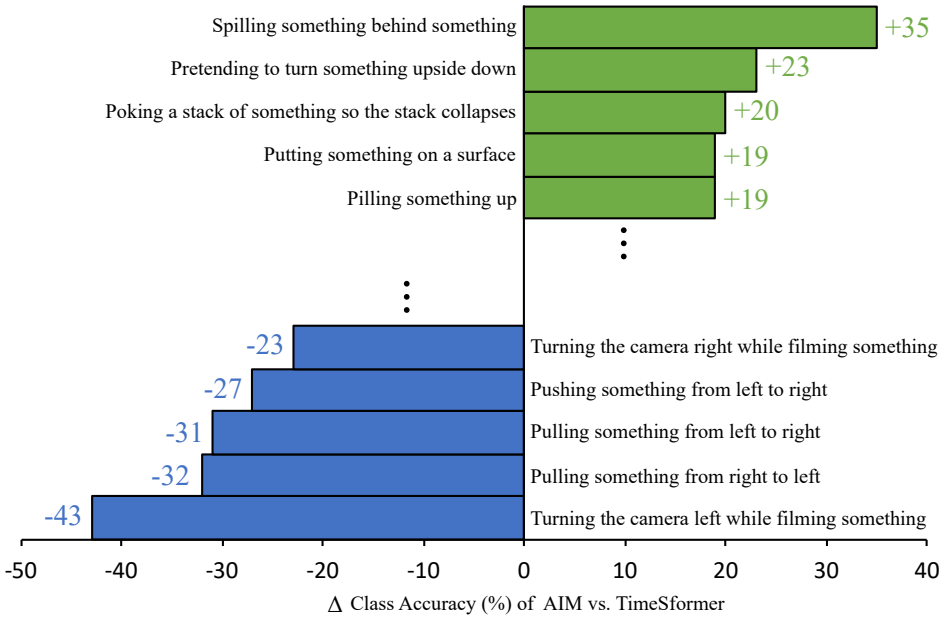


Figure 7.3: The figure shows the differences of each class's accuracy of AIM and TimeSformer on Something-Something-v2. Here we only plot the top-5 and bottom-5 classes.

right" vs. "Pulling something from right to left"). In order to tell these actions apart, the model needs to distinguish between the nuances, especially in motion. Given most of model parameters are frozen in our method, AIM may lack the capacity to capture such complex temporal information.

Visualization. we present the attention map visualizations of the frozen space-only model, Spatial Adaptation (SA) model, Spatial Adaptation plus Temporal Adaptation (TA) model, and the full finetuned TimeSformer.

On Figure 7.4 left, we visualize an action "Brush Painting" from Kinetics-400 dataset. We can see that the attention maps of the frozen space-only model are very scattered, and it doesn't attend to the brush region in the first two frames. Adding SA enhances the attention on the brush, but the model still focuses on areas that are unrelated to the action. Further adding TA helps the model

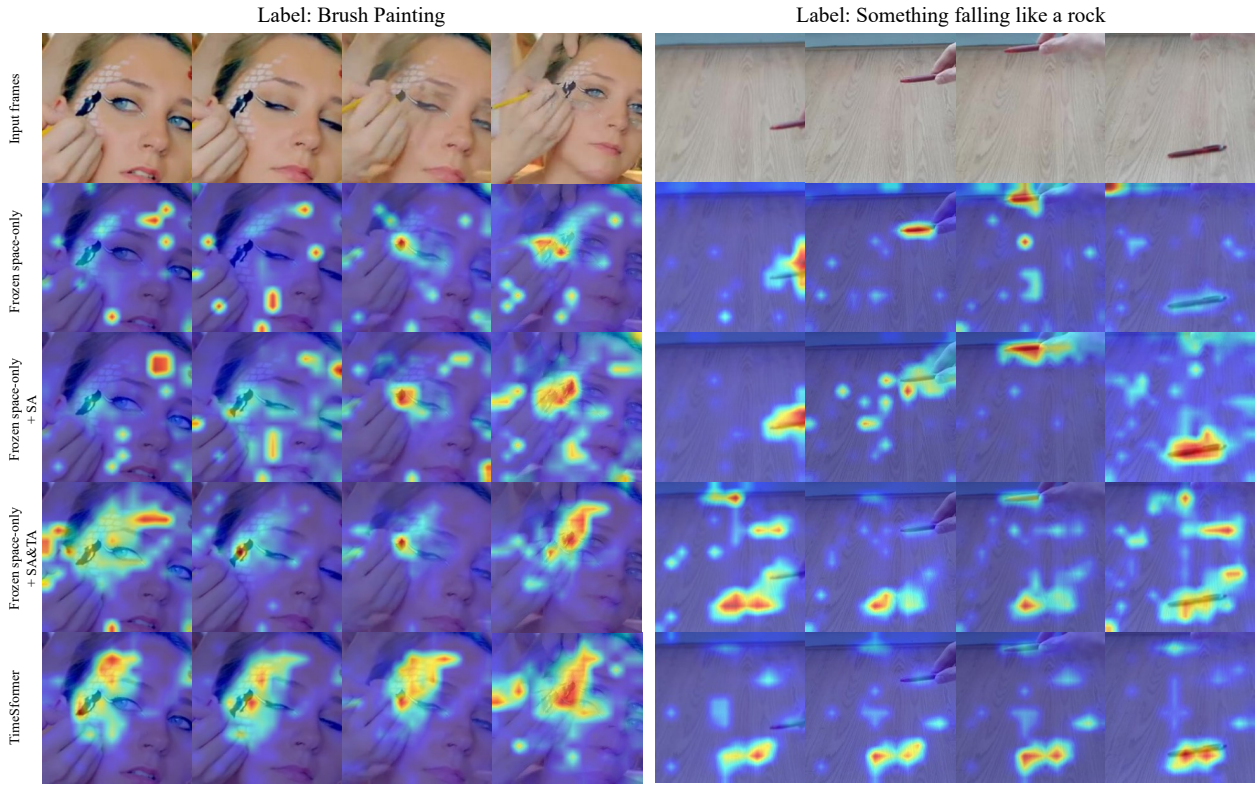


Figure 7.4: Attention map visualizations of AIM variants and the full finetuned TimeSformer. With the help of temporal adaptation (TA), our method is able to focus on motion salient regions which helps to make a correct prediction.

to learn temporal information. We can see that the model now focuses more on the brush painting area, which is similar to what full finetuned TimeSformer does.

On Figure 7.4 right, we visualize an action "Something falling like a rock" from Something-Something-v2 dataset. To correctly recognize this action, the model needs to learn how the object moves in the input frames. We first observe that both the frozen space-only model and SA model have good attention on the object, but they fail to model the movement of the object which leads to wrong prediction. In contrast, TA helps the model to learn the relationship among input frames. The attention map shows that the model not only focuses on the object but also learns the track of the object. Instead, TimeSformer always attends to the bottom region without showing the object

path.

7.5 Summary

In this work, we propose a new way to efficiently transfer pre-trained image models for video action recognition. We introduce spatial adaptation, temporal adaptation and joint adaptation to gradually add spatiotemporal reasoning to an image model. Since only newly added Adapters are updated, our training cost is substantially lower than other full finetuned video models. Yet we achieve comparable or even better performance than prior arts on four benchmarks. Our method is simple and generally applicable, which has the potential to leverage more powerful image foundation models in the future. Despite all the benefits, one limitation is that our simple strategy of reusing spatial attention for temporal modeling might not be strong enough for temporally challenging videos. Since video temporal modeling can be viewed as a form of sequence modeling, we might be able to reuse pre-trained weights from text or audio models instead of image models in the future.

CHAPTER 8: CONCLUSION AND FUTURE WORK

8.1 Conclusion

In this dissertation, we improve the efficiency and effectiveness of deep representation learning from multiple perspectives. The proposed methods not only improve the training efficiency, inference efficiency and data efficiency of previous learning methods, but also enhance the performance, robustness and scalability of learned representations. We demonstrate the effectiveness of the proposed methods on a wide range of tasks and domains such as image classification, object detection, instance segmentation, video action recognition and action detection.

In Chapter 3, we point out that traditional neural networks are static, and we propose a new method to learn adaptive representations. Our adaptive neural network can run at different computation complexities during inference time. Therefore, we only need one network to meet the dynamic resource budgets in real devices. Our method highlights the importance of simultaneously considering network width and input resolution for efficient representation learning and mutually train different network configurations. It outperforms traditional neural networks on various tasks under different model complexities. It also bears the benefits of training and deploying only one model to meet diverse resource budgets.

In Chapter 4, we extend the method to learn adaptive spatiotemporal representations for video understanding. We propose to asymmetrically sample subnetworks, input resolutions and frames in the mutual training process. After training, the adaptive network can run at different widths, resolutions and number of frames. We demonstrate its effectiveness and efficiency on multiple video understanding tasks including video recognition and action detection.

In Chapter 5, we extend the mutual learning method to improve the representation learning of the

target full network. We regularize the representation learning by ensuring that subnetworks make consistent predictions with the full network when fed with differently transformed images. Our method helps the network to learn robust and generalizable representations which achieves better performance on multiple tasks.

Besides learning methods, we explore efficient neural architecture search (NAS) methods in Chapter 6. We show that recent training-free NAS metrics are not fairly evaluated. Their performance is no better than the trivial number-of-parameter metric while being much more complicated to compute. Based on our observations, we proposed a new efficient training-based NAS method which outperforms previous methods with significantly smaller search cost. Our method is also more robust to different search spaces.

In Chapter 7, we propose a new spatiotemporal representation learning method by bootstraping from image models. Our method freezes the pre-trained image model, re-uses spatial self-attention for temporal modeling, and introduces few light-weight Adapters to tune the representations. It largely saves the training cost compared to traditional full finetuning and achieves even better performance. The method is also generally applicable to different model structures and future stronger image foundation models.

8.2 Future Work

Chapter 3 and 4 introduce the methods to learn adaptive representations in convolutional neural networks. With the increasing popularity and importance of the (Vision) Transformer structures, it would be interesting and insightful to explore novel methods in learning adaptive representations for Transformer structures.

The representation learning methods discussed in this dissertation primarily focus on supervised

learning. However, it would indeed be intriguing to investigate the applicability and effectiveness of the proposed methods, such as, adaptive mutual learning (Chapter 3 and 4), GradAug (Chapter 5), and AIM (Chapter 7), in the context of unsupervised representation learning.

Finally, in light of recent advancements in foundation models, such as Stable Diffusion [134], Segment Anything [85], ChatGPT [121], and LLaMA [156], it has become crucial to explore efficient methods for leveraging these models and customizing them to address specific tasks of interest. Chapter 7 of this dissertation introduces AIM, which presents a means of adapting image models for video action recognition. This concept holds potential for extension to other tasks, such as adapting image generative models for video generation, video editing, and 3D object generation. Moreover, it opens up possibilities for cross-modality model adaptation, such as adapting large language models to large vision models.

LIST OF REFERENCES

[1] Official implementation of te-nas. `https://github.com/VITA-Group/TENAS`.

[2] M. S. Abdelfattah, A. Mehrotra, Ł. Dudziak, and N. D. Lane. Zero-cost proxies for lightweight nas. In *International Conference on Learning Representations*, 2020.

[3] A. Arnab, M. Dehghani, G. Heigold, C. Sun, M. Lučić, and C. Schmid. Vivit: A video vision transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6836–6846, 2021.

[4] H. Bahng, A. Jahanian, S. Sankaranarayanan, and P. Isola. Visual prompting: Modifying pixel space to adapt pre-trained models. *arXiv preprint arXiv:2203.17274*, 2022.

[5] H. Bao, L. Dong, S. Piao, and F. Wei. BEiT: BERT Pre-Training of Image Transformers. *arXiv preprint arXiv:2106.08254*, 2021.

[6] E. Ben Zaken, Y. Goldberg, and S. Ravfogel. BitFit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1–9, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[7] G. Bender, P.-J. Kindermans, B. Zoph, V. Vasudevan, and Q. Le. Understanding and simplifying one-shot architecture search. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 550–559. PMLR, 10–15 Jul 2018.

[8] G. Bertasius, H. Wang, and L. Torresani. Is space-time attention all you need for video understanding? In *Proceedings of the International Conference on Machine Learning (ICML)*, July 2021.

[9] L. Bossard, M. Guillaumin, and L. Van Gool. Food-101 – mining discriminative components with random forests. In *European Conference on Computer Vision*, 2014.

[10] A. Brock, T. Lim, J. M. Ritchie, and N. Weston. Smash: one-shot model architecture search through hypernetworks. In *International Conference on Learning Representations*, 2017.

[11] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901, 2020.

[12] H. Cai, C. Gan, T. Wang, Z. Zhang, and S. Han. Once-for-all: Train one network and specialize it for efficient deployment. In *International Conference on Learning Representations*, 2019.

[13] S. Carbonnelle and C. De Vleeschouwer. Layer rotation: a surprisingly powerful indicator of generalization in deep networks? *arXiv preprint arXiv:1806.01603*, 2018.

[14] J. Carreira, E. Noland, C. Hillier, and A. Zisserman. A short note on the kinetics-700 human action dataset. *arXiv preprint arXiv:1907.06987*, 2019.

[15] 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*, pages 6299–6308, 2017.

[16] H. Chen, Y. Wang, C. Xu, B. Shi, C. Xu, Q. Tian, and C. Xu. Addernet: Do we really need multiplications in deep learning? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1468–1477, 2020.

[17] K. Chen, J. Wang, J. Pang, Y. Cao, Y. Xiong, X. Li, S. Sun, W. Feng, Z. Liu, J. Xu, et al. Mmdetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019.

[18] K. Chen, J. Wang, J. Pang, Y. Cao, Y. Xiong, X. Li, S. Sun, W. Feng, Z. Liu, J. Xu, Z. Zhang, D. Cheng, C. Zhu, T. Cheng, Q. Zhao, B. Li, X. Lu, R. Zhu, Y. Wu, J. Dai, J. Wang, J. Shi, W. Ouyang, C. C. Loy, and D. Lin. MMDetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019.

[19] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017.

[20] S. Chen, C. Ge, Z. Tong, J. Wang, Y. Song, J. Wang, and P. Luo. Adaptformer: Adapting vision transformers for scalable visual recognition. *arXiv preprint arXiv:2205.13535*, 2022.

[21] W. Chen, X. Gong, X. Liu, Q. Zhang, Y. Li, and Z. Wang. Fasterseg: Searching for faster real-time semantic segmentation. In *International Conference on Learning Representations*, 2019.

[22] W. Chen, X. Gong, and Z. Wang. Neural architecture search on imagenet in four gpu hours: A theoretically inspired perspective. In *International Conference on Learning Representations*, 2020.

[23] X. Chen and C.-J. Hsieh. Stabilizing differentiable architecture search via perturbation-based regularization. In *International Conference on Machine Learning*, pages 1554–1565. PMLR, 2020.

[24] X. Chen, L. Xie, J. Wu, and Q. Tian. Progressive differentiable architecture search: Bridging the depth gap between search and evaluation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1294–1303, 2019.

[25] X. Chen, S. Xie, and K. He. An empirical study of training self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9640–9649, 2021.

[26] Y. Chen, X. Dai, M. Liu, D. Chen, L. Yuan, and Z. Liu. Dynamic convolution: Attention over convolution kernels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11030–11039, 2020.

[27] Y. Chen, X. Dai, M. Liu, D. Chen, L. Yuan, and Z. Liu. Dynamic relu. In *European Conference on Computer Vision*, pages 351–367. Springer, 2020.

[28] Y. Chen, Y. Kalantidis, J. Li, S. Yan, and J. Feng. Multi-fiber networks for video recognition. In *Proceedings of the european conference on computer vision (ECCV)*, pages 352–367, 2018.

[29] Y. Chen, T. Yang, X. Zhang, G. Meng, X. Xiao, and J. Sun. Detnas: Backbone search for object detection. *Advances in Neural Information Processing Systems*, 32:6642–6652, 2019.

[30] A. Coates, A. Ng, and H. Lee. An analysis of single-layer networks in unsupervised feature learning. In G. Gordon, D. Dunson, and M. Dudík, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 215–223, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR.

[31] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le. Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 113–123, 2019.

[32] J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[33] 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*, pages 248–255. Ieee, 2009.

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

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

[36] X. Dong, J. Bao, D. Chen, W. Zhang, N. Yu, L. Yuan, D. Chen, and B. Guo. Cswin transformer: A general vision transformer backbone with cross-shaped windows. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12124–12134, 2022.

[37] X. Dong and Y. Yang. Nas-bench-201: Extending the scope of reproducible neural architecture search. In *International Conference on Learning Representations*, 2019.

[38] X. Dong and Y. Yang. Searching for a robust neural architecture in four gpu hours. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1761–1770, 2019.

[39] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2020.

[40] H. Fan, Y. Li, B. Xiong, W.-Y. Lo, and C. Feichtenhofer. Pyslowfast. `https://github.com/facebookresearch/slowfast`, 2020.

[41] H. Fan, B. Xiong, K. Mangalam, Y. Li, Z. Yan, J. Malik, and C. Feichtenhofer. Multiscale vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6824–6835, 2021.

[42] Q. Fan, C.-F. Chen, and R. Panda. Can an image classifier suffice for action recognition? In *International Conference on Learning Representations*, 2022.

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

[44] C. Feichtenhofer, H. Fan, Y. Li, and K. He. Masked autoencoders as spatiotemporal learners. *arXiv preprint arXiv:2205.09113*, 2022.

[45] 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*, pages 6202–6211, 2019.

[46] 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*, pages 6202–6211, 2019.

[47] Y. Gao, X. Shi, Y. Zhu, H. Wang, Z. Tang, X. Zhou, M. Li, and D. N. Metaxas. Visual Prompt Tuning for Test-time Domain Adaptation. *arXiv preprint arXiv:2210.04831*, 2022.

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

[49] G. Ghiasi, T.-Y. Lin, and Q. V. Le. Dropblock: A regularization method for convolutional networks. In *Advances in Neural Information Processing Systems*, pages 10727–10737, 2018.

[50] R. Girdhar, M. Singh, N. Ravi, L. van der Maaten, A. Joulin, and I. Misra. Omnivore: A single model for many visual modalities. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16102–16112, 2022.

[51] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[52] P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

[53] R. Goyal, S. Ebrahimi Kahou, V. Michalski, J. Materzynska, S. Westphal, H. Kim, V. Haenel, I. Fruend, P. Yianilos, M. Mueller-Freitag, et al. The" something something" video database for learning and evaluating visual common sense. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5842–5850, 2017.

[54] C. Gu, C. Sun, D. A. Ross, C. Vondrick, C. Pantofaru, Y. Li, S. Vijayanarasimhan, G. Toderici, S. Ricco, R. Sukthankar, et al. Ava: A video dataset of spatio-temporally localized atomic visual actions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6047–6056, 2018.

[55] Z. Guo, X. Zhang, H. Mu, W. Heng, Z. Liu, Y. Wei, and J. Sun. Single path one-shot neural architecture search with uniform sampling. In *European Conference on Computer Vision*, pages 544–560. Springer, 2020.

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

[57] K. Han, Y. Wang, Q. Tian, J. Guo, C. Xu, and C. Xu. Ghostnet: More features from cheap operations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1580–1589, 2020.

[58] K. Hara, H. Kataoka, and Y. Satoh. Learning spatio-temporal features with 3d residual networks for action recognition. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 3154–3160, 2017.

[59] J. He, C. Zhou, X. Ma, T. Berg-Kirkpatrick, and G. Neubig. Towards a unified view of parameter-efficient transfer learning. In *International Conference on Learning Representations*, 2022.

[60] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16000–16009, 2022.

[61] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9729–9738, 2020.

[62] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[63] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[64] 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*, pages 770–778, 2016.

[65] T. He, Z. Zhang, H. Zhang, Z. Zhang, J. Xie, and M. Li. Bag of tricks for image classification with convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 558–567, 2019.

[66] D. Hendrycks and T. Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *arXiv preprint arXiv:1903.12261*, 2019.

[67] R. Herzig, E. Ben-Avraham, K. Mangalam, A. Bar, G. Chechik, A. Rohrbach, T. Darrell, and A. Globerson. Object-region video transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3148–3159, 2022.

[68] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. De Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pages 2790–2799. PMLR, 2019.

[69] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, et al. Searching for mobilenetv3. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1314–1324, 2019.

[70] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[71] E. J. Hu, yelong shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

[72] J. Hu, L. Shen, and G. Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018.

[73] Y. Hu, Y. Liang, Z. Guo, R. Wan, X. Zhang, Y. Wei, Q. Gu, and J. Sun. Angle-based search space shrinking for neural architecture search. In *European Conference on Computer Vision*, pages 119–134. Springer, 2020.

[74] G. Huang, D. Chen, T. Li, F. Wu, L. van der Maaten, and K. Weinberger. Multi-scale dense networks for resource efficient image classification. In *International Conference on Learning Representations*, 2018.

[75] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger. Deep networks with stochastic depth. In *European conference on computer vision*, pages 646–661. Springer, 2016.

[76] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[77] C. Jia, Y. Yang, Y. Xia, Y.-T. Chen, Z. Parekh, H. Pham, Q. Le, Y.-H. Sung, Z. Li, and T. Duerig. Scaling up visual and vision-language representation learning with noisy text supervision. In *International Conference on Machine Learning*, pages 4904–4916. PMLR, 2021.

[78] M. Jia, L. Tang, B.-C. Chen, C. Cardie, S. Belongie, B. Hariharan, and S.-N. Lim. Visual prompt tuning. *arXiv preprint arXiv:2203.12119*, 2022.

[79] J. Jiang, Y. Cao, L. Song, S. Z. Y. Li, Z. Xu, Q. Wu, C. Gan, C. Zhang, and G. Yu. Human centric spatio-temporal action localization. In *ActivityNet Workshop on CVPR*, 2018.

[80] S. Jie and Z.-H. Deng. Convolutional bypasses are better vision transformer adapters. *arXiv preprint arXiv:2207.07039*, 2022.

[81] C. Ju, T. Han, K. Zheng, Y. Zhang, and W. Xie. Prompting visual-language models for efficient video understanding. *arXiv preprint arXiv:2112.04478*, 2021.

[82] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017.

[83] M. G. Kendall. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93, 1938.

[84] E. Kim, C. Ahn, and S. Oh. Nestednet: Learning nested sparse structures in deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8669–8678, 2018.

[85] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, et al. Segment anything. *arXiv preprint arXiv:2304.02643*, 2023.

[86] A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. *(Technical Report, University of Toronto*, 2009.

[87] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[88] H. Kuang, Y. Zhu, Z. Zhang, X. Li, J. Tighe, S. Schwertfeger, C. Stachniss, and M. Li. Video contrastive learning with global context. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3195–3204, 2021.

[89] G. Larsson, M. Maire, and G. Shakhnarovich. Fractalnet: Ultra-deep neural networks without residuals. *arXiv preprint arXiv:1605.07648*, 2016.

[90] J. Lee, L. Xiao, S. Schoenholz, Y. Bahri, R. Novak, J. Sohl-Dickstein, and J. Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. *Advances in neural information processing systems*, 32:8572–8583, 2019.

[91] N. Lee, T. Ajanthan, and P. Torr. Snip: Single-shot network pruning based on connection sensitivity. In *International Conference on Learning Representations*, 2018.

[92] B. Lester, R. Al-Rfou, and N. Constant. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*, 2021.

[93] K. Li, Y. Wang, G. Peng, G. Song, Y. Liu, H. Li, and Y. Qiao. Uniformer: Unified transformer for efficient spatial-temporal representation learning. In *International Conference on Learning Representations*, 2021.

[94] L. Li and A. Talwalkar. Random search and reproducibility for neural architecture search. In *Uncertainty in artificial intelligence*, pages 367–377. PMLR, 2020.

[95] X. L. Li and P. Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, Online, Aug. 2021. Association for Computational Linguistics.

[96] Y. Li, Y. Li, and N. Vasconcelos. Resound: Towards action recognition without representation bias. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 513–528, 2018.

[97] Y. Li, C.-Y. Wu, H. Fan, K. Mangalam, B. Xiong, J. Malik, and C. Feichtenhofer. Mvitv2: Improved multiscale vision transformers for classification and detection. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4794–4804, 2022.

[98] 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*, pages 7083–7093, 2019.

[99] J. Lin, C. Gan, and S. Han. Tsm: Temporal shift module for efficient video understanding. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7083–7093, 2019.

[100] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.

[101] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[102] Z. Lin, S. Geng, R. Zhang, P. Gao, G. de Melo, X. Wang, J. Dai, Y. Qiao, and H. Li. Frozen clip models are efficient video learners. *arXiv preprint arXiv:2208.03550*, 2022.

[103] C. Liu, L.-C. Chen, F. Schroff, H. Adam, W. Hua, A. L. Yuille, and L. Fei-Fei. Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 82–92, 2019.

[104] C. Liu, B. Zoph, M. Neumann, J. Shlens, W. Hua, L.-J. Li, L. Fei-Fei, A. Yuille, J. Huang, and K. Murphy. Progressive neural architecture search. In *Proceedings of the European conference on computer vision (ECCV)*, pages 19–34, 2018.

[105] H. Liu, K. Simonyan, O. Vinyals, C. Fernando, and K. Kavukcuoglu. Hierarchical representations for efficient architecture search. In *International Conference on Learning Representations*, 2018.

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

[107] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[108] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022, 2021.

[109] Z. Liu, J. Ning, Y. Cao, Y. Wei, Z. Zhang, S. Lin, and H. Hu. Video swin transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3202–3211, 2022.

[110] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[111] C. Luo and A. L. Yuille. Grouped spatial-temporal aggregation for efficient action recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5512–5521, 2019.

[112] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European conference on computer vision (ECCV)*, pages 116–131, 2018.

[113] J. Mellor, J. Turner, A. Storkey, and E. J. Crowley. Neural architecture search without training. In *International Conference on Machine Learning*, pages 7588–7598. PMLR, 2021.

[114] Y. Meng, C.-C. Lin, R. Panda, P. Sattigeri, L. Karlinsky, A. Oliva, K. Saenko, and R. Feris. Ar-net: Adaptive frame resolution for efficient action recognition. In *European Conference on Computer Vision*, pages 86–104. Springer, 2020.

[115] Y. Meng, R. Panda, C.-C. Lin, P. Sattigeri, L. Karlinsky, K. Saenko, A. Oliva, and R. Feris. Adafuse: Adaptive temporal fusion network for efficient action recognition. In *International Conference on Learning Representations*, 2020.

[116] J. Mok, B. Na, J.-H. Kim, D. Han, and S. Yoon. Demystifying the neural tangent kernel from a practical perspective: Can it be trusted for neural architecture search without training? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11861–11870, 2022.

[117] G. F. Montufar, R. Pascanu, K. Cho, and Y. Bengio. On the number of linear regions of deep neural networks. *Advances in Neural Information Processing Systems*, 27:2924–2932, 2014.

[118] A. Neelakantan, L. Vilnis, Q. V. Le, I. Sutskever, L. Kaiser, K. Kurach, and J. Martens. Adding gradient noise improves learning for very deep networks. *arXiv preprint arXiv:1511.06807*, 2015.

[119] B. Ni, H. Peng, M. Chen, S. Zhang, G. Meng, J. Fu, S. Xiang, and H. Ling. Expanding language-image pretrained models for general video recognition. *arXiv preprint arXiv:2208.02816*, 2022.

[120] A. Oliver, A. Odena, C. A. Raffel, E. D. Cubuk, and I. Goodfellow. Realistic evaluation of deep semi-supervised learning algorithms. In *Advances in Neural Information Processing Systems*, pages 3235–3246, 2018.

[121] OpenAI. ChatGPT: OpenAI's Conversational AI Language Model. https://openai.com/blog/chatgpt, 2022.

[122] B. Pan, R. Panda, C. L. Fosco, C.-C. Lin, A. J. Andonian, Y. Meng, K. Saenko, A. Oliva, and R. Feris. Va-red^2: Video adaptive redundancy reduction. In *International Conference on Learning Representations*, 2020.

[123] W. Peng, X. Hong, and G. Zhao. Video action recognition via neural architecture searching. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 11–15. IEEE, 2019.

[124] H. Pham, M. Guan, B. Zoph, Q. Le, and J. Dean. Efficient neural architecture search via parameters sharing. In *International Conference on Machine Learning*, pages 4095–4104. PMLR, 2018.

[125] A. Piergiovanni, A. Angelova, A. Toshev, and M. S. Ryoo. Evolving space-time neural architectures for videos. In *Proceedings of the IEEE international conference on computer vision*, pages 1793–1802, 2019.

[126] S. Qian, H. Shao, Y. Zhu, M. Li, and J. Jia. Blending anti-aliasing into vision transformer. *Advances in Neural Information Processing Systems*, 34:5416–5429, 2021.

[127] S. Qian, Y. Zhu, W. Li, M. Li, and J. Jia. What makes for good tokenizers in vision transformer? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[128] Z. Qing, S. Zhang, Z. Huang, X. Wang, Y. Wang, Y. Lv, C. Gao, and N. Sang. Mar: Masked autoencoders for efficient action recognition. *arXiv preprint arXiv:2207.11660*, 2022.

[129] Z. Qiu, T. Yao, and T. Mei. Learning spatio-temporal representation with pseudo-3d residual networks. In *proceedings of the IEEE International Conference on Computer Vision*, pages 5533–5541, 2017.

[130] A. Quattoni and A. Torralba. Recognizing indoor scenes. *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 413–420, 2009.

[131] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR, 2021.

[132] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le. Regularized evolution for image classifier architecture search. In *Proceedings of the aaai conference on artificial intelligence*, volume 33, pages 4780–4789, 2019.

[133] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: towards real-time object detection with region proposal networks. *IEEE transactions on pattern analysis and machine intelligence*, 39(6):1137–1149, 2016.

[134] 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*, pages 10684–10695, 2022.

[135] M. S. Ryoo, A. Piergiovanni, A. Arnab, M. Dehghani, and A. Angelova. Tokenlearner: Adaptive space-time tokenization for videos. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

[136] 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*, pages 4510–4520, 2018.

[137] L. Sevilla-Lara, S. Zha, Z. Yan, V. Goswami, M. Feiszli, and L. Torresani. Only time can tell: Discovering temporal data for temporal modeling. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 535–544, 2021.

[138] G. A. Sigurdsson, S. Divvala, A. Farhadi, and A. Gupta. Asynchronous temporal fields for action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 585–594, 2017.

[139] G. A. Sigurdsson, G. Varol, X. Wang, A. Farhadi, I. Laptev, and A. Gupta. Hollywood in homes: Crowdsourcing data collection for activity understanding. In *European Conference on Computer Vision*, pages 510–526. Springer, 2016.

[140] P. Y. Simard, D. Steinkraus, J. C. Platt, et al. Best practices for convolutional neural networks applied to visual document analysis. In *Icdar*, volume 3, 2003.

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

[142] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

[143] C. Summers and M. J. Dinneen. Improved mixed-example data augmentation. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1262–1270. IEEE, 2019.

[144] C. Sun, A. Shrivastava, S. Singh, and A. Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *Proceedings of the IEEE international conference on computer vision*, pages 843–852, 2017.

[145] C. Sun, A. Shrivastava, C. Vondrick, K. Murphy, R. Sukthankar, and C. Schmid. Actor-centric relation network. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 318–334, 2018.

[146] Y.-L. Sung, V. Nair, and C. A. Raffel. Training neural networks with fixed sparse masks. *Advances in Neural Information Processing Systems*, 34:24193–24205, 2021.

[147] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[148] R. Takahashi, T. Matsubara, and K. Uehara. Ricap: Random image cropping and patching data augmentation for deep cnns. In *Asian Conference on Machine Learning*, pages 786–798, 2018.

[149] H. Tan, J. Lei, T. Wolf, and M. Bansal. Vimpac: Video pre-training via masked token prediction and contrastive learning. *arXiv preprint arXiv:2106.11250*, 2021.

[150] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, and Q. V. Le. Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2820–2828, 2019.

[151] M. Tan and Q. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pages 6105–6114. PMLR, 2019.

[152] M. Tan, R. Pang, and Q. V. Le. Efficientdet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10781–10790, 2020.

[153] H. Tanaka, D. Kunin, D. L. Yamins, and S. Ganguli. Pruning neural networks without any data by iteratively conserving synaptic flow. *Advances in Neural Information Processing Systems*, 33, 2020.

[154] A. Tarvainen and H. Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Advances in neural information processing systems*, pages 1195–1204, 2017.

[155] Z. Tong, Y. Song, J. Wang, and L. Wang. Videomae: Masked autoencoders are data-efficient learners for self-supervised video pre-training. *arXiv preprint arXiv:2203.12602*, 2022.

[156] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[157] 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*, pages 4489–4497, 2015.

[158] D. Tran, H. Wang, L. Torresani, and M. Feiszli. Video classification with channel-separated convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5552–5561, 2019.

[159] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri. A closer look at spatiotemporal convolutions for action recognition. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 6450–6459, 2018.

[160] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri. A closer look at spatiotemporal convolutions for action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6450–6459, 2018.

[161] C. Wang, G. Zhang, and R. Grosse. Picking winning tickets before training by preserving gradient flow. In *International Conference on Learning Representations*, 2019.

[162] M. Wang, J. Xing, and Y. Liu. Actionclip: A new paradigm for video action recognition. *arXiv preprint arXiv:2109.08472*, 2021.

[163] R. Wang, D. Chen, Z. Wu, Y. Chen, X. Dai, M. Liu, Y.-G. Jiang, L. Zhou, and L. Yuan. Bevt: Bert pretraining of video transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14733–14743, 2022.

[164] W. Wang et al. Image as a Foreign Language: BEiT Pretraining for All Vision and Vision-Language Tasks. *arXiv preprint arXiv:2208.10442*, 2022.

[165] W. Wang, E. Xie, X. Li, D.-P. Fan, K. Song, D. Liang, T. Lu, P. Luo, and L. Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 568–578, 2021.

[166] X. Wang, R. Girshick, A. Gupta, and K. He. Non-local neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7794–7803, 2018.

[167] X. Wang, X. Xiong, M. Neumann, A. Piergiovanni, M. S. Ryoo, A. Angelova, K. M. Kitani, and W. Hua. Attentionnas: Spatiotemporal attention cell search for video classification. In *European Conference on Computer Vision*, pages 449–465. Springer, 2020.

[168] Y. Wang, K. Lv, R. Huang, S. Song, L. Yang, and G. Huang. Glance and focus: a dynamic approach to reducing spatial redundancy in image classification. *Advances in Neural Information Processing Systems*, 33, 2020.

[169] C. Wei, H. Fan, S. Xie, C.-Y. Wu, A. Yuille, and C. Feichtenhofer. Masked feature prediction for self-supervised visual pre-training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14668–14678, 2022.

[170] B. Wu, X. Dai, P. Zhang, Y. Wang, F. Sun, Y. Wu, Y. Tian, P. Vajda, Y. Jia, and K. Keutzer. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10734–10742, 2019.

[171] B. Wu, A. Wan, X. Yue, P. Jin, S. Zhao, N. Golmant, A. Gholaminejad, J. Gonzalez, and K. Keutzer. Shift: A zero flop, zero parameter alternative to spatial convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9127–9135, 2018.

[172] C.-Y. Wu, M. Zaheer, H. Hu, R. Manmatha, A. J. Smola, and P. Krähenbühl. Compressed video action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6026–6035, 2018.

[173] W. Wu, Z. Sun, and W. Ouyang. Revisiting classifier: Transferring vision-language models for video recognition. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2023.

[174] W. Wu, X. Wang, H. Luo, J. Wang, Y. Yang, and W. Ouyang. Bidirectional cross-modal knowledge exploration for video recognition with pre-trained vision-language models. *arXiv preprint arXiv:2301.00182*, 2022.

[175] 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*, 32:7780–7789, 2019.

[176] 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*, pages 1278–1287, 2019.

[177] L. Xiao, J. Pennington, and S. Schoenholz. Disentangling trainability and generalization in deep neural networks. In *International Conference on Machine Learning*, pages 10462–10472. PMLR, 2020.

[178] S. Xie, C. Sun, J. Huang, Z. Tu, and K. Murphy. Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 305–321, 2018.

[179] Z. Xie, Z. Zhang, Y. Cao, Y. Lin, J. Bao, Z. Yao, Q. Dai, and H. Hu. Simmim: A simple framework for masked image modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9653–9663, 2022.

[180] H. Xiong, L. Huang, M. Yu, L. Liu, F. Zhu, and L. Shao. On the number of linear regions of convolutional neural networks. In *International Conference on Machine Learning*, pages 10514–10523. PMLR, 2020.

[181] Y. Xu, L. Xie, X. Zhang, X. Chen, G.-J. Qi, Q. Tian, and H. Xiong. Pc-darts: Partial channel connections for memory-efficient architecture search. In *International Conference on Learning Representations*, 2019.

[182] Y. Yamada, M. Iwamura, T. Akiba, and K. Kise. Shakedrop regularization for deep residual learning. *IEEE Access*, 7:186126–186136, 2019.

[183] S. Yan, X. Xiong, A. Arnab, Z. Lu, M. Zhang, C. Sun, and C. Schmid. Multiview transformers for video recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3333–3343, 2022.

[184] B. Yang, G. Bender, Q. V. Le, and J. Ngiam. Condconv: Conditionally parameterized convolutions for efficient inference. *Advances in Neural Information Processing Systems*, 32:1307–1318, 2019.

[185] L. Yang, Y. Han, X. Chen, S. Song, J. Dai, and G. Huang. Resolution adaptive networks for efficient inference. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2369–2378, 2020.

[186] T. Yang, S. Zhu, C. Chen, S. Yan, M. Zhang, and A. Willis. Mutualnet: Adaptive convnet via mutual learning from network width and resolution. In *European Conference on Computer Vision*, pages 299–315. Springer, 2020.

[187] T. Yang, S. Zhu, M. Mendieta, P. Wang, R. Balakrishnan, M. Lee, T. Han, M. Shah, and C. Chen. Mutualnet: Adaptive convnet via mutual learning from different model configurations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(1):811–827, 2021.

[188] C. Ying, A. Klein, E. Christiansen, E. Real, K. Murphy, and F. Hutter. Nas-bench-101: Towards reproducible neural architecture search. In *International Conference on Machine Learning*, pages 7105–7114. PMLR, 2019.

[189] J. Yu and T. S. Huang. Universally slimmable networks and improved training techniques. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1803–1811, 2019.

[190] J. Yu, P. Jin, H. Liu, G. Bender, P.-J. Kindermans, M. Tan, T. Huang, X. Song, R. Pang, and Q. Le. Bignas: Scaling up neural architecture search with big single-stage models. In *European Conference on Computer Vision*, pages 702–717. Springer, 2020.

[191] J. Yu, L. Yang, N. Xu, J. Yang, and T. Huang. Slimmable neural networks. In *International Conference on Learning Representations*, 2018.

[192] K. Yu, C. Sciuto, M. Jaggi, C. Musat, and M. Salzmann. Evaluating the search phase of neural architecture search. In *International Conference on Learning Representations*, 2019.

[193] L. Yuan, D. Chen, Y.-L. Chen, N. Codella, X. Dai, J. Gao, H. Hu, X. Huang, B. Li, C. Li, et al. Florence: A new foundation model for computer vision. *arXiv preprint arXiv:2111.11432*, 2021.

[194] L. Yuan, Y. Chen, T. Wang, W. Yu, Y. Shi, Z.-H. Jiang, F. E. Tay, J. Feng, and S. Yan. Tokens-to-token vit: Training vision transformers from scratch on imagenet. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 558–567, 2021.

[195] 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*, pages 6023–6032, 2019.

[196] S. Zagoruyko and N. Komodakis. Wide residual networks. In *British Machine Vision Conference 2016*. British Machine Vision Association, 2016.

[197] S. Zagoruyko and N. Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

[198] X. Zhai, A. Kolesnikov, N. Houlsby, and L. Beyer. Scaling Vision Transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

[199] B. Zhang, J. Yu, C. Fifty, W. Han, A. M. Dai, R. Pang, and F. Sha. Co-training transformer with videos and images improves action recognition. *arXiv preprint arXiv:2112.07175*, 2021.

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

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

[202] X. Zhang, P. Hou, X. Zhang, and J. Sun. Neural architecture search with random labels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10907–10916, 2021.

[203] X. Zhang, X. Zhou, M. Lin, and J. Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6848–6856, 2018.

[204] Y. Zhang, X. Li, C. Liu, B. Shuai, Y. Zhu, B. Brattoli, H. Chen, I. Marsic, and J. Tighe. Vidtr: Video transformer without convolutions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13577–13587, 2021.

[205] B. Zhou, A. Andonian, A. Oliva, and A. Torralba. Temporal relational reasoning in videos. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 803–818, 2018.

[206] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Learning deep features for discriminative localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2921–2929, 2016.

[207] J. Zhou, C. Wei, H. Wang, W. Shen, C. Xie, A. Yuille, and T. Kong. ibot: Image bert pre-training with online tokenizer. *arXiv preprint arXiv:2111.07832*, 2021.

[208] Y. Zhu, Z. Lan, S. Newsam, and A. Hauptmann. Hidden two-stream convolutional networks for action recognition. In *Computer Vision–ACCV 2018: 14th Asian Conference on Computer Vision, Perth, Australia, December 2–6, 2018, Revised Selected Papers, Part III 14*, pages 363–378. Springer, 2019.

[209] Y. Zhu, X. Li, C. Liu, M. Zolfaghari, Y. Xiong, C. Wu, Z. Zhang, J. Tighe, R. Manmatha, and M. Li. A comprehensive study of deep video action recognition. *arXiv preprint arXiv:2012.06567*, 2020.

[210] M. Zolfaghari, Y. Zhu, P. Gehler, and T. Brox. Crossclr: Cross-modal contrastive learning for multi-modal video representations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1450–1459, 2021.

[211] B. Zoph and Q. V. Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.

[212] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8697–8710, 2018.