TOWARDS EFFICIENT DEEP LEARNING: FROM COMPRESSION, SEARCH TO UNIFICATION

A Dissertation

by

JUNRU WU

Submitted to the Graduate and Professional School of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

March 2023

Major Subject: Computer Science

ABSTRACT

Deep learning has gained considerable interest due to its record-breaking performance in a variety of different domains, including computer vision, natural language processing, multimodal understanding, etc. Meanwhile, deep neural networks are usually parameter-heavy, inefficient, and highly specialized. As a result, there has been a growing demand to improve the efficiency and interoperability of deep neural networks motivated by different needs. In this dissertation, we proposed to address those problems via serial of approaches, including (a) reducing the memory storage and energy footprint via parameter sharing (b) improving the trade-off between performance and computation via neural architecture search (c) unifying neural architectures across different modalities via cross-modality gradient harmonization.

ACKNOWLEDGMENTS

Throughout the course of my Ph.D. studies and the writing of this dissertation, I have received an immense amount of support, for which I would like to express my heartfelt gratitude.

First and foremost, I would like to thank my Ph.D. advisor, Professor Zhangyang (Atlas) Wang, for his invaluable guidance, research insights, and dedicated supervision. His expertise, insightful comments, and inspirational thoughts have been instrumental in my academic journey.

I would also like to extend my gratitude to my dissertation committee members, Professor Shuiwang Ji, Professor Xiaoning Qian, and Professor Nima Kalantari, for their time, advice, and unwavering support.

Furthermore, I am grateful to my internship mentors across different industry research labs, in chronological order:

Google Research: Dr. Feng Han, Dr. Yi Liang, Dr. Tianqi Liu, Dr. Frederick Liu, Dr. Hongkun Yu, Dr. Jialu Liu, and Dr. Cong Yu. Microsoft Research: Dr. Dongdong Chen, Dr. Xiyang Dai, Dr. Yinpeng Chen, Dr. Mengchen Liu, and Dr. Lu Yuan. NEC Labs America: Dr. Xiang Yu, Dr. Buyu Liu, and Prof. Manmohan Chandraker. ByteDance AI Labs: Dr. Linjie Yang, Dr. Ding Liu, Dr. Chen Fang, and Dr. Jianchao Yang.

They provided me with extraordinary opportunities to conduct research in industry setting and give me access to extensive computational resources that greatly accelerate my research progress. I also want to thank all of my collaborators for their diligent efforts and invaluable advice.

Last but not least, I want to express my appreciation to my parents for their guidance and advice throughout my Ph.D. journey. Additionally, I am grateful to my friends, who provided a joyful distraction, allowing me to unwind and recharge outside of my research.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a dissertation committee consisting of Professor Zhangyang (Atlas) Wang, Professor Shuiwang Ji and Professor Nima Kalantari of the Department of Computer Science, and Professor Xiaoning Qian of the Department of Electrical and Computer Engineering.

All other work conducted for the thesis (or) dissertation was completed by the student independently.

Funding Sources

The Graduate study was supported by National Science Foundation (NSF), Defense Advanced Research Projects Agency (DAPRA), and gift funding from NEC Labs America.

TABLE OF CONTENTS

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

1. INTRODUCTION[1]

Deep learning has gained significant interest due to its record-breaking performance in various domains, including computer vision[10, 11, 12], natural language processing[13, 14, 15], and multimodal learning[16, 17, 6]. However, deep neural networks are often parameter-heavy[18, 10] and inefficient[13]. Consequently, there is a growing demand to enhance the efficiency of deep neural networks across multiple dimensions, motivated by the need to amortize computational costs during (pre-)training, reduce memory, computation, and energy footprints, and enable deployment on resource-constrained platforms.

In response to these demands, we aim to achieve efficient deep learning by combining efforts from various aspects, including (a) Neural Network Compression, (b) Neural Architecture Search, and (c) Neural Architecture Unification.

1.1 Neural Network Compression

The recent surge in edge computing has led to a tremendous need to deploy CNNs on resource-constrained mobile devices. Traditional neural networks, being large and inefficient, are not well-suited for such deployment. We propose using neural network compression techniques to bridge this gap, enabling the deployment of existing neural networks on resource-constrained devices.

1.2 Neural Architecture Search

Traditionally, neural networks have been designed manually[10, 18], yielding considerable performance but often being inefficient due to limited human expertise. Neural Architecture Search (NAS), on the other hand, aims to automate the architectural design process, ensuring an optimal trade-off between performance and efficiency by employing automated algorithms to search for the ideal architecture.

[1]Partial text and images of this chapter is reprinted with permission from "Deep k-means: Re-training and parameter sharing with harder cluster assignments for compressing deep convolutions." by Junru Wu et al. In International Conference on Machine Learning, pp. 5363-5372. PMLR, 2018., Copyright 2023 by Junru Wu.

1.3 Neural Architecture Unification

In the past, specialized neural architectures tailored for different modalities and tasks were required to achieve high performance. However, this approach is inherently inefficient when scaling up to accommodate the exponentially increasing number of modalities and tasks, as it necessitates training numerous specialized neural networks. With the recent advancements in transformer architectures, a unified transformer[6] has proven effective in tackling a variety of visual, audio, and language tasks. We refer to this new, unified approach as Neural Architecture Unification (NAU).

2. NEURAL NETWORK COMPRESSION[1]

2.1 Introduction

Convolutional neural networks (CNNs) have gained considerable interest due to their record-breaking performance in many recognition tasks [10, 11, 12]. In parallel, there has been a tremendously growing need to bring CNNs into resource-constrained mobile devices in line with the recent surge of edge computing in which raw data are processed locally in edge devices using their embedded machine learning algorithms [19] [20]. The advantage lies in that local processing avoids transferring data back and forth between data centers and edge devices, thus reducing communication cost, latency, and enhancing privacy. However, deploying CNNs into resource-constrained platforms is a non-trivial task. Devices at the edge, such as smart phones and wearables, have limited energy, computation and storage resources since they are battery-powered and have a small form factor. In contrast, powerful CNNs require a large number of weights that corresponds to considerable storage and memory bandwidth. For example, the amount of weights in state-of-the-art CNNs AlexNet and VGG-16 are over 200MB and 500MB, respectively [21]. Further, CNN-based applications can drain a battery very quickly if executed frequently. For example, smartphones nowadays cannot even run classification using AlexNet in real-time for more than one hour [22].

To close the gap between the constrained resources of edge devices and the growing complexity of CNNs, compression techniques have been widely investigated to reduce the precision of weights and the number of operations during or after CNN training in order to shrink their large implementation cost while maintaining the desired inference performance. Various CNN compression techniques have been proposed, such as weight compression [23] [21] [24] and decomposition [25] [26] [27], and compact architectures [28] [29]. However, there are two major shortcomings in existing CNN compression techniques.

[1]Partial text and images of this chapter is reprinted with permission from "Deep k-means: Re-training and parameter sharing with harder cluster assignments for compressing deep convolutions." by Junru Wu et al. In International Conference on Machine Learning, pp. 5363-5372. PMLR, 2018., Copyright 2023 by Junru Wu.

- A myriad of CNN compression techniques focus on the fully-connected layers of CNNs which conventionally have dominant parameters. However, recent successful CNNs tend to shift more parameters towards convolutional layers and have only one or even no fully-connected layers. For example, 85% of the parameters lie in the convolutional layers of GoogleNet [8]. Despite the growing trend toward CNN models using more convolutional layers and fewer fully-connected layers, only few compression techniques are dedicated for convolutional layers.

- The majority of CNN compression techniques, including most of the very few that focus on compressing convolutional layers, are designed to merely reduce the CNN model size or the amount of computation, which does not necessarily lead to reduced energy consumption. In fact, the recent work [22] argues that the number of weights, multiply-and-accumulate (MAC) operations, and speedup ratio are often not good approximations for energy consumption, which also heavily depends on memory data movement and more. For example, the authors show an interesting result that although SqueezeNet [28] has 51.8$\times$ fewer weights than AlexNet, it consumes 33% more energy. A compression technique that can be evaluated to reduce both model size and energy-aware complexity is hence highly desired for enabling extensive resource-constrained CNN applications.

2.1.1 Contribution

In this paper, we propose *Deep k-Means*, a compression pipeline that is well suited for trimming down the complexity of convolutional layers that dominate both the model size as well as energy consumption of recently developed state-of-the-art CNNs. *Deep k-Means* consists of two steps. First, a novel spectrally relaxed k-means regularization is developed to enforce highly clustered weight structures during re-training. After that, compression is performed via weight-sharing, by only recording cluster centers and weight assignment indexes. We evaluate the performance of *Deep k-Means* in comparison with several state-of-the-art compression techniques focused on compressing convolutional layers. The results show that *Deep k-Means* consistently achieves higher accuracy at the same compression ratio (CR) as its competitors. Furthermore, *Deep k-Means* is also evaluated in terms of energy-aware metrics developed by us, and its compressed models show favorable energy efficiency as well. Our main contributions are summarized as follows:

- We introduce a novel spectrally relaxed k-means regularization that automatically learns hard(er) assignments of convolutional layer weights during re-training, to favor the subsequent compression via k-means weight-sharing. Our regularization approach is effective, efficient, simple to implement and use, and easily scalable to large CNN models.

- Inspired by a recently developed dataflow called "row-stationary", that minimizes data movement energy consumption on CNN hardware implementation [30], we reformulate the weights into row vectors for weigh-sharing clustering. Such a formulation has the potential to result in CNN models that are in favor of energy-efficient hardware implementation.

- In order to bridge the gap between algorithm and hardware design of CNNs, we propose an improved set of energy-aware metrics based on [31]. Our energy consumption estimation results are verified to be consistent with those from the tool in [22], which was extrapolated from actual hardware measurements. We expect our metrics to broadly benefit future research in energy-aware CNN design.

2.1.2 Related Work

Parameter pruning and sharing has been used both to reduce network complexity and to avoid over-fitting. Early pruning approaches include Biased Weight Decay [32], Optimal Brain Damage [33], and Optimal Brain Surgeon [34]. Recent works [35] made use of the redundancy among neurons. The Deep Compression method introduced in [21] employed a three stage pipeline to prune the redundant connections, quantize the weights via scalar weight sharing, and then encode the quantized weights using Huffman coding. An effective soft weight-sharing method described in [9] showed competitive CRs on state-of-the-art CNNs, e.g., Wide ResNet.

With fully-connected layers traditionally considered as the memory bottleneck, numerous works focused on compressing these layers. For example, [36] proposed applying k-means clustering to the densely-connected layers and showed a good balance between model size and accuracy. [37] proposed HashedNet that used a low-cost hash function to group weights into hash buckets for parameter sharing. On the other hand, a few recent works embraced the trend towards more convolutional layers in CNNs and attempted to compress convolutional layers. For example, [8] proposed an architecture called FreshNets to compress filters of convolutional layers in the frequency domain. The recent work [1] iteratively pruned filters based on the classification accuracy reduction index, and achieved substantially higher classification accuracy compared to other structural compression schemes, e.g., [38, 39].

2.2 Proposed Approach

2.2.1 Parameter Sharing via Row-wise K-means

Assuming a convolutional layer $\in \mathbb{R}^{s \times s \times c \times m}$, where s denotes the filter size, c the input channel number, and m the output channel number. Following the convention in CNNs, we reshape it as a matrix $W \in \mathbb{R}^{s \times N}$, where $N = s \times c \times m$, each column vector $\in \mathbb{R}^{s}$ being <u>a row</u> from an original convolutional filter. Following the product quantization approach for fully-connected layers in [40], we treat all columns of W as N samples, and apply k-means to assign them with K clusters. When $K \ll N$, we need only to store the cluster indexes and codebooks after k-means, thus effectively compressing the convolution kernels. We show the detailed pipeline in Figure 2.1 and we define the *cluster rate* for each layer as $\frac{K}{N}$ here.

For compressing multiple convolutional layers, we adopt a "uniform parameter sharing" scheme for simplicity, i.e., each convolutional layer chooses its K value such that all layers have the same cluster rate, expect for the first layer whose cluster rate is often set higher.

We notice other alternatives to enforce structured parameter sharing among convolutional layers using k-means, e.g., reshaping each convolutional filter as vectors R^{s^2} and then clustering over $c \times m$ samples, or converting each output channel into $\mathbb{R}^{s \times s \times c}$ and clustering over resulting m

samples. In practice, we find their performance to be close (with k chosen in different proper ways). One major motivation for choosing the row-wise k-means is that it could lead to higher data reuse opportunity and thus result in more energy-efficient hardware implementations, according to the row-stationary dataflow recently proposed in [30], which has shown to be superior in terms of energy efficiency compared to other dataflows. Another motivation arises from reducing the complexity (see Section 2.2.2).

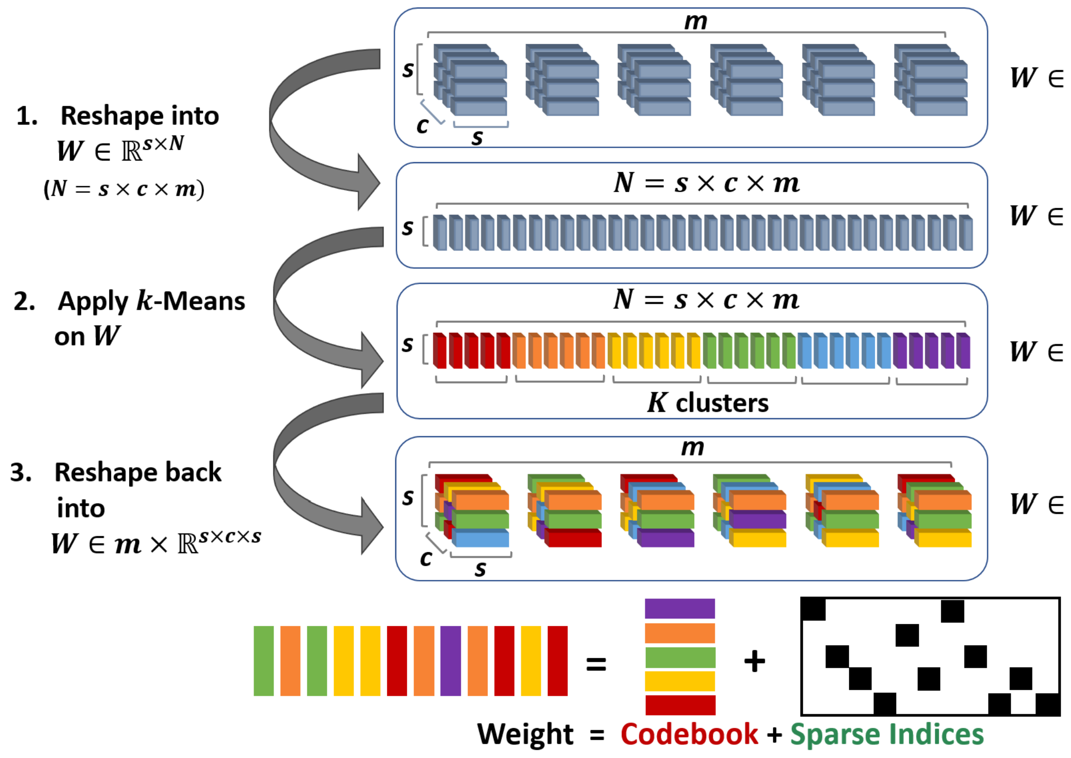

1. Reshape into $W \in \mathbb{R}^{s\times N}$ $(N = s \times c \times m)$
2. Apply k-Means on W
3. Reshape back into $W \in m \times \mathbb{R}^{s\times c\times s}$
m
s
c
$N = s \times c \times m$
K clusters
$W \in m \times \mathbb{R}^{s\times c\times s}$
$W \in \mathbb{R}^{s\times N}$
Weight = Codebook + Sparse Indices

Figure 2.1: An illustration of our proposed Deep-k-Means methods: (a) We first reshape it as a matrix $W \in \mathbb{R}^{s\times N}$, where $N = s \times c \times m$, each column vector $\in \mathbb{R}^s$ being a row from an original convolutional filter. (b) We then treat all columns of W as N samples, and apply k-means to assign them with K clusters. (c) After that, we reshape $W \in \mathbb{R}^{s\times N}$ back to its original shape. Since $K \ll N$, we need only to store the cluster indexes and codebooks after k-means, thus effectively compressing the convolution kernels.

2.2.2 K-means Regularized Re-Training

Simply pruning or sharing weights in CNNs will usually hurt the inference accuracy. Re-training has often been exploited to enforce the favorable structures in the pruned/shared weights and compensate for the accuracy loss [21]. In order to be compatible with k-means parameter sharing, we would favor a re-training scheme that "naturally" encourages the weights to be concentrated tightly around, or exactly at, a number of cluster components which are optimized for high predictive accuracy. The goal is fulfilled by introducing a novel *spectrally relaxed k-Means regularization* below.

The original sum-of-squares function of k-Means usually employs a Lloyd-type algorithm to solve. The spectral relaxation technique of k-Means was introduced in [41], by first equivalently re-formulating sum-of-squares into a trace form with special constraints. Specifically, to cluster N samples of $\mathbb{R}^s$, represented as $W \in \mathbb{R}^{s\times N}$, into K clusters, the spectral relaxation converts the k-means objective into the following problem:

$$\min_{W;F\in\mathcal{F}} Tr(W^TW) - Tr(F^TW^TWF), \tag{2.1}$$

where Tr denotes the matrix trace. $F \in \mathbb{R}^{N\times k}$ is the normalized cluster index matrix, and $\mathcal{F}$ denotes its special structure requirement: $F_{ij} = 1/\sqrt{n_j}$ if column i belongs to the cluster j and there is a total of n_j samples in the cluster j; and $F_{ij} = 0$ otherwise, $i = 1, ..., N$, $j = 1, ..., K$, and $\sum_{j=1}^{K} n_j = N$. The original spectral relaxation [41] considers W as given; thus (2.1) is reduced to:

$$\max_{F\in\mathcal{F}} Tr(F^TW^TWF) \tag{2.2}$$

The authors of [41] then proposed ignoring the special structure of F and let it be an arbitrary orthogonal matrix. (2.2) is thus relaxed to a trace maximization problem over a Stiefel manifold:

$$\max_{F} Tr(F^TW^TWF),\ s.t.\ F^TF = I \tag{2.3}$$

It results in a closed-form solution of F, by composing the first k singular vectors of W, according to the well-known Ky Fan theorem.

As a critical difference with [41], here our goal is not to cluster a *static* W. Rather, we would like to encourage W to stay "suited" for k-means during the dynamic re-training, without incurring a significant increase in complexity. We are thus motivated to utilize (2.1) as a regularization term on learning W, rather than a stand-alone objective. We discuss just one convolutional layer W for simplicity: assume that the original CNN training minimizes the energy function $E(W)$, w.r.t. W. The retraining minimizes the regularized objective below (λ is a scalar):

$$\begin{aligned} &\min_{W,F} E(W) + \frac{\lambda}{2}[Tr(W^T W) - Tr(F^T W^T W F)], \\ &s.t.\, F^T F = I \end{aligned} \tag{2.4}$$

Note that F is treated as an auxiliary variable to promote a clustered structure in W. Solving (2.4) could be iterated between the updates of W and F. Updating W can follow the standard stochastic gradient descent (SGD), with the gradient given as: $\nabla E(W) + \lambda W(I - FF^T)$. F is updated using the same closed-form solution to (2.3), by computing the k-truncated singular value decomposition (SVD) of W.

By the interaction between F and W during re-training, the regularization keeps W in a highly clustered state, in addition to optimizing it for inference accuracy. Although F has been relaxed from the "hard" normalized cluster index matrix to an arbitrary orthogonal one, we observe in practice that it still tends to enforce weights close to those taking around K unique vector values, i.e., encouraging "approximately hard" (or "harder" than soft weight sharing) K-cluster assignments during re-training.

In *Deep k-Means*, starting from an uncompressed pre-trained model as initialization, we will re-train it with adding this novel data-dependent weight regularizer (2.4) to each convolutional layer, while other training protocols remain unchanged. The re-training typically converges into a much smaller number of epochs than in the original training. After that, we apply row-wise k-means on the learned W for the final parameter-sharing step.

Complexity Analysis For each convolutional layer W, the extra complexity incurred by applying the spectrally relaxed k-means regularization term includes two parts: (1) updating W: the only extra burden is to compute $\lambda W(I - FF^T)$, which takes $\mathcal{O}(sKN)$ or $\mathcal{O}(s^2cmK)$ (computing WFF^T); (2) updating F via SVD, which costs $\mathcal{O}(s^2N)$ or $\mathcal{O}(s^3cm)$: that also serves another motivation to create W with lower row dimensions (e.g., s rather than s^2 or s^2c), since it will reduce the SVD complexity of W. Considering that s is usually small, the total extra complexity $\mathcal{O}((s^2K + s^3)cm)$ is quite affordable, enabling our methods to scale well for modern CNNs. In practice, we also implement the F update in a very "lazy" way so that SVD will merely be computed once for every five epochs, for further accelerating the re-training, with only marginal impacts on the result.

2.2.3 Comparison with Existing Work

Directly enforcing a k-means friendly weight structure is not straightforward. [9] presents an elegant and inspiring Bayesian regularization form of "soft cluster assignment". During retraining, the authors fit a Gaussian mixture model (GMM) prior model over the weights, to encourage the distribution of weights to be close to K clusters. After re-training, each weight was quantized to the mean of the GMM component that takes most responsibility, for parameter sharing. Their pipeline is the closet peer work to ours, with the major difference being that we pursue harder cluster assignment during re-training. As is well known, GMM is reduced to k-means when the mixture variance gets close to zero. Therefore, the retraining process in [9] could also be viewed as a "softened" version of k-means. However, the differences between the two methods manifest in multiple folds:

- First, our "harder" cluster assignment is directly derived from the original k-means objective (2.1). We expect it to be better aligned with the k-means parameter sharing stage. Our experimental observations show that this leads to more skewed weight distributions, and achieves better results than [9].
- Second, compared to the Bayesian form in [9], our regularization adds very little extra com-

plexity to the standard SGD. The implementation only calls for minor changes (a new regularizer term); and thanks to its low complexity, it is ready to be applied to larger-scale CNNs.

- Third, [9] discussed their high sensitivity to the choices of learning rates for mixture parameters (e.g., means, log-variances): a higher learning rate may cause model collapse and a lower one results in slow convergence. In contrast, *Deep k-Means* has merely one hyperparameter λ. We find *Deep k-Means* insensitive to λ (λ between 10^{-4} and 10^{-3} is found to work almost equally well). *Deep k-Means* needs no special learning rate scheduling. It is also free of postprocessing, e.g., removing redundant components as [9] needed to.

2.3 Energy-Aware Metrics for CNN Energy Consumption Estimation

While CR or reduction in the number of operations are widely adopted by existing CNN compression techniques as generic performance metrics, these metrics are not necessarily tied to improved energy efficiency as pinpointed by [22] according to their energy estimation tool extrapolated from actual hardware measurements. Therefore, it is important to evaluate compression techniques using a set of energy-aware metrics other than CR.

However, it is non-trivial to estimate the energy consumption of CNNs because a significant portion of energy consumption in CNNs is consumed by data movement, which mainly depends on the employed memory hierarchy and dataflow when implementing CNNs and is thus difficult to be estimated directly from the model. An energy estimation tool extrapolated from actual hardware measurements was proposed by [22] to bridge the gap between algorithm and hardware design. Unfortunately, their tool currently only supports AlexNet and GoogLeNetv1.

We hereby propose the following energy-aware metrics:

- **Computational cost** measures the computational resources needed to generate a single decision and is defined in terms of the number of 1 bit full adders (FAs), which is a canonical building block of arithmetic units. Specifically, assuming that the arithmetic operations are executed using the commonly employed ripple carry adder and BaughWooley multiplier architectures, the number of FAs needed to compute a D-dimensional dot product between the activations and weights is [42]:

$$DB_{\mathbf{w}}B_{\mathbf{x}} + (D-1)(B_{\mathbf{x}} + B_{\mathbf{w}} + \lceil \log_2(D) \rceil - 1) \tag{2.5}$$

where $B_{\mathbf{w}}$ and $B_{\mathbf{x}}$ denote the fixed-point precision assigned to the weights and activations, respectively.

- **Weight representational cost** measures the storage complexity and data movement costs corresponding to the weights and is defined as the product between the total number of bits needed to represent all weight parameters and the total number of times that the weights are used to compute convolutions:

  $$N_{\mathbf{w}} \left|\mathcal{W}\right| B_{\mathbf{w}} \tag{2.6}$$

  where $N_{\mathbf{w}}$ and $\mathcal{W}$ denote the total number of times that the weights are used to compute convolutions and the index sets of all weights in the network, respectively.

- **Activation representational cost** is similar to the weight representational cost above and is defined as:

  $$N_{\mathbf{x}} \left|\mathcal{X}\right| B_{\mathbf{x}} \tag{2.7}$$

  $N_{\mathbf{x}}$ and $\mathcal{X}$ denote the total number of times that the activations are used to compute convolutions and the index sets of all activations in the network, respectively.

The concepts of computational and representational costs were first proposed in [31] to describe network complexity. To better reflect the CNN energy cost, we modify the definition of representational cost in [31] to include the number of times that weights or activations are loaded for computing convolutions, in order to capture the associated data movement cost. Specifically, if a certain weight filter is removed due to compression, then the corresponding activation would be loaded less frequently, thus leading to reduced data movement costs. This can be reflected by the reduction of $N_{\mathbf{x}}$ in our modified definition but not the originally defined representational cost in [31]. Also, we separate the representational costs for the weights and activations to evaluate the impact of compression in more detail.

2.4 Experiments

We evaluate *Deep k-Means* in terms of CR and energy-aware metrics respectively, with the resulting accuracy loss Δ after compression, using two sets of experiments. The default λ is 10^{-4}. Unless otherwise specified, we will focus on compressing convolutional layers only.

For the first set of experiments on CR, we first create a simple baseline CNN for simulation experiments w.r.t. varying CR. The CR definition follows [21]. We then compare *Deep k-Means* against four latest and competitive comparison baselines for compressing convolutional layers: Ultimate Tensorization [7], FreshNet [8], Greedy Filter Pruning [1], and Soft Weight-Sharing [9]. The first three have been optimized towards compressing CNNs dominated by the convolutions and reported results on their self-designed models. [9] outperformed strong baselines such as [43] on the standard MNIST benchmark; the authors then reported compression results on the state-of-the-art Wide ResNet model [44] that mainly consist of convolutions. We also compare *Deep k-Means* with the baseline of *Deep k-Means* without re-training (*Deep k-Means WR*), i.e., directly performing row-wise k-means on original weights.

For the second set of experiments, we first validate the estimated energy consumption using our metrics to match the actual hardware-based extrapolation [22], and then evaluate *Deep k-Means* against the aforementioned baselines from an energy consumption perspective. While it is overall challenging to estimate energy consumption accurately due to the multitude of factors involved, our proposed metrics are simple, effective (i.e., showing a good match with the results extrapolated by actual hardware measurement), and thus can help CNN model designers understand various design trade-offs. We also provide insights regarding the impact of different types (i.e., parameter-sharing or pruning) of compression techniques on the computational and representational costs in Eqs. (2.5), (2.6) and (2.7).

2.4.1 Comparison on Compression Ratio

2.4.1.1 Comparison with Ultimate Tensorization

Model	Δ (%)	CR
TT-conv (naive)	-2.4	2.02
TT-conv (naive)	-3.1	2.90
TT-conv	-0.8	2.02
TT-conv	-1.5	2.53
TT-conv	-1.4	3.23
TT-conv	-2.0	4.02
Deep k-Means	+0.05	2
Deep k-Means	-0.04	4

Table 2.1: Compressing TT-conv-CNN in [7]. Reprinted with permission from [2].
.

[7] proposed a tensor factorization based method specifically for compressing convolutional layers. The authors proposed a Tensor Train (TT) Decomposition approach for convolutional kernel, denoted as *TT-conv (naive)*. It could be further enhanced by introducing a new type of TT-conv layer, denoted as *TT-conv*. The authors evaluated TT-conv (naive) and TT-conv on a self-designed architecture, called TT-conv-CNN, consisting of six convolutional layers and one fully-connected layer. TT-conv-CNN is dominated by the convolutions (occupying 99.54% parameters of the network), and the authors reported the uncompressed model's top-1 accuracy of 90.7% on CIFAR-10 [45].

We evaluate *Deep k-Means* on the TT-conv-CNN model at CR = 2 and 4. Table 2.1 compares them with the compression results in [7]. *Deep k-means* incurs minimal accuracy loss even at CR = 4. More surprisingly, it even slightly increases the accuracy after compression at CR = 2. It concurs with the previous observations by [9, 46]: removing parameter redundancy improves CNN generalization on some small networks.

2.4.1.2 Comparison with FreshNet

The Frequency-Sensitive Hashed Nets (FreshNets) was proposed in [8] to exploit inherent redundancy in convolutional layers. The authors observed that convolutional weights to be typically smooth and low-frequency. They were thus motivated to first convert filter weights to the frequency domain, after which they group frequency parameters into hash buckets to achieve parameter sharing. The authors evaluated their method on their self-designed CNN (referred to as *FreshNet-CNN* hereinafter) consisting of five convolutional layers and one fully-connected layer. They reported the uncompressed FreshNet-CNN to obtain the top-1 accuracy of 85.09% on CIFAR-10.

Table 2.2 reports the compression results of *Deep k-Means* and *Deep k-Means WR* on FreshNet-CNN at CR = 16. We also include two original baselines in [8]: low-rank decomposition (LRD) [47] and HashedNet [37]. In this example, even the accuracy of *Deep k-Means WR* is very competitive. After re-training, *Deep k-Means* shows a sharp further improvement.

Model	Δ (%)	CR
LRD	-8.32	16
HashedNet	-9.79	16
FreshNet	-6.51	16
Deep k-Means WR	-5.95	16
Deep k-Means	-1.30	16

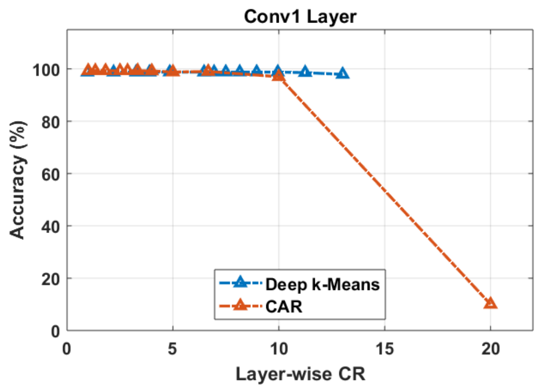
Table 2.2: Compressing FreshNet-CNN in [8]. Reprinted with permission from [2].

.

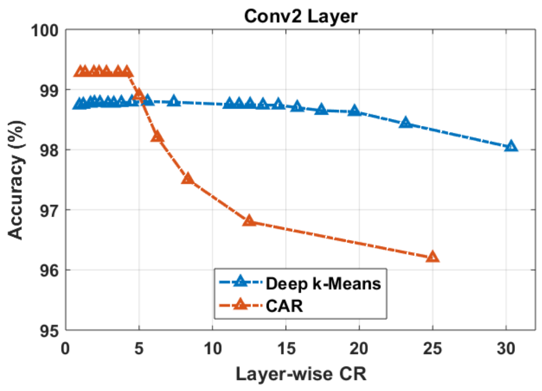
2.4.1.3 Comparison with Greedy Filter Pruning

The recent work [1] introduced a greedy structural compression scheme that prunes redundant convolutional filters in a trained CNN, based on a classification accuracy reduction (CAR) algorithm. The authors reported promising results on LeNet-5, AlexNet and ResNet-50, and their evaluations adopted a unique layer-wise compression fashion: taking LeNet-5 for example, each time the authors pruned filters in one convolutional layer (first or second) while leaving other layers untouched, and then reported the overall accuracy.

We compare *Deep k-Means* with CAR (with re-training, the best performer in [1]) using LeNet-5 on MNIST, and follow their layer-wise compression setting. Thus, unlike our other experiments, we report the accuracy w.r.t. “layer-wise” CR, i.e., measuring how many times the current layer is compressed, rather than the overall CR that measures the entire model. As Figure 2.2 shows, both *Deep k-Means* and CAR produce similar results at small layer-wise CRs; CAR is more competitive at small CRs for Conv2. However, *Deep k-Means* is clearly superior at high layer-wise CRs for both layers.



(a) Comparison in the first convolutional layer



(b) Comparison in the second convolutional layer

Figure 2.2: Compressing LeNet following the layer-wise setting in [1]: (a) The overall classification accuracy of LeNet when only the first convolutional layer (Conv1) is compressed, w.r.t. layer-wise CR; (b) The overall classification accuracy of LeNet when only the second convolutional layer (Conv2) is compressed, w.r.t. layer-wise CR. Reprinted with permission from [2].

2.4.1.4 Comparison with Soft Weight Sharing

[9] reported the compression performance of soft weight-sharing on the state-of-the-art Wide ResNet model [44], a convolution-dominant CNN with 2.7M parameters, at one single CR = 45 using CIFAR-10 (the uncompressed baseline top-1 error is 6.48%). Thanks to the light computational burden of *Deep k-Means*, we are able to evaluate various CRs. Note that at the same CR,

Model	Δ (%)	CR
Soft Weight-Sharing	-2.02	45
Deep k-Means WR	-16.02	45
Deep k-Means WR	-25.45	47
Deep k-Means WR	-45.08	50
Deep k-Means	-1.63	45
Deep k-Means	-2.23	47
Deep k-Means	-4.49	50

Table 2.3: Compressing Wide ResNet in comparison to soft weight-sharing [9]. Reprinted with permission from [2].

.

soft weight-sharing and *Deep k-Means* will lead to identical layer-wise dimensions and the same number of unique weights in each layer. Thus, their performance difference can only arise from the effects of their different regularization ways during re-training.

Promoting Sparsity in Re-Training. During the review stage, *one anonymous reviewer* commented that [9] tried to explicitly enforce weight values to a cluster centered at zero, while the above default routine of *Deep k-Means* had not such constraint. Such a sparsity-promotion operation may marginally decrease compression performance as it restricts the flexibility of setting centroids, but can gain more in both speedup and energy savings [48]. To ensure a fair comparison with [9], we implement a similar sparsity-promoting feature for *Deep k-Means*, *in this specific experiment only*. Without referring to sophisticated options such as semi-supervised clustering [49], we follow a simple heuristic which incurs almost no extra complexity: at each time of "lazy update" for layer $W \in \mathbb{R}^{s \times N}$, we first rank all N columns of W in terms of their ℓ_2 norms. We then assign the pN ($0 < p < 1$) smallest-norm columns to one cluster with a fixed center at zero, before solving (2.4). At the parameter-sharing step, we similarly threshold pN smallest-norm columns in W to be all-zero, and then perform $(k-1)$-clustering for remaining columns. The group of layer-wise p that we used for all 16 layers is: [0, 0.3, 0.4, 0.5, 0.4, 0.4, 0.5, 0.5, 0.5, 0.5, 0.5, 0.6, 0.9, 0.5, 0.75, 0.9].

Table 2.3 demonstrates the superiority of *Deep k-Means* (with the above-described sparsity promotion) over [9], by comparing their top-1 accuracy drops: 1.63% versus 2.02 %, at CR = 45. We further display the results at CR = 47 and 50, with a smooth accuracy decrease.

2.4.1.5 Evaluation with GoogleNet on ImageNet

We finally evaluate *Deep k-Means* on the GoogleNet [50] trained with the ImageNet ILSVRC12 dataset [51]. We use single center crop during testing, and evaluate the performance based on the *top-1* and *top-5* accuracy drops on the validation set, compared to the uncompressed baseline whose *top-1* accuracy is 69.76% and *top-5* 89.63%. We include two comparison methods: one-shot network compression [52], and low-rank regularization [53]. According to Table 2.4, *Deep k-Means* proves to scale well on large models/datasets, and achieves significantly better results over the two baselines: its compression at CR ≤ 3 is almost lossless, with top-5 errors again observed to slightly increase after compression. The GoogleNet compression performance is found to deteriorate quickly when CR > 4.

Model	$\Delta^{\dagger}$ %	$\Delta^{\ddagger}$ %	CR
One-shot [52]	N/A	-0.24	1.28
Low-rank [53]	N/A	-0.42	2.84
Deep k-Means WR	-1.22	-0.65	1.5
Deep k-Means WR	-3.7	-2.46	2
Deep k-Means WR	-13.72	-10.05	3
Deep k-Means WR	-48.95	-48.82	4
Deep k-Means	-0.26	0.00	1.5
Deep k-Means	-0.17	+0.06	2
Deep k-Means	-0.36	+0.03	3
Deep k-Means	-1.95	-1.14	4

Table 2.4: Compressing GoogLeNet on ILSVRC12 († and ‡ are top-1 and top-5 accuracies respectively). Reprinted with permission from [2].

2.4.2 Comparison on Energy-Aware Metrics

2.4.2.1 Energy-Aware Metrics Verification

We first evaluate our energy-aware metrics by comparing its estimated energy consumption with that of the tool in [22]. Note that the unit of energy: 1) in [22] is normalized in terms of number of MAC operations while the computational cost in Eq. (2.5) is normalized in terms of number of FAs; and 2) for the representational cost in Eqs. (2.6) and (2.7) is different from that of the computational cost in Eq. (2.5). Therefore, we first normalize the representational cost in terms of the computational cost assuming that a global on-chip buffer is employed, implying that the representational cost of a MAC is about 6 times that of performing a MAC computation [30]. This normalized representational cost is then added to the computational cost to obtain our total energy. Lastly, we normalize this total energy in terms of the number of MACs to be the same as that of [30].

We calculate the coefficient of determination (R^2), between the estimated energy consumptions using our proposed metrics, and using the tool in [22], of the same compressed models. We use *Deep k-means* to compress both AlexNet and GoogLeNetv1 [2], which are the *only two CNN models* currently supported by [22]. The energy consumptions are estimated as we choose the *cluster rate* to vary between: (AlexNet) [0.5, 0.3, 0.25, 0.2, 0.15, 0.1, 0.05, 0.01], and (GoogLeNetv1) [0.33, 0.18, 0.05, 0.012], respectively, to ensure negligible accuracy loss. We have R^2 to be **0.9931** for AlexNet, and **0.9675** for GoogLeNetv1, suggesting the estimated energy consumptions using our proposed metrics to be *strongly linearly correlated* with the results extrapolated from actual hardware measurements [22]. Yet different from their tool, our metrics are generally applicable to any CNN.

[2]For both networks, we employ our proposed methods in convolutional layers only. For AlexNet, we only quantize weight and activation to 8 bit and to 16 bit in fully-connected layers, respectively. For GoogLeNetv1, we use the one with global average pooling, which has no fully-connected layer.

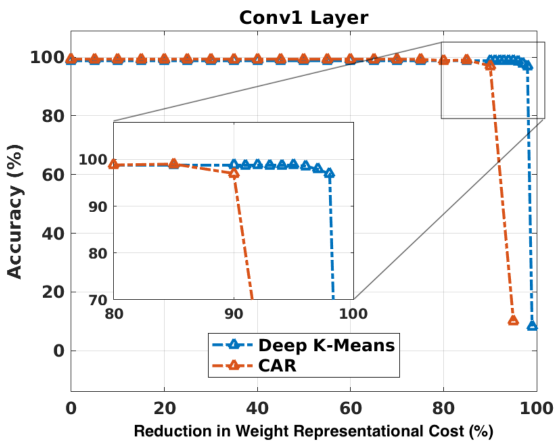
2.4.2.2 Comparison with Greedy Filter Pruning

An ideal CNN model to be deployed on resource-constrained platforms should simultaneously possess compact model size and low energy cost. In general, the resulting computational/representational cost reduction via compression depends on how the network is trimmed, i.e., which parts of the network is compressed. Conceptually, we point out that different compression schemes (e.g., parameter-sharing versus pruning) will affect the analysis of the computational and representational costs defined in Eqs. (2.5), (2.6) and (2.7). First, the weight representational cost is directly proportional to CR in both parameter-sharing (e.g. *Deep k-Means* and soft weight-sharing) and pruning (e.g. CAR) cases, because they both in effect can reduce the term $|\mathcal{W}|$ in (2.6). Second, the activation representational cost is proportional to CR for the case of weight pruning, but is independent of CR if the compression is done by weight sharing. This is because weight pruning results in skipping of the corresponding computations and thus can reduce the number of times that the corresponding activations are used (i.e., $\mathcal{W}$ in (2.7)), whereas there is no computation or connection skipping in the case of weight-sharing. Third, the computational cost is again proportional to CR for weight pruning; yet it would become input-dependent when it comes to weight sharing. Specifically, only when all the weights corresponding to the same input/activation are shared, the computational cost reduction ratio becomes equal to CR.

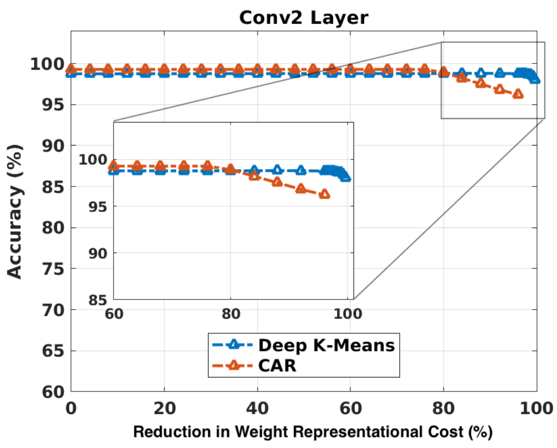
Deep k-Means has constantly obtained the best CR performance among the aforementioned baselines. To provide a concrete example, we choose the CAR (with re-training) baseline in [1], which produces slightly inferior but still competitive CR results, and discuss its potential energy efficiency improvement compared with *Deep k-Means*. The same layer-wise compression setting in Section 2.4.1.3 is adopted, for the first two convolutional layers of LeNet-5. A similar analysis could be done for other methods too.

Before we compare the impact of Deep k-Means and the baselines on the computational and representational costs, we discuss the relationship between CR and these energy-aware metrics to provide conceptual insights.

First, the weight representational cost is directly proportional to CR in both weight-sharing (e.g.



(a) Comparison in the first convolutional layer



(b) Comparison in the second convolutional layer

Figure 2.3: Comparison between *Deep k-Means* and CAR, in terms of the ratio between weight representational cost reduction. Reprinted with permission from [2].

Deep k-Means and soft weight-sharing) and weight pruning (e.g. CAR) cases, because they both in effect can reduce the term $|\mathcal{X}|$ in (2.6). Second, the activation representational cost is directly proportional to CR for the case of compression via weight pruning, and is independent of CR for the case of compression via weight sharing. This is because weight pruning results in skipping of the corresponding computations and thus can reduce the number of times that the corresponding activations are used (i.e., $N_{\mathbf{x}}$ in (2.7)), whereas there is no computation or connection skipping in the case of weight-sharing. Third, the computational cost is directly proportional to CR when using the compression is performed through weight pruning, and it would depends on the positions of shared weights for the case of compression through weight sharing. Specifically, only when all the weights corresponding to the same input/activation can be shared, the ratio on the computational cost reduction over the original model is equal to CR.

Deep k-Means compresses the network via weight sharing, whereas CAR relies on weight pruning. The accuracy versus CR comparison (i.e., Figure 2.2) in Section 2.4.1.3 shows that *Deep k-Means* is clearly superior to CAR at high layer-wise CRs, for either of the two convolutional layers. The potential energy consumption comparison between *Deep k-Means* and CAR in terms of the three metrics are as follows[3].

First, *Deep k-Means* will achieve higher weight representational cost reduction since it mostly offers higher CR with the same or even better accuracy, in particular at high CRs. Figure 2.3 (a) and (b) compares the accuracy versus weight representational cost reduction ratio of *Deep k-Means* and CAR for compressing the first and second convolutional layers in LeNet 5, respectively[4]. We observe that *Deep k-Means* achieves about 10% and 17.2% higher weight representational cost reduction, respectively, compared to CAR when compressing the first and second layers, without incurring accuracy loss. Second, CAR always outperforms *Deep k-Means* in terms of activation representational cost, because it removes filters and thus reduces the numbers of feature maps. In theory, CAR can achieve up to (layer-wise) "CR times" better activation representational cost reduction ratio than *Deep k-Means*. Third, the achievable computational cost reduction by *Deep k-Means* is either smaller or equal to that of CAR, depending on the inputs.

2.5 Conclusion and Discussions

We proposes *Deep k-Means*, a retraining-then-parameter-sharing pipeline for compressing convolutional layers in deep CNNs. A novel spectrally relaxed k-means regularization is derived to make hard assignments of convolutional layer weights to learned cluster centers during re-training. *Deep k-Means* demonstrates clear superiority over several recently-proposed competitive methods, in terms of both compression ratio and energy efficiency. Our future work will exploit more adaptive cluster rates for different layers instead of the current uniform scheme. Based on our proposed metrics, we also aim to incorporate more energy-aware regularizations into *Deep k-Means* for direct minimization of energy consumptions.

[3]We are unable to directly verify the total energy consumption of CAR using our metrics due to the lack of their model parameters or pre-trained model.

[4]We did not consider the cost of weight assignment indexes as it is negligible due to achievable high CR.

3. NEURAL ARCHITECTURE SEARCH[1]

3.1 Introduction

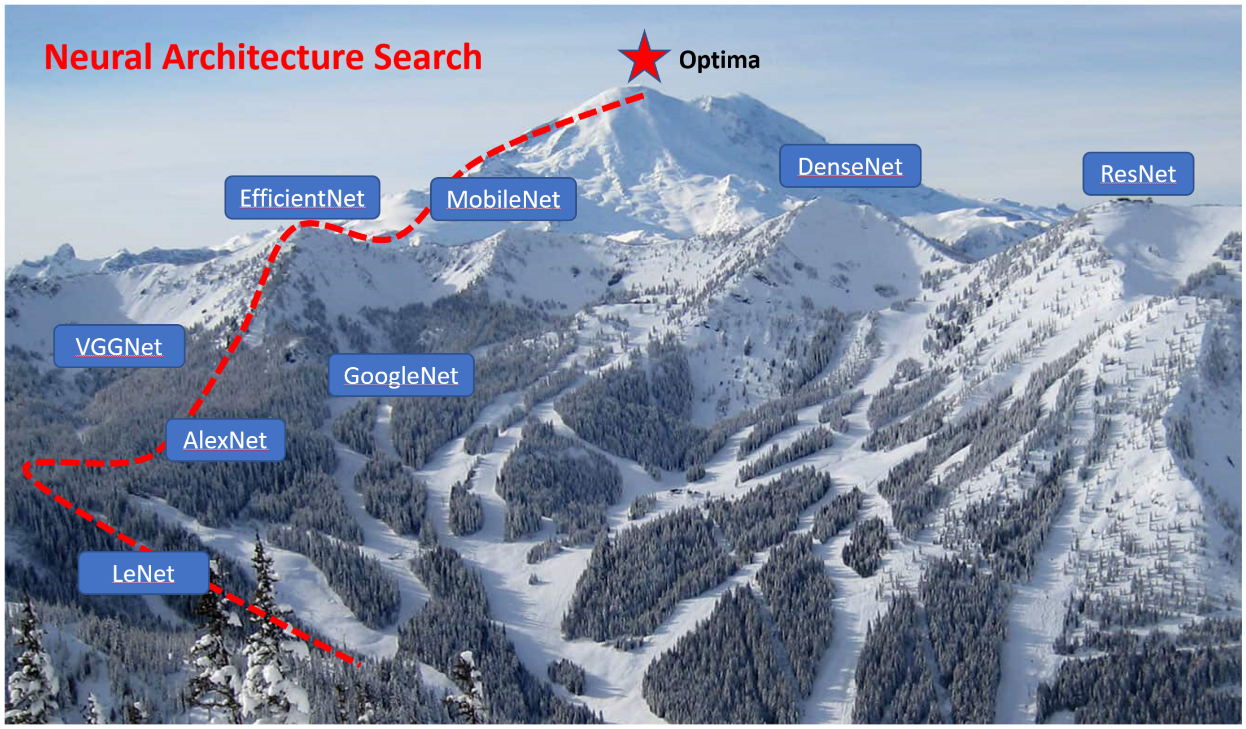


Figure 3.1: An analogy can be drawn between the Neural Architecture Search (NAS) process and mountaineering. To illustrate this, we use Attitude to represent the performance of Neural Architectures and Latitude/Longitude coordinates to depict the distance between different Neural Architectures. For example, the coordinate of ResNet is closer to DenseNet than that of MobileNet. The rugged terrain of the mountain reflects the complexity of the neural architecture subspace. The main objective of NAS is to explore this intricate subspace and identify the path that leads to the optimal architecture.

Neural Architecture Search (NAS)[54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65] methods aim to identify the best network architecture by exploring the relationship between architecture

[1]Partial text and images of this chapter is reprinted with permission from "Stronger nas with weaker predictors." by Junru Wu et al. In Advances in Neural Information Processing Systems 34 (2021): 28904-28918., Copyright 2023 by Junru Wu.

and performance. This is achieved by searching the architecture-to-performance manifold, which can be compared to a mountaineering expedition, as illustrated in Figure 3.1. In this analogy, performance is analogous to altitude, while the distance between different neural architectures is represented by the latitude and longitude coordinates. For instance, the coordinate of ResNet is closer to DenseNet than MobileNet. The rugged terrain of the mountain symbolizes the complexity of the neural architecture subspace. The goal of NAS is to navigate this intricate terrain and discover the path that leads to the optimal architecture.

Some common approaches in NAS includes reinforced-learning-based [66], evolution-based [67, 68] or gradient-based [54, 69] methods. However, in order to cover the entire search space, they often train and evaluate a large number of architectures, leading to tremendous computation cost. Recently, predictor-based NAS methods alleviate this problem with two key steps: one sampling step to sample some architecture-performance pairs, and another performance modeling step to fit the performance distribution by training a proxy accuracy predictor. An in-depth analysis of existing methods [55] found that most of those methods [58, 59, 70, 60, 61, 62, 71] consider these two steps independently and attempt to model the performance distribution over the whole architecture space using a ***strong***[2] predictor. However, since the architecture space is often exponentially large and highly non-convex, even a very strong predictor model has difficulty fitting the whole space given limited samples. Meanwhile, different types of predictors often demand handcraft design of the architecture representations to improve their performance.

This paper reflects on a fundamental question for predictor-based NAS: *"if our final goal is to find the best architecture, do we really need to model the whole space well?"*. We investigate the alternative of utilizing a few ***weak***[2] predictors to fit small local spaces, and to progressively move the search space towards the subspace where good architecture resides. Intuitively, we assume the whole space could be divided into different sub-spaces, some of which are relatively good while

[2]"Strong" vs "Weak" predictors: we name a "weak" predictor if it only predicts a local subspace of the search space thus can be associated with our iterative sampling scheme; such predictors therefore usually do not demand very heavily parameterized models. On the contrary, "strong" predictors predict the global search space and are often associated with uniform sampling. The terminology of strong versus weak predictors does not represent their number of parameters or the type of NAS predictor used. An overparameterized NAS predictor with our iterative sampling scheme may still be considered as a "weak" predictor.

some are relatively bad. We tend to choose the good ones while discarding the bad ones, which makes sure more samples will be focused on modeling only the good subspaces and then find the best architecture. It greatly simplifies the learning task of each predictor. Eventually, a line of progressively evolving weak predictors can connect a path to the best architecture.

We present a novel, general framework that requires only to estimate a series of weak predictors progressively along the search path, we denoted it as **WeakNAS** in the rest of the paper. To ensure moving towards the best architecture along the path, at each iteration, the sampling probability of better architectures keep increasing through the guidance of the previous weak predictor. Then, the consecutive weak predictors with better samples will be trained in the next iteration. We iterate until we arrive at an embedding subspace where the best architectures reside and can be accurately assessed by the final weak predictor.

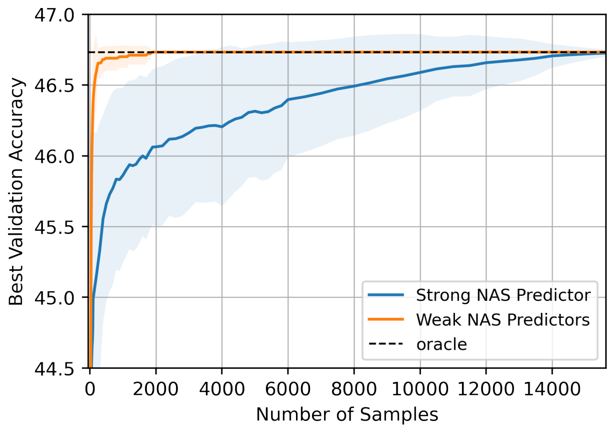


Figure 3.2: Comparison between our method using a set of weak predictors (iterative sampling), and a single strong predictor (random sampling) on NAS-Bench-201. For fair comparison, the NAS predictor in both methods adtops the same type of MLP described in 3.2.4. Solid lines and shadows denote the mean and standard deviation (std), respectively. Reprinted with permission from [3].

Compared to the existing predictor-based NAS, our proposal represents a new line of attack and has several merits. First, since only weak predictors are required, it yields better sample efficiency. As shown in Figure 3.2, it costs significantly fewer samples to find the top-performance

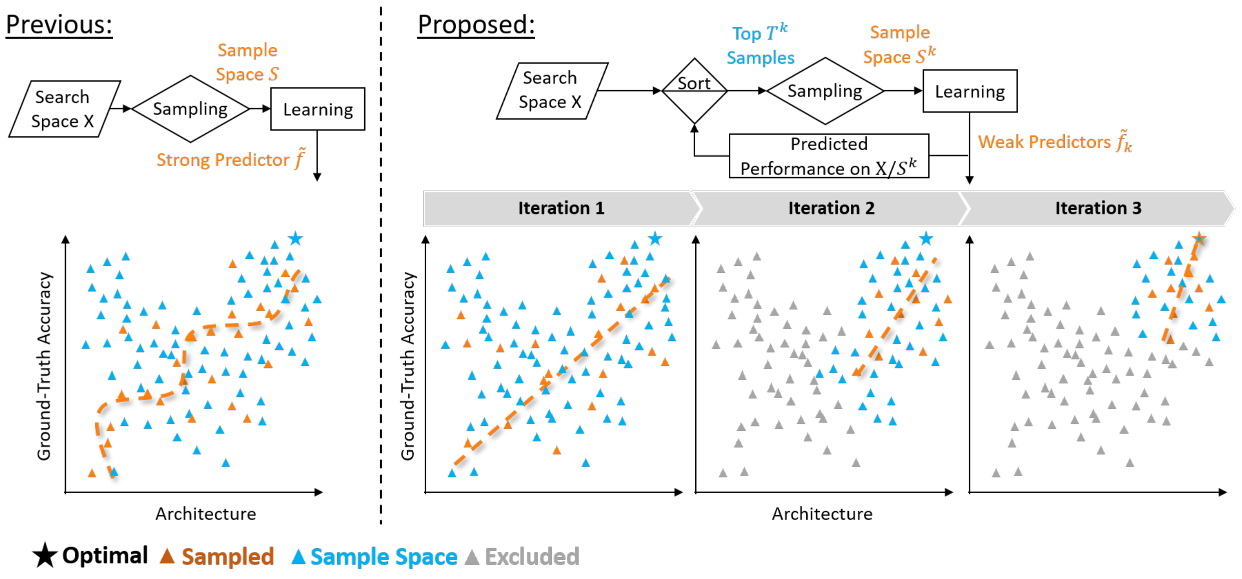


Figure 3.3: An illustration of WeakNAS's progressive approximation. Previous predictor-based NAS uniformly sample in the whole search space to fit a strong predictor. Instead, our method progressively shrinks the sample space based on predictions from previous weak predictors, and update new weak predictors towards subspace of better architectures, hence focusing on fitting the search path. Reprinted with permission from [3].

architecture than using one strong predictor, and yields much lower variance in performance over multiple runs. Second, it is flexible to the choices of architecture representation (e.g., different architecture embeddings) and predictor formulation (e.g., multilayer perceptron (MLP), gradient boosting regression tree, or random forest). Experiments show our framework performs well in all their combinations. Third, it is highly generalizable to other open search spaces, e.g. given a limited sample budget, we achieve the state-of-the-art ImageNet performance on the NASNet and MobileNet search spaces. Detailed comparison with state-of-the-art predictor-based NAS [72, 73, 74, 61] is presented in Section 4.

3.2 Our Framework

3.2.1 Reformulating Predictor-based NAS as Bi-Level Optimization

Given a search space of network architectures X and an architecture-to-performance mapping function $f : X \to P$ from the architecture set X to the performance set P, the objective is to find the best neural architecture x^* with the highest performance $f(x)$ in the search space X:

$$x^* = \arg\max_{x \in X} f(x) \tag{3.1}$$

A naïve solution is to estimate the performance mapping $f(x)$ through the full search space. However, this is prohibitively expensive since all architectures have to be exhaustively trained from scratch. To address this problem, predictor-based NAS learns a proxy predictor $\tilde{f}(x)$ to approximate $f(x)$ by using some architecture-performance pairs, which significantly reduces the training cost. In general, predictor-based NAS can be re-cast as a bi-level optimization problem:

$$x^* = \arg\max_{x \in X} \tilde{f}(x|S), \ \text{ s.t. } \tilde{f} = \arg\min_{S, \tilde{f} \in \tilde{\mathcal{F}}} \sum_{s \in S} \mathcal{L}(\tilde{f}(s), f(s)) \tag{3.2}$$

where $\mathcal{L}$ is the loss function for the predictor $\tilde{f}$, $\tilde{\mathcal{F}}$ is a set of all possible approximation to f, $S := \{S \subseteq X \mid |S| \leq C\}$ all architectures satisfying the sampling budget C. C is directly related to the total training cost, e.g., the total number of queries. Our objective is to minimize the loss $\mathcal{L}$ based on some sampled architectures S.

Previous predictor-based NAS methods attempt to solve Equation 3.2 with two sequential steps: (1) *sampling* some architecture-performance pairs and (2) *learning* a proxy accuracy predictor. For the first step, a common practice is to sample training pairs S uniformly from the search space X to fit the predictor. Such sampling is however inefficient considering that the goal of NAS is only to find well-performed architectures without caring for the bad ones.

3.2.2 Progressive Weak Predictors Emerge Naturally as A Solution to the Optimization

Optimization Insight: Instead of first (uniformly) sampling the whole space and then fitting the predictor, we propose to jointly evolve the sampling S and fit the predictor $\tilde{f}$, which helps achieve better sample efficiency by focusing on only relevant sample subspaces. That could be mathematically formulated as solving Equation 3.2 in a new coordinate descent way, that iterates between optimizing the architecture *sampling* and predictor *fitting* subproblems:

$$\text{(Sampling)} \quad \tilde{P}^k = \{\tilde{f}_k(s) | s \in X \setminus S^k\},\ S_M \subset \text{Top}_N(\tilde{P}^k),\ S^{k+1} = S_M \cup S^k, \quad \text{where } \text{Top}_N(\tilde{P}^k) \text{ denote the set of top N architectures in } \tilde{P}^k \tag{3.3}$$

$$\text{(Predictor Fitting)} \quad x^* = \underset{x \in X}{\arg\max}\, \tilde{f}(x|S^{k+1}),\ \text{s.t. } \tilde{f}_{k+1} = \underset{\tilde{f}_k \in \tilde{\mathcal{F}}}{\arg\min} \sum_{s \in S^{k+1}} \mathcal{L}(\tilde{f}(s), f(s)) \tag{3.4}$$

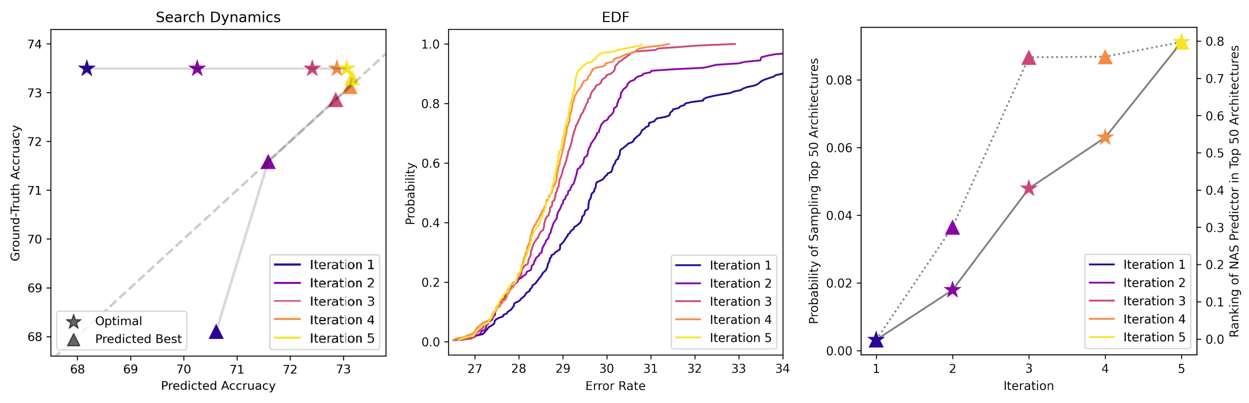
In comparison, existing predictor-based NAS methods could be viewed as running the above coordinate descent *for just one iteration* – a special case of our general framework.

As well known in optimization, many iterative algorithms only need to solve (subsets of) their subproblems inexactly [75, 76, 77] for properly ensuring convergence either theoretically or empirically. Here, using a strong predictor to fit the whole space could be treated as solving the predictor fitting subproblem relatively precisely, while adopting a weak predictor just imprecisely solves that. Previous methods solving Equation 3.2 truncate their solutions to "one shot" and hinge on solving subproblems with higher precision. Since we now take a joint optimization view and allow for multiple iterations, we can afford to only use weaker predictors for the fitting subproblem per iteration.

Implementation Outline: The above coordinate descent solution has clear interpretations and is straightforward to implement. Suppose our iterative methods has K iterations. We initialize S^1 by randomly sampling a few samples from X, and train an initial predictor $\tilde{f}_1$. Then at iterations $k = 2, \ldots K$, we jointly optimize the sampling set S^k and predictor $\tilde{f}_k$ in an alternative manner.

Subproblem 1: Architecture Sampling. At iteration $k+1$, we first sort all architectures[3] in the search space X (excluding all the samples already in S^k) according to its predicted performance $\tilde{P}^k$ at every iteration k. We then randomly sample M new architectures from the top N ranked architectures in $\tilde{P}^k$. Note this step both reduces the sample budget, and controls the exploitation-exploration trade-off (see Section 3.3.2). The newly sampled architectures together

[3]One only exception is the Section 3.3.3 open-domain experiments: we will sub-sample all architectures in the search space before sorting. More details can be found in Section 3.3.1.2



(a) (b) (c)

Figure 3.4: Visualization of the search dynamics in NAS-Bench-201 Search Space. (best viewed in color) (a) The trajectory of the predicted best architecture and global optimal through out 5 iterations; (b) Error *empirical distribution function* (EDF) of the predicted top-200 architectures throughout 5 iterations (c) Triangle marker: probability of sampling top-50 architectures throughout 5 iterations; Star marker: Kendall's Tau ranking of NAS predictor in Top 50 architectures through out 5 iterations. Reprinted with permission from [3].

with S^k become S^{k+1}.

Subproblem 2: (Weak) Predictor Fitting. We learn a predictor $\tilde{f}^{k+1}$, by minimizing the loss $\mathcal{L}$ of the predictor $\tilde{f}^{k+1}$ based on sampled architectures S^{k+1}. We then evaluate architectures using the learned predictor $\tilde{f}^{k+1}$ to get the predicted performance $\tilde{P}^{k+1}$.

As illustrated in Figure 3.3, through alternating iterations, we progressively evolve weak predictors to focus on sampling along the search path, thus simplifying the learning workload of each predictor. With these coarse-to-fine iterations, the predictor $\tilde{f}^k$ would guide the sampling process to gradually zoom into the promising architecture samples. In addition, the promising samples S^{k+1} would in turn improve the performance of the updated predictor $\tilde{f}^{k+1}$ among the well-performed architectures, hence the ranking of sampling space is also refined gradually. In other words, the solution quality to the subproblem 2 will gradually increase as a natural consequence of the guided zoom-in. For derivation, we simply choose the best architecture predicted by the final weak predictor. This idea is related to the classical ensembling [78], yet a new regime to NAS.

Proof-of-Concept Experiment.Figure 3.4 (a) shows the progressive procedure of finding the

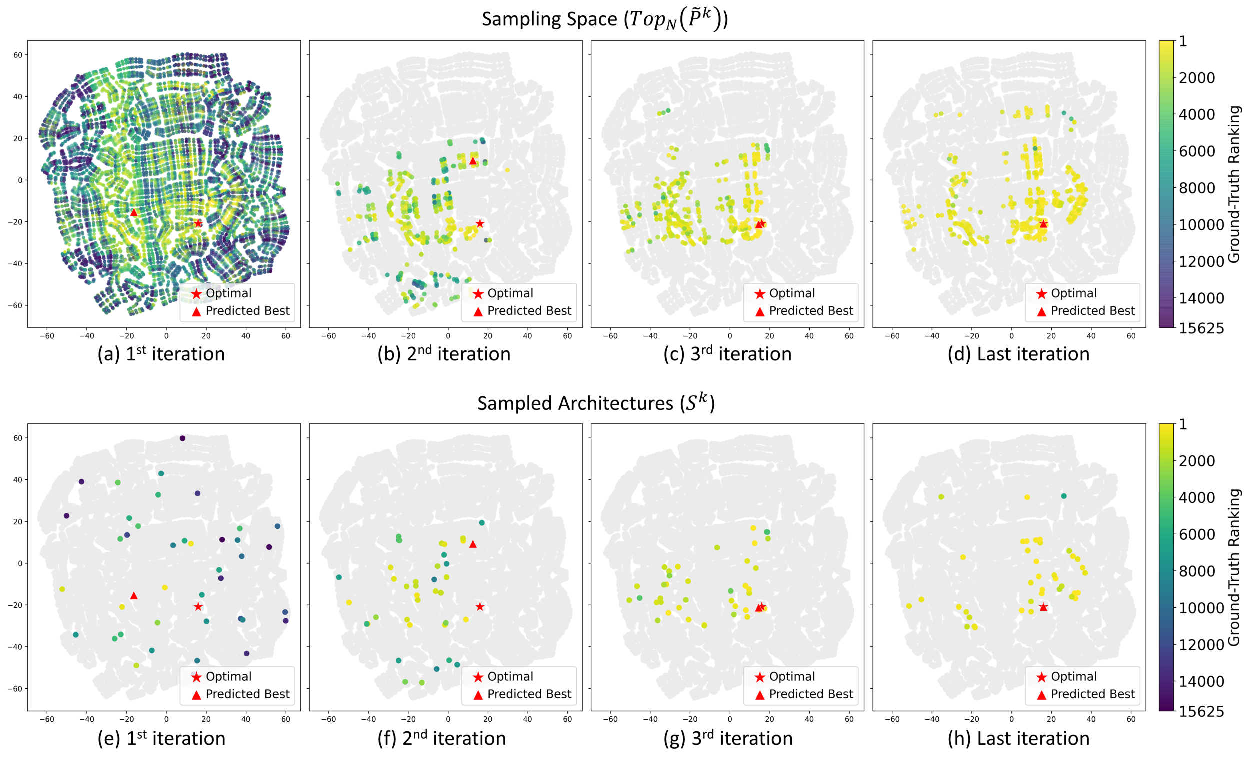


Figure 3.5: Visualization of search dynamics in NAS-Bench-201 Search Space via t-SNE. At i-th iteration, we randomly sample M = 40 new architectures from the top N = 400 ranked architectures in $\tilde{P}^k$. The top row from (a)-(d) show the sampling space $Top_N(\tilde{P}^k)$, and the bottom row from (e)-(h) show the sampled architectures S^k. The performance ranking of architectures is encoded by color, and those not-sampled architectures are colored in grey. Reprinted with permission from [3].

optimal architecture x^* and learning the predicted best architecture $\tilde{x}^*_k$ over 5 iterations. As we can see from Figure 3.4 (a), the optimal architecture and the predicted best one are moving towards each other closer and closer, which indicates that the performance of predictor over the optimal architecture(s) is growing better. In Figure 3.4 (b), we use the error *empirical distribution function* (EDF) [79] to visualize the performance distribution of architectures in the subspace. We plot the EDF of the top-200 models based on the predicted performance over 5 iterations. As is shown, the subspace of top-performed architectures is consistently evolving towards more promising architecture samples over 5 iterations. Then in Figure 3.4 (c), we validate that the probabilities of sampling better architectures within the top N predictions keep increasing. Based on this property, we can just sample a few well-performing architectures guided by the predictive model to estimate another better weak predictor. The same plot also suggests that the NAS predictor's ranking among the top-performed models is gradually refined, since more and more architectures in the top region are sampled.

In Figure 3.5, we also show the t-SNE visualization of the search dynamic in NAS-Bench-201 search space. We can observe that: (1) NAS-Bench-201 search space is highly structured; (2) the sampling space $Top_N(\tilde{P}^k)$ and sampled architectures S^k are both consistently evolving towards more promising regions, as can be noticed by the increasingly warmer color trend.

3.2.3 Relationship to Bayesian Optimization: A Simplification and Why It Works

Our method can be alternatively regarded as a **vastly simplified variant** of Bayesian Optimization (BO). It does not refer to any explicit uncertainty-based modeling such as Gaussian Process (which are often difficult to scale up); instead it adopts a *very simple step function* as our acquisition function. For a sample x in the search space X, our special “acquisition function" can be written as:

$$acq(x) = u(x - \theta) \cdot \epsilon \tag{3.5}$$

where the step function $u(x)$ is 1 if $x \geq \theta$, and 0 otherwise; ϵ is a random variable from the uniform distribution $U(0, 1)$; and θ is the threshold to split Top$N$ from the rest, according to their predicted performance $\tilde{P}^k(x)$. We then choose the samples with the M largest acquisition values:

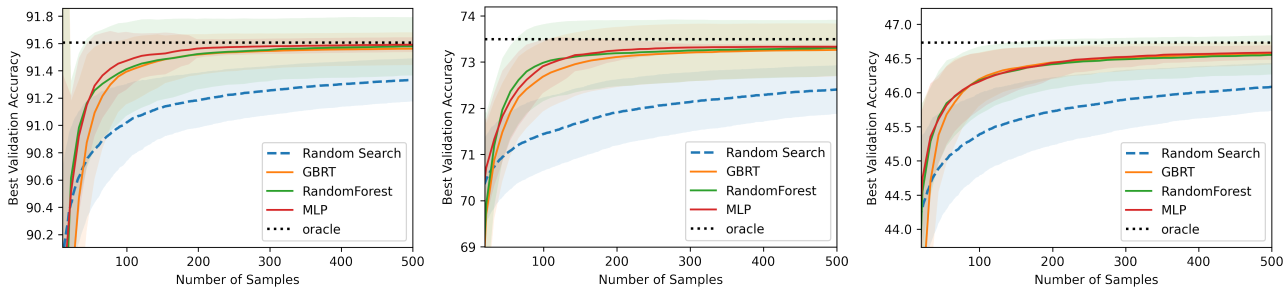
$$S_M = \underset{\text{Top}M}{\arg\max}\, acq(x) \tag{3.6}$$

Why such “oversimplified BO" can be effectively for our framework? We consider the reason to be the inherently structured NAS search space. Specifically, existing NAS spaces are created either by varying operators from a pre-defined operator set (DARTS/NAS-Bench-101/201 Search Space) or by varying kernel size, width or depth (MobileNet Search Space). Therefore, as shown in Figure 3.5, the search spaces are often highly-structured, and the best performers gather close to each other.

Here comes our underlying prior assumption: *we can progressively connect a piecewise search path from the initialization, to the finest subspace where the best architecture resides.* At the beginning, since the weak predictor only roughly fits the whole space, the sampling operation will be “noisier", inducing more exploration. When it comes to the later stage, the weak predictors fit better on the current well-performing clusters, thus performing more exploitation locally. Therefore our progressive weak predictor framework provides a natural evolution between exploration and exploitation, without explicit uncertainty modeling, thanks to the prior of special NAS space structure.

Another exploration-exploitation trade-off is implicitly built in the adaptive sampling step of our subproblem 1 solution. To recall, at each iteration, instead of choosing all Top N models by the latest predictor, we randomly sample M models from Top N models to explore new architectures in a stochastic manner. By varying the ratio $\epsilon = M/N$ and the sampling strategy (e.g., uniform, linear-decay or exponential-decay), we can balance the sampling exploitation and exploration per step, in a similar flavor to the ϵ-greedy [80] approach in reinforcement learning.

3.2.4 Our Framework is General to Predictor Models and Architecture Representations



(a) CIFAR10 (b) CIFAR100 (c) ImageNet16-120

Figure 3.6: Evaluations of robustness across different predictors on NAS-Bench-201. Solid lines and shadow regions denote the mean and std, respectively. Reprinted with permission from [3].

Our framework is designed to be generalizable to various predictors and features. In predictor-based NAS, the objective of fitting the predictor $\tilde{f}$ is often cast as a regression [60] or ranking [58] problem. The choice of predictors is diverse, and usually critical to final performance [58, 59, 55, 60, 61, 62]. To illustrate our framework is generalizable and robust to the specific choice of predictors, we compare the following predictor variants.

- *Multilayer perceptron (MLP)*: MLP is the common baseline in predictor-based NAS [58] due to its simplicity. For our weak predictor, we use a 4-layer MLP with hidden layer dimension of (1000, 1000, 1000, 1000).

- *Regression Tree*: tree-based methods are also popular [62, 81] since they are suitable for categorical architecture representations. As our weak predictor, we use the Gradient Boosting Regression Tree (GBRT) based on XGBoost [82], consisting of 1000 Trees.

- *Random Forest*: random forests differ from GBRT in that they combines decisions only at the end rather than along the hierarchy, and are often more robust to noise. For each weak predictor, we use a random forest consisting of 1000 Forests.

The features representations to encode the architectures are also instrumental. Previous methods hand-craft various features for the best performance, e.g., raw architecture encoding [59], supernet statistics [83], and graph convolutional network encoding [60, 58, 61, 72] Our framework is also agnostic to various architecture representations, and we compare the following:

- *One-hot vector*: In NAS-Bench-201 [84], its DARTS-style search space has fixed graph connectivity, hence the one-hot vector is commonly used to encode the choice of operator.

- *Adjacency matrix*: In NAS-Bench-101, we used the same encoding scheme as in [85, 59], where a 7×7 adjacency matrix represents the graph connectivity and a 7-dimensional vector represents the choice of operator on every node.

As shown in Figure 3.6, all predictor models perform similarly across different datasets. Comparing performance on NAS-Bench-101 and NAS-Bench-201, although they use different architecture encoding methods, our method still performs similarly well among different predictors. This demonstrates that our framework is robust to various predictor and feature choices.

3.3 Experiments

3.3.1 Implementation details

Setup: For all experiments, we use an Intel Xeon E5-2650v4 CPU and a single Tesla P100 GPU, and use the Multilayer perceptron (MLP) as our default NAS predictor, unless otherwise specified.

3.3.1.1 Implementation details on NAS-Bench Search Spaces

NAS-Bench-101 [85] provides a Directed Acyclic Graph (DAG) based cell structure. The connectivity of DAG can be arbitrary with a maximum number of 7 nodes and 9 edges. Each nodes on the DAG can choose from operator of 1×1 convolution, 3×3 convolution or 3×3 max-pooling. After removing duplicates, the dataset consists of 423,624 diverse architectures trained on CIFAR10[45]. **NAS-Bench-201** [84] is a more recent benchmark with a reduced DARTS-like search space. The DAG of each cell is fixed, and one can choose from 5 different operations (1×1 convolution, 3×3 convolution, 3×3 avg-pooling, skip, no connection), on each of the 6 edges, totaling 15,625 architectures. It is trained on 3 different datasets: CIFAR10, CIFAR100 and ImageNet16-120 [86]. For experiments on both NAS-Benches, we followed the same setting as [61].

For random search and regularized evolution[67] baseline, we use the public implementation from this link[4]. For random search, we selection 100 random architectures at each iteration. For regularized evolution, We set the initial population to 10, and the sample size each iteration to 3.

3.3.1.2 Implementation details on Open Domain Search Space

Open Domain Search Space: We follow the same NASNet search space used in [87] and MobileNet Search Space used in [88] to directly search for the best architectures on ImageNet[89]. Due to the huge computational cost to evaluate sampled architectures on ImageNet, we leverage a weight-sharing supernet approach. On NASNet search space, we use Single-Path One-shot [90] approach to train our SuperNet, while on MobileNet Search Space we reused the pre-trained supernet from OFA[88]. We then use the supernet accuracy as the performance proxy to train weak predictors. We clarify that despite using supernet, our method is more accurate than exist-

[4]`https://github.com/D-X-Y/AutoDL-Projects`

ing differentiable weight-sharing methods, meanwhile requiring less samples than evolution based weight-sharing methods, as manifested in Table 3.10 and 3.11. We adopt PyTorch and image models library (timm) [91] to implement our models and conduct all ImageNet experiments using 8 Tesla V100 GPUs. For derived architecture, we follow a similar training from scratch strategies used in LaNAS[74].

Open Domain Setting: We extend WeakNAS to open domain settings by (a) Construct the evaluation pool $\bar{X}$ by uniform sampling the whole search space X (b) Apply WeakNAS in the evaluation space $\bar{X}$ to find the best performer. (c) Train the best performer architecture from scratch.

For instance, when working with MobileNet search space that includes $\approx 10^{18}$ architectures, we uniformly sample 10K models as an evaluation pool, and further apply WeakNAS with a sample budget of 800 or 1000. When working with NASNet search space that includes $\approx 10^{21}$ architectures, we uniformly sample 100K models as an evaluation pool, and further apply WeakNAS with a sample budget of 800.

In the following part, we take MobileNet open domain search space as a example, however we follow a similar procedure for NASNet search space.

(a) Construct the evaluation pool $\bar{X}$ from the search space X We uniformly sample an evaluation pool to handle the extremely large MobileNet search space ($|X| \approx 10^{18}$), since its not doable to predict the performance of all architectures in X. We use uniform sampling due to a recent study [79] reveal that human-designed NAS search spaces usually contain a fair proportion of good models compared to random design spaces, for example, in Figure 9 of [79], it shows that in NASNet/Amoeba/PNAS/ENAS/DARTS search spaces, Top 5% of models only have a <1% performance gap to the global optima. In practice, the uniform sampling strategy has been widely verified as effective in other works of predictor-based NAS such as [60, 92, 93], For example, [60] [92][93] set to be 112K, 15K, 20K in a search space of 10^{18} networks. In our case, we set $|\bar{X}|$ = 10K.

(b) Apply WeakNAS in the evaluation space $\bar{X}$ We then further apply WeakNAS in the evaluation pool $\bar{X}$. This is because even with the evaluation pool $|\bar{X}|$ = 10K, it still takes days to evaluate all those models on ImageNet (in a weight-sharing SuperNet setting). Since the evaluation pool $\bar{X}$ was uniformly sampled from NAS search space X, it preserves the highly-structured nature of X. As a result, we can leverage WeakNAS to navigate through the highly-structured search space. WeakNAS build a iterative process, where it searches for some top-performing cluster at the initial search iteration and then "zoom-in" the cluster to find the top performers within the same cluster (as shown in Figure 3.5). At $k-th$ iteration, WeakNAS balance the exploration and exploitation trade-off by sampling 100 models from the Top 1000 predictions of the predictor $\tilde{f}^k$, it use the promising samples to further improve performance of the predictor in the next iteration $\tilde{f}^{k+1}$. We leverage WeakNAS to further decrease the number of queries to find the optimal in $\bar{X}$ by 10 times, the search cost has dropped from 25 GPU hours (evaluate all 10K samples in random evaluation pool) to 2.5 GPU hours (use WeakNAS in 10K random evaluation pool), while still achieving a solid performance of 81.3% on ImageNet (MobileNet Search Space).

(c) Train the best performer architecture from scratch. We follow a similar setting in LaNAS[74], where we use Random Erase and RandAug, a drop out rate of 0.3 and a drop path rate of 0.0, we also use exponential moving average (EMA) with a decay rate of 0.9999. During training and evaluation, we set the image size to be 236x236 (In NASNet search space, we set the image size to be 224x224). We train for 300 epochs with warm-up of 3 epochs, we use a batch size of 1024 and RMSprop as the optimizer. We use a cosine decay learning rate scheduler with a starting learning rate of 1e-02 and a terminal learning rate of 1e-05

3.3.2 Ablation Studies

We conduct a series of ablation studies on the effectiveness of proposed method on NAS-Bench-101.

3.3.2.1 Ablation on our iterative scheme

To validate the effectiveness of our iterative scheme, In Table 3.1, we initialize the initial Weak Predictor $\tilde{f}_1$ with 100 random samples, and set $M = 10$, after progressively adding more weak predictors (from 1 to 191), we find the performance keeps growing. This demonstrates the key property of our method that probability of sampling better architectures keeps increasing as more iteration goes. It's worth noting that the quality of random initial samples M_0 may also impact on the performance of WeakNAS, but if $|M_0|$ is sufficiently large, the chance of hitting good samples (or its neighborhood) is high, and empirically we found $|M_0|$=100 to already ensure highly stable performance at NAS-Bench-101: a more detailed ablation can be found in the Section 3.3.2.3.

Sampling	#Predictor	#Queries	Test Acc.(%)	SD(%)	Test Regret(%)	Avg. Rank
Uniform	1 Strong Predictor	2000	93.92	0.08	0.40	135.0
Iterative	1 Weak Predictor	100	93.42	0.37	0.90	6652.1
	11 Weak Predictors	200	94.18	0.14	0.14	5.6
	91 Weak Predictors	1000	94.25	0.04	0.07	1.7
	191 Weak Predictors	2000	94.26	0.04	0.06	1.6
Optimal	-	-	94.32	-	0.00	1

Table 3.1: Ablation on the effectiveness of our iterative scheme on NAS-Bench-101. Reprinted with permission from [3].

3.3.2.2 Ablation on exploitation-exploration trade-off

We then study the exploitation-exploration trade-off in Table 3.2 in NAS-Bench-101 (a similar ablation in Mobilenet Search space on ImageNet is also included in Table 3.3) by investigating two settings: (a) We gradually increase N to allow for more exploration, similar to controlling ϵ in the epsilon-greedy [80] approach in the RL context; (b) We vary the sampling strategy from Uniform, Linear-decay to Exponential-decay (top models get higher probabilities by following either linear-decay or exponential-decay distribution).

NAS-Bench Search Spaces We empirically observed that: (a) The performance drops more (Test Regret 0.22% vs 0.08%) when more exploration (TopN=1000 vs TopN=10) is used. This indicates that extensive exploration is not optimal for NAS-Bench-101; (b) Uniform sampling method yields better performance than sampling method that biased towards top performing model (e.g. linear-decay, exponential-decay). This indicates good architectures are evenly distributed within the Top

100 predictions of Weak NAS, therefore a simple uniform sampling strategy for exploration is more optimal in NAS-Bench-101. To conclude, our Weak NAS Predictor strikes a good balance between exploration and exploration.

Apart from the above exploitation-exploration trade-off of WeakNAS, we also explore the possibility of integrating other meta-sampling methods. We found that the local search algorithm could achieve comparable performance, while using Semi-NAS [73] as a meta sampling method could further boost the performance of WeakNAS: more details are in Section 3.3.2.5.

Sampling (M from TopN)	M	TopN	#Queries	Test Acc.(%)	SD(%)	Test Regret(%)	Avg. Rank
Exponential-decay	10	100	1000	93.96	0.10	0.36	85.0
Linear-decay	10	100	1000	94.06	0.08	0.26	26.1
Uniform	**10**	**100**	**1000**	**94.25**	**0.04**	**0.07**	**1.7**
Uniform	10	1000	1000	94.10	0.19	0.22	14.1
Uniform	**10**	**100**	**1000**	**94.25**	**0.04**	**0.07**	**1.7**
Uniform	10	10	1000	94.24	0.04	0.08	1.9

Table 3.2: Ablation on exploitation-exploration trade-off on NAS-Bench-101. Reprinted with permission from [3].

Open Domain Search Space For the ablation on open-domain search space, we follow the same setting in the Section 3.3.1.2 and use Mobilenet Search space on ImageNet, however due to the prohibitive cost of training model from scratch in Section 3.3.1.2 (c), we directly use accuracy derived from supernet.

WeakNAS uniformly samples M samples from TopN predictions at each iteration, thus we can adjust N/M ratio to balance the exploitation-exploration trade-off. In Table 3.3, we set the total number of queries at 100, fix M at 10 and while adjusting N from 10 (more exploitation) to 1000 (more exploration), and use optimal in the 10K evaluation pool to measure the ranking and test regret. We found WeakNAS is quite robust within the range where N/M = 2.5 - 10, achieving the best performance at the sweet spot of N/M = 5. However, its performance drops significantly (by rank), while doing either too much exploitation (N/M <2.5) or too much exploration (N/M >25).

Sampling methods	M	TopN	#Queries	SuperNet Test Acc.(%)	SD(%)	Test Regret(%)	Avg. Rank
Uniform	-	-	100	79.0609	0.0690	0.1671	94.58
Iterative	10	10	100	79.1552	0.0553	0.0728	20.69
	10	25	100	79.1936	0.0289	0.0344	4.68
	10	**50**	**100**	**79.2005**	**0.0300**	**0.0275**	**4.05**
	10	100	100	79.1954	0.0300	0.0326	4.63
	10	250	100	79.1755	0.0416	0.0525	10.58
	10	500	100	79.1710	0.0388	0.0570	10.80
	10	1000	100	79.1480	0.0459	0.0800	19.70
	10	2500	100	79.1274	0.0597	0.1006	33.64

Table 3.3: Ablation on exploitation-exploration trade-off over 100 runs on MobleNet Search Space over ImageNet. Reprinted with permission from [3].

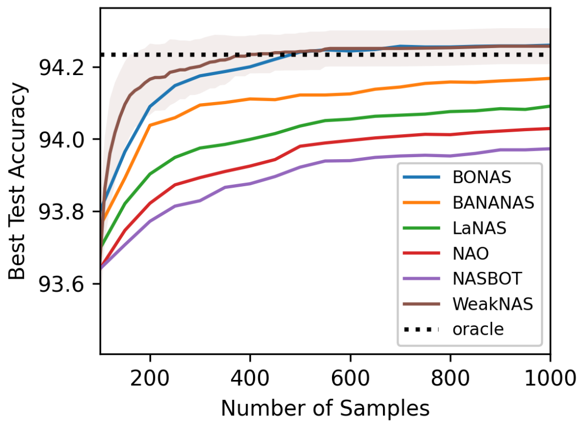


Figure 3.7: Comparison with SoTA methods on NAS-Bench-101. Solid lines and shadow regions denote the mean and std, respectively. Reprinted with permission from [3].

3.3.2.3 Ablation on number of initial samples

We conduct a controlled experiment in varying the number of initial samples $|M_0|$ in Table 3.4. On NAS-Bench-101, we vary $|M_0|$ from 10 to 200, and found a "warm start" with good initial samples is crucial for good performance. Too small number of $|M_0|$ might makes the predictor lose track of the good performing regions. As shown in Table 3.4. We empirically found $|M_0|$=100 can ensure highly stable performance on NAS-Bench-101.

$\|M_0\|$	#Queries	Test Acc.(%)	SD(%)	Test Regret(%)	Avg. Rank
10	1000	94.14	0.10	0.18	9.1
100	**1000**	**94.25**	**0.04**	**0.07**	**1.7**
200	1000	94.19	0.08	0.13	5.2
10	200	94.04	0.13	0.28	33.5
100	**200**	**94.18**	**0.14**	**0.14**	**5.6**
200	200	93.78	1.45	0.54	558.0
Optimal	-	94.32	-	0.00	1.0

Table 3.4: Ablation on number of initial samples M_0 on NAS-Bench-101. Reprinted with permission from [3].

3.3.2.4 Ablation on the architecture encoding

We also compare WeakNAS with NAS using custom architecture representation either in a unsupervised way such as arch2vec [94], or a supervised way such as CATE [95]. We compare the effect of using different architecture encodings in in Table 3.5. We found when combined with CATE embedding [95], the performance of WeakNAS can be further improved, compared to WeakNAS baseline with adjacency matrix encoding used in [85]. This also leads to stronger performance than cate-DNGO-LS baseline in CATE [95], which demonstrates that CATE embedding [95] is an orthogonal contribution to WeakNAS, and they are mutually compatible.

Methods	#Queries	Test Acc.(%)	SD(%)	Test Regret(%)	Avg. Rank
CATE (cate-DNGO-LS)[95]	150	94.10	-	0.22	12.3
WeakNAS + Adjacency matrix[85]	150	94.10	0.19	0.22	12.3
WeakNAS + CATE[95]	150	94.19	0.12	0.13	5.24

Table 3.5: Ablation on architecture encoding on NAS-Bench-101. Reprinted with permission from [3].

3.3.2.5 Abaltion of meta-sampling methods in WeakNAS

We also show that local search algorithm (hill climbing) or Semi-NAS [73] can be used as a meta sampling method in WeakNAS, which could further boost the performance of WeakNAS, here are the implementation details.

Local Search Given a network architecture embedding s in NAS-Bench-101 Search Space, we first define a nearest neighbour function $N(s)$ as architecture that differ from s by a edge or a operation. At each iteration, we random sample a initial sample s_i from TopN predictions $\text{Top}_N(\tilde{P}^k)$ and sample all of its nearest neighbour architecture in $N(v_0)$. We then let the new $s_{i+1} = \arg\max_{s \in N(s_i)} f(s)$. We repeat the process iteratively until we reach a local maximum such that $\forall v \in N(s), f(s) \geqslant f(v)$ or the sampling budget M of the iteration is reached.

Semi-NAS At the sampling stage of each iteration in WeakNAS, we further use Semi-NAS as a meta-sampling methods. Given a meta search space of 1000 architectures and a sample budget of 100 queries each iteration. We following the setting in Semi-NAS, using the same 4-layer MLP NAS predictor in WeakNAS and uses pseudo labels as noisy labels to augment the training set, therefore we are able to leverage “unlabeled samples" (e.g., architectures with accuracy generated by the predictors) to update the predictor. We set the initial sample to be 10, and sample 10 more samples each iteration. Note that at the start of k-th WeakNAS iteration, we inherent the weight of Semi-NAS predictor from the previous (k-1)-th WeakNAS iteration.

Sampling (M from TopN)	M	N	#Queries	Test Acc.(%)	SD(%)	Test Regret(%)	Avg. Rank
WeakNAS	100	1000	1000	94.25	0.04	0.07	1.7
Local Search	-	-	1000	94.24	0.03	0.08	1.9
Semi-NAS	-	-	1000	94.26	0.02	0.06	1.6

Table 3.6: Ablation on meta-sampling methods on NAS-Bench-101. Reprinted with permission from [3].

3.3.3 Comparison to State-of-the-art (SOTA) Methods

NAS-Bench-101: On NAS-Bench-101 benchmark, we compare our method with several popular methods [67, 96, 74, 55, 60, 73, 72, 94, 97, 95, 98].

Table 3.8 shows that our method significantly outperforms baselines in terms of sample efficiency. Specifically, our method costs 964×, 447×, 378×, 245×, 58×, and 7.5× less samples to reach the optimal architecture, compared to Random Search, Regularized Evolution [67], MCTS [96], Semi-NAS[73], LaNAS[74], BONAS[72], respectively. We then plot the best accuracy against number of samples in Table 3.7 and Figure 3.7 to show the sample efficiency on the NAS-Bench-101, from which we can see that our method consistently costs fewer sample to reach higher accuracy.

We conduct a controlled experiment on NAS-Bench-201 by varying number of samples. As shown in Figure 3.8, our average performance over different number of samples is clearly better than Regularized Evolution [67] in all three subsets, with better stability indicated by confidence intervals.

NAS-Bench-201: We further evaluate on NAS-Bench-201, and compare with random search, Regularized Evolution [67], Semi-NAS[73], LaNAS[74], BONAS[72]. As shown in Table 3.8, we conduct searches on all three subsets (CIFAR10, CIFAR100, ImageNet16-120) and report the average number of samples needed to reach global optimal on the testing set over 100 runs. It shows that our method has the smallest sample cost among all settings.

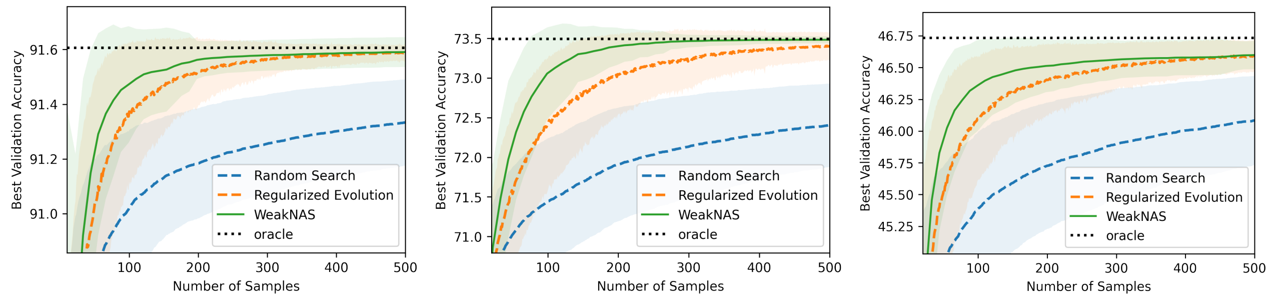
Open Domain Search: we further apply our method to open domain search without ground-truth,

Method	#Queries	Test Acc.(%)	SD(%)	Test Regret(%)	Avg. Rank
Random Search	2000	93.64	0.25	0.68	1750.0
NAO [55]	2000	93.90	0.03	0.42	168.1
Reg Evolution [67]	2000	93.96	0.05	0.36	85.0
Semi-NAS [73]	2000	94.02	0.05	0.30	42.1
Neural Predictor [60]	2000	94.04	0.05	0.28	33.5
WeakNAS	**2000**	**94.26**	**0.04**	**0.06**	**1.6**
Semi-Assessor [97]	1000	94.01	-	0.31	47.1
LaNAS [74]	1000	94.10	-	0.22	14.1
BONAS [72]	1000	94.22	-	0.10	3.0
WeakNAS	**1000**	**94.25**	**0.04**	**0.07**	**1.7**
Arch2vec [94]	400	94.10	-	0.22	14.1
WeakNAS	**400**	**94.24**	**0.04**	**0.08**	**1.9**
LaNAS [74]	200	93.90	-	0.42	168.1
BONAS [72]	200	94.09	-	0.23	18.0
WeakNAS	**200**	**94.18**	**0.14**	**0.14**	**5.6**
NASBOWLr [99]	150	94.09	-	0.23	18.0
CATE (cate-DNGO-LS) [95]	150	94.10	-	0.22	12.3
WeakNAS	**150**	**94.10**	**0.19**	**0.22**	**12.3**
Optimal	-	94.32	-	0.00	1.0

Table 3.7: Comparing searching efficiency by limiting the total query amounts on NAS-Bench-101. Reprinted with permission from [3].

Method	NAS-Bench-101	NAS-Bench-201		
Dataset	CIFAR10	CIFAR10	CIFAR100	ImageNet16-120
Random Search	188139.8	7782.1	7621.2	7726.1
Reg Evolution [67]	87402.7	563.2	438.2	715.1
MCTS [96]	73977.2	†528.3	†405.4	†578.2
Semi-NAS [73]	†47932.3	-	-	-
LaNAS [74]	11390.7	†247.1	†187.5	†292.4
BONAS [72]	1465.4	-	-	-
WeakNAS	**195.2**	**182.1**	**78.4**	**268.4**

Table 3.8: Comparison on the number of samples required to find the global optimal on NAS-Bench-101 and NAS-Bench-201. † denote reproduced results using adapted code. Reprinted with permission from [3].



(a) CIFAR10 (b) CIFAR100 (c) ImageNet16-120

Figure 3.8: Comparison to SOTA on NAS-Bench-201 by varying number of samples. Solid lines and shadow regions denote the mean and std, respectively. Reprinted with permission from [3].

and compare with several popular methods [87, 67, 100, 55, 101, 93, 74]. As shown in Tables 3.10 and 3.11, using the fewest samples (and only a fraction of GPU hours) among all, our method can achieve state-of-the-art ImageNet top-1 accuracies with comparable parameters and FLOPs. Our searched architecture is also competitive to expert-design networks. On the NASNet Search Space, compared with the SoTA predictor-based NAS method LaNAS (Oneshot) [74], our method reduces 0.6% top-1 error while using less GPU hours. On the MobileNet Search Space, we improve the previous SoTA LaNAS [74] to 81.3% top-1 accuracy on ImageNet while costing less FLOPs.

BO-based NAS methods [72, 99]: BO-based methods in general treat NAS as a black-box optimization problem, for example, BONAS [72] customizes the classical BO framework in NAS with GCN embedding extractor and Bayesian Sigmoid Regression to acquire and select candidate architectures. The latest BO-based NAS approach, NASBOWL [99], combines the Weisfeiler-Lehman graph kernel in BO to capture the topological structures of the candidate architectures.

Compare with those BO-based method, our WeakNAS is an “oversimplified" version of BO as explained in Section 3.2.3. Interestingly, results in Table 3.7 suggests that WeakNAS is able to outperform BONAS [72], and is comparable to NASBOWLr [99] on NAS-Bench-101, showcasing that the simplification does not compromise NAS performance. We hypothesize that the following factors might be relevant: (1) the posterior modeling and uncertainty estimation in BO might be noisy; (2) the inherently structured NAS search space (shown in Figure 3.5) could enable a “shortcut" simplification to explore and exploit. In addition, the conventional uncertainty modeling in BO, such as the Gaussian Process used by [99], is not as scalable when the number of queries is large. In comparison, the complexity of WeakNAS scales almost linearly, as can be verified in Table 3.13. In our experiments, we observe WeakNAS to perform empirically more competitively than current BO-based NAS methods at larger query numbers, besides being way more efficient.

To further convince that WeakNAS is indeed an effective simplification compared to the explicit posterior modeling in BO, we report an apple-to-apple comparison, by use the same weak predictor from WeakNAS, plus obtaining its uncertainty estimation by calculating its variance using a deep ensemble of five model [102]; we then use the classic Expected Improvement (EI) [103] acquisition function. Table 3.9 confirms that such BO variant of WeakNAS is inferior our proposed formulation.

Method	#Queries	Test Acc.(%)	SD(%)	Test Regret(%)	Avg. Rank
WeakNAS	**1000**	**94.25**	**0.04**	**0.07**	**1.7**
WeakNAS (BO Variant)	1000	94.12	0.15	0.20	8.7
Optimal	-	94.32	-	0.00	1.0

Table 3.9: Comparing to the BO variant of WeakNAS on NAS-Bench-101. Reprinted with permission from [3].

LaNAS [74]: LaNAS and our framework both follow the divide-and-conquer idea, yet with two methodological differences: *(a) How to split the search space*: LaNAS learns a *classifier* to do binary “hard” partition on the search space (no ranking information utilized) and split it into two equally-sized subspaces. Ours uses a *regressor* to regress the performance of sampled architectures, and utilizes the ranking information to sample a percentage of the top samples (“soft” partition), with the sample size N being controllable. *(b) How to do exploration*: LaNAS uses Upper Confidence Bound (UCB) to explore the search space by not always choosing the best subspace (left-most node) for sampling, while ours always chooses the best subspace and explore new architectures by adaptive sampling within it, via adjusting the ratio $\epsilon = M/N$ to randomly sample M

models from Top N. Tables 3.7 and 3.8 shows the superior sample efficiency of WeakNAS over LaNAS on NAS-Bench-101/201.

Semi-NAS [73] and Semi-Assessor[97]: Both our method and Semi-NAS/Semi-Assessor use an iterative algorithm containing prediction and sampling. The main difference lies in the use of pseudo labels: Semi-NAS and Semi-Assessor use pseudo labels as noisy labels to augment the training set, therefore being able to leverage “unlabeled samples" (e.g., architectures without true accuracies, but with only accuracies generated by the predictors) to update their predictors. Our method explores an **orthogonal** innovative direction, where the “pseudo labels" generated by the current predictor guide our sampling procedure, but are **never used** for training the next predictor.

That said, we point out that our method can be **complementary** to those semi-supervised methods [73, 97], thus they can further be **integrated** as one, For example, Semi-NAS can be used as a meta sampling method, where at each iteration we further train a Semi-NAS predictor with pseudo labeling strategy to augment the training set of our weak predictors. We show in Table 3.6 that the combination of our method with Semi-NAS can further boost the performance of WeakNAS.

BRP-NAS [61]: BRP-NAS uses a stronger GCN-based binary relation predictor which utilize extra topological prior, and leveraged a different scheme to control exploitation and exploration trade-off compare to our WeakNAS. Further, BRP-NAS also use a somehow unique setting, i.e. evaluating Top-40 predictions by the NAS predictor instead of the more common setting of Top-1 [55, 72, 74, 73]. Therefore, we include our comparison to BRP-NAS and more details in Section 3.3.4.

3.3.4 Comparison to BRP-NAS

Evaluation strategy: BRP-NAS[61] uses a unique setting that differs from other predictor-based NAS, i.e., evaluating Top 40 predictions by the NAS predictor instead of Top 1 prediction, and the later was commonly followed by others[55, 72, 74, 73] and WeakNAS.

Sampling strategy: WeakNAS uses a different sampling strategy than that of BRP-NAS, given a sample budget of M, BRP-NAS picks both samples from Top-K and $(M - K)$ random models from the entire search space, while our WeakNAS only picks M random models in Top-N, thus

Model	Queries(#)	Top-1 Err.(%)	Top-5 Err.(%)	Params(M)	FLOPs(M)	GPU Days
MobileNetV2	-	25.3	-	6.9	585	-
ShuffletNetV2	-	25.1	-	5.0	591	-
SNAS[104]	-	27.3	9.2	4.3	522	1.5
DARTS[54]	-	26.9	9.0	4.9	595	4.0
P-DARTS[105]	-	24.4	7.4	4.9	557	0.3
PC-DARTS[106]	-	24.2	7.3	5.3	597	3.8
DS-NAS[106]	-	24.2	7.3	5.3	597	10.4
NASNet-A [87]	20000	26.0	8.4	5.3	564	2000
AmoebaNet-A [67]	10000	25.5	8.0	5.1	555	3150
PNAS [100]	1160	25.8	8.1	5.1	588	200
NAO [55]	1000	24.5	7.8	6.5	590	200
LaNAS [74] (Oneshot)	800	24.1	-	5.4	567	3
LaNAS [74]	800	23.5	-	5.1	570	150
WeakNAS	**800**	**23.5**	**6.8**	**5.5**	**591**	**2.5**

Table 3.10: Comparison to SOTA results on ImageNet using NASNet search space. Reprinted with permission from [3].

Model	Queries(#)	Top-1 Acc.(%)	Top-5 Acc.(%)	FLOPs(M)	GPU Days*
Proxyless NAS[107]	-	75.1	92.9	-	-
Semi-NAS[73]	300	76.5	93.2	599	-
BigNAS[101]	-	76.5	-	586	-
FBNetv3[93]	20000	80.5	95.1	557	-
OFA[88]	16000	80.0	-	595	1.6
LaNAS[74]	800	80.8	-	598	0.3
WeakNAS	**1000**	**81.3**	**95.1**	**560**	**0.16**
	800	**81.2**	**95.2**	**593**	**0.13**

Table 3.11: Comparison to SOTA results on ImageNet using MobileNet search space. *Does not include supernet training cost. Reprinted with permission from [3].

is a more “greedy” strategy. BRP-NAS controls the exploitation and exploration trade-off by adjusting $\alpha = (M - K)/M$, however they did not have any ablation discussing the exploitation and exploration trade-off and only empirically choose $\alpha = 0.5$ as the default ratio. Our WeakNAS instead controls the exploitation and exploration trade-off by adjusting N/M ratio, and we did a comprehensive analysis on the exploitation and exploration trade-off on both NAS-Bench and MobileNet Search Space on ImageNet in Section 3.3.2.

NAS predictor: BRP-NAS uses a stronger GCN-based binary relation predictors which utilizes extra topological prior, on the other hand, our framework generalizes to all choices of predictors, including MLP, Regression Tree and Random Forest, thus is not picky on the choice of predictors.

To fairly compare with BRP-NAS, we follow the exact same setting for our WeakNAS predictor, e.g., incorporating the same graph convolutional network (GCN) based predictor and using Top-40 evaluation. As shown in Table 3.12, at 100 training samples, WeakNAS can achieve comparable performance to BRP-NAS [61].

Method	#Train	#Queries	Test Acc.(%)	SD(%)	Test Regret(%)	Avg. Rank
BRP-NAS [61]	100	140	94.22	-	0.10	3.0
WeakNAS	**100**	**140**	**94.23**	**0.09**	**0.09**	**2.3**
Optimal	-	-	94.32	-	0.00	1.0

Table 3.12: Comparison to BRP-NAS on NAS-Bench-101. Reprinted with permission from [3].

3.3.5 Runtime analysis of WeakNAS

We show the runtime analysis of WeakNAS and its BO variant in Table 3.13. We can see the BO variant is much slower in training proxy models due the ensembling of multiple models. Moreover, it’s also several magnitude slower when deriving new samples, due to the calculation of its Expected Improvement (EI) acquisition function [103] being extremely costly.

Method	Predictors	Config	Train proxy model (s/arch)	Derive new samples (s/arch)
WeakNAS	MLP	4 layers @1000 hidden	8.59×10^{-5}	3.53×10^{-5}
	Gradient Boosting Tree	1000 Trees	5.70×10^{-4}	5.54×10^{-7}
	Random Forrest	1000 Forests	3.20×10^{-3}	1.77×10^{-4}
WeakNAS (BO Variant)	5 x MLPs	EI acquisition	2.84×10^{-4}	1.32×10^{-1}

Table 3.13: Runtime Comparsion of WeakNAS. Reprinted with permission from [3].

3.3.6 Founded Best Architecture on Open Domain Search

We show the best architecture founded by WeakNAS on the MobileNet search space with 800/1000 queries in Table 3.14.

Id	**Block**	**Kernel**	**#Out Channel**	**Expand Ratio**
WeakNAS @ 593 MFLOPs, #Queries=800				
0	Conv	3	24	-
1	IRB	3	24	1
2	IRB	3	32	4
3	IRB	5	32	6
4	IRB	7	48	4
5	IRB	5	48	3
6	IRB	7	48	4
7	IRB	3	48	6
8	IRB	3	96	4
9	IRB	7	96	6
10	IRB	5	96	6
11	IRB	7	96	3
12	IRB	3	136	6
13	IRB	3	136	6
14	IRB	5	136	6
15	IRB	5	136	3
16	IRB	7	192	6
17	IRB	5	192	6
18	IRB	3	192	4
19	IRB	5	192	3
20	Conv	1	192	-
21	Conv	1	1152	-
22	FC	-	1536	-

Id	**Block**	**Kernel**	**#Out Channel**	**Expand Ratio**
WeakNAS @ 560 MFLOPs, #Queries=1000				
0	Conv	3	24	-
1	IRB	3	24	1
2	IRB	5	32	3
3	IRB	3	32	3
4	IRB	3	32	4
5	IRB	3	32	3
6	IRB	5	48	4
7	IRB	5	48	6
8	IRB	5	48	4
9	IRB	7	96	4
10	IRB	5	96	6
11	IRB	7	96	6
12	IRB	3	136	6
13	IRB	5	136	6
14	IRB	5	136	6
15	IRB	7	192	6
16	IRB	5	192	6
17	IRB	3	192	6
18	IRB	5	192	3
19	Conv	1	192	-
20	Conv	1	1152	-
21	FC	-	1536	-

Table 3.14: Neural architecture found by WeakNAS on ImageNet using MobileNet search space. Reprinted with permission from [3].

3.4 Conclusions and Discussions

We present a novel predictor-based NAS framework named WeakNAS that progressively shrinks the sampling space, by learning a series of weak predictors that can connect towards the best architectures. By co-evolving the sampling stage and learning stage, our weak predictors can progressively evolve to sample towards the subspace of best architectures, thus greatly simplifying the learning task of each predictor. Extensive experiments on popular NAS benchmarks prove that the proposed method is both sample-efficient and robust to various combinations of predictors and architecture encoding means. However, WeakNAS is still limited by the human-designed encoding of neural architectures, and our future work plans to investigate how to jointly learn the predictor and encoding in our framework.

For broader impact, the excellent sample-efficiency of WeakNAS reduces the resource and energy consumption needed to search for efficient models, while still maintaining SoTA performance. That can effectively serve the goal of GreenAI, from model search to model deployment. It might meanwhile be subject to the potential abuse of searching for models serving malicious purposes.

4. NEURAL ARCHITECTURE UNIFICATION[1]

4.1 Introduction

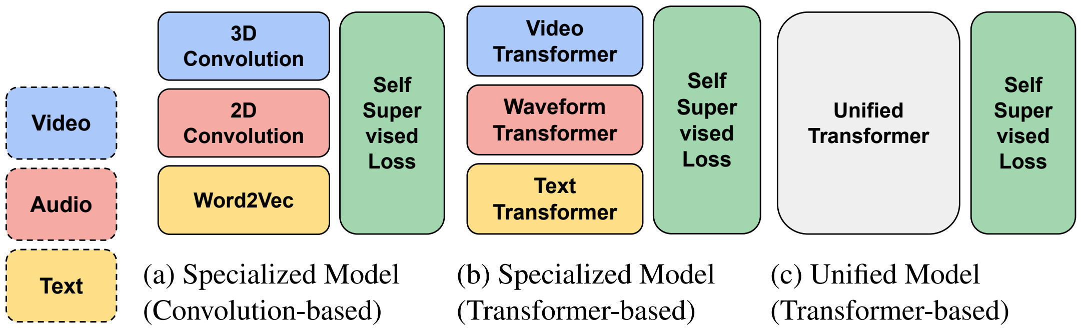


Figure 4.1: Evolution of Neural Architectures in Self-supervised Multimodal Learning

Since the advent of deep learning, specialized neural architectures are usually designed to tackle different modalities and tasks, for example, 3D convolution[108, 109, 110] over raw video/optical flow is preferred in video classification, 2D convolution[111] over spectrogram is preferred in audio classification, while in NLP tasks word2vec[112] model is the popular choice (shown in Figure 4.1a). Meanwhile, the transformer architectures was first introduced in NLP[113], thanks to its capability to capture "global" statistics using self-attention, it has proven to be a strong baseline beyond its original scope in tasks across a wide range of modality, including image [114, 115], video[116, 117], audio[117] and natural language processing[13][15] (shown in Figure 4.1b), with its performance surpassing many previous specialized neural architectures. Recently, VATT[6] has move a step further, validate the idea of using a modality-agnostic single-backbone transformer (shown in Figure 4.1c), to work on video, audio and text modalities, we named this line of approaches Neural Architecture Unification (NAU).

[1]Partial text and images of this chapter is reprinted with permission from “Scaling Multimodal Pre-Training via Cross-Modality Gradient Harmonization.” by Junru Wu et al. In Advances in Neural Information Processing Systems 35 (2022), Copyright 2023 by Junru Wu.

On the other hand, the recent success on multimodal pre-training [16, 118] demonstrates the versatility to synergize the rich multimodal information and to benefit a variety of downstream tasks. Many methods of this category [119, 120, 121, 16], especially the latest contrastive pre-training methods [118, 122, 123, 6], consider the most organic supervision as the *cross-modality alignment* (**CMA**), e.g., in the same video sequence, the video frames, audio stream, and text scripts are temporally aligned and hence should naturally have correspondence. Such assumed cross-modal correspondence could be exploited as “mutual supervision" for each modality to learn consistent semantic features with others. For example, the latest Video-Audio-Text Transformer (VATT) [6] hinges on this multimodal CMA *a priori* to train a transformer-based architecture (even a single backbone by sharing weights among three modalities), using two contrastive losses to align video-text and video-audio pairs, respectively. Their results set many new records in downstream tasks with neither supervised pre-training nor predefined inductive biases, showing strong “universal" generalizability despite the modality and domain gaps. In this paper, we focus on studying gradients in a modality-agnostic single-backbone setting of VATT [6].

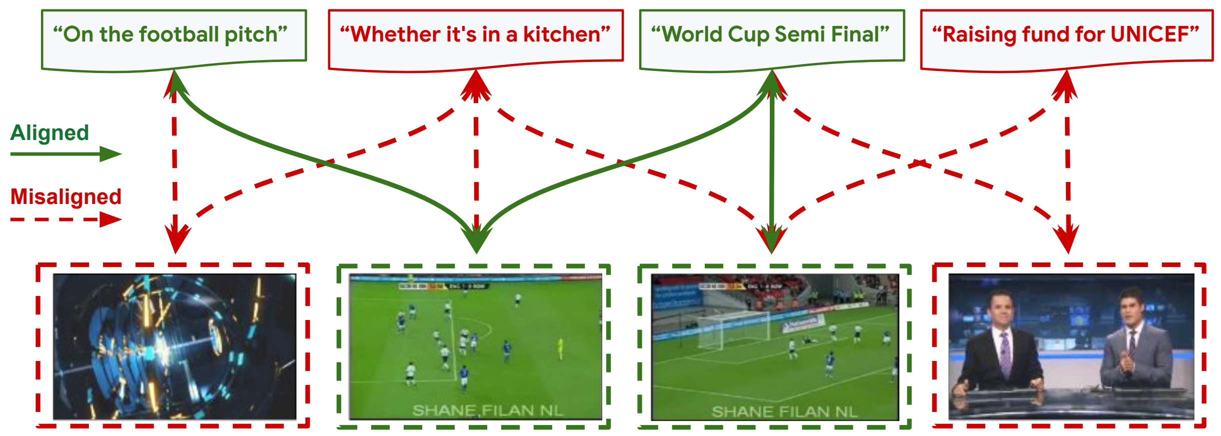


Figure 4.2: Examples of cross-modality alignment and misalignment existing in the complicated and non-narrative Youtube8M dataset [4]. Here we show a video sequence whose theme is a football match, while in some of its clips: (a) the text mentions content unrelated to the football, e.g., "kitchen","raising fund"; and (b) the visual content could also be unrelated to the football, e.g., TV cutscenes, the scene of the broadcast studio, etc. Reprinted with permission from [5].

Unfortunately, the CMA assumption rests on a very shaky foundation, especially for uncurated videos collected in the wild. The CMA can be weak even for the instructional videos such as HowTo100M [124] that are commonly adopted as the current multimodal pre-training data source. For instance, a speaker can refer to something before or after actually demonstrating it visually; she or he could also skip verbally explaining something that is visually happening, because that may be trivial or be already clear enough in the visual context [125]. Such discrepancy would be amplified if considering many other outliers in video (e.g. background objects) or language (e.g. off-topic small talks). The semantic misalignment will be even more severe, and in unpredictable forms, when we go beyond instructional videos and explore training with non-narrative, free-from internet videos. Figure 4.2 shows a few examples of such cross-modal misalignment in the non-narrative Youtube8M dataset [4], which would put CMA-guided pre-training in jeopardy. This important problem is yet under-studied in the self-supervised multimodal pre-training literature. That said, there has been a few recent efforts such as Multiple-Instance Learning (MIL) NCE [125] and noisy pair pruning as in [126].

We conjecture that the vanilla CMA might provide only weak, noisy and even misleading supervision for multimodal pre-training; hence it might constitute a hidden bottleneck when scaled up with more realistic data (beyond instructional videos) whose modalities are poorly aligned. We verify our conjecture by inspecting two large-scale multimodal datasets, HowTo100M [127] and Youtube8M [4]. Specifically, when we adopt pairwise CMA losses (e.g. video-audio, video-text) within the cross-modal sample triplets (video, audio, text), a majority of the gradients strongly conflict with each other in terms of their directions. We further identify (in Figure 4.4) that such strong conflict in gradients are correlated to noisiness of samples, indicating the noisy and misaligned supervision could be an important cause of gradient conflicts. Moreover, the cross-modal conflict leads to not only unstable training convergence, but also modality-specific overfitting; e.g. semantically strong representations for one modality while collapsed representations for another [128], yielding weak overall performance in cross-modality tasks such as Text-Video Retrieval as shown in Table 4.1.

Build upon the above conjectures and observations, we propose to harmonize those gradients so that they become mutually compatible in learning the unified representations. Specifically, we introduce two techniques: (i) cross-modal gradient realignment, where we modify different pairwise CMA loss gradients to align their directions for the same sample triplet, using gradient surgery [129] developed for multi-task learning; and (ii) gradient-based curriculum learning, where we leverage the gradient conflict information to indicate sample noisiness (e.g. a triplet whose CMA gradients are in more agreement is considered less “noisy" and would be prioritized for cross-modal pre-training) and a curriculum learning strategy is developed accordingly. Both techniques are found to boost VATT performance on downstream tasks, not only on instructional videos, but even more when the more complicated non-narrative data is involved.

Our contributions can be summarized as the following:

- We suggest that the commonly assumed CMA might be an important yet overlooked performance hurdle for scaling up multimodal pre-training, and we observe severe misalignment even from the state-of-the-art (SOTA) model [6] on the relatively aligned data [124].
- We propose to consistently improve the pre-training pipeline in the only baseline in this direction, VATT (modality-agnostic setting), resulting in better downstream performance.
- With the help of our proposed techniques, we consistently improve the pre-training of VATT, resulting in better downstream performance, e.g. up to 58% gain in the rank metric of video-text retrieval task. Moreover, we are able to scale VATT pre-training to more complicated non-narrative Youtube8M dataset, which further yields new state-of-the-art performance in a modality-agnostic setting.

4.2 Related Work

4.2.1 Self-Supervised Multimodal Pre-training

The most organic self-supervision can arguably be found in the multimodal data that are abundantly available in the digital world: their temporal alignment naturally lead to cross-modality mutual supervision, hence meaningful features can be learned without requiring human annotation.

Multimodal pre-training can thus leverage richer semantic cues than single-modality counterparts, and outperform in various downstream tasks such as image classification [118, 6], video action recognition [118, 6, 125, 123], audio event classification [118, 6, 123], and video-text retrieval [118, 6, 125]. Most of those methods formulate pre-text tasks, including cross-modality correspondence [119, 120, 121, 123], cross-modality clustering [130], cross-modality longer context prediction [131], or a combination of multiple pre-text tasks [17].

Contrastive learning [132, 133] has recently emerged as one dominant approach in self-supervised learning. In multimodal pre-training, the temporal co-occurrence naturally supplies the positive sample to contrast with. [118] adopts a multimodal versatile network to embed each modality into the same vector space, that is trained from end to end with multimodal pairwise contrastive losses (video-audio, and video text). CrossCLR [122] further takes the intra-modality similarities into account. [123] generalizes the instance discrimination idea to design a cross-modal discrimination task, i.e., predicting which audio matches a video. Most recently, VATT [6] combines the strength of convolution-free Transformer and multimodal contrastive learning. Specifically, it first validate the idea of using a modality-agnostic single-backbone transformer, to work on video, audio and text modalities, it follows the exact BERT [13] and ViT [114] architecture, except injecting modality-specific tokenization layers and linear projections.

Despite their success, existing multimodal contrastive learning methods [118, 122, 6] hinge on the “free" cross-modalities correspondence mined through the multimodal contrastive loss, i.e., the CMA assumption. As a result, they are often trained with well curated or narrated video datasets, such as AudioSet [134], YouCook2 [127], and HowTo100M [124]. It remains questionable whether such CMA-guided contrastive learning can generalize well to larger-scale multimodal data in the wild. In fact, even in the HowTo100M dataset, the authors of [124] estimate that around 50% of clip-narration pairs are not well aligned. The validity of CMA has arisen concerns. Most related to us is [125] which explicitly models the misalignment noise using the MIL-NCE loss, that is inherited by VATT [6].

4.2.2 Multi-task versus Multimodal Learning

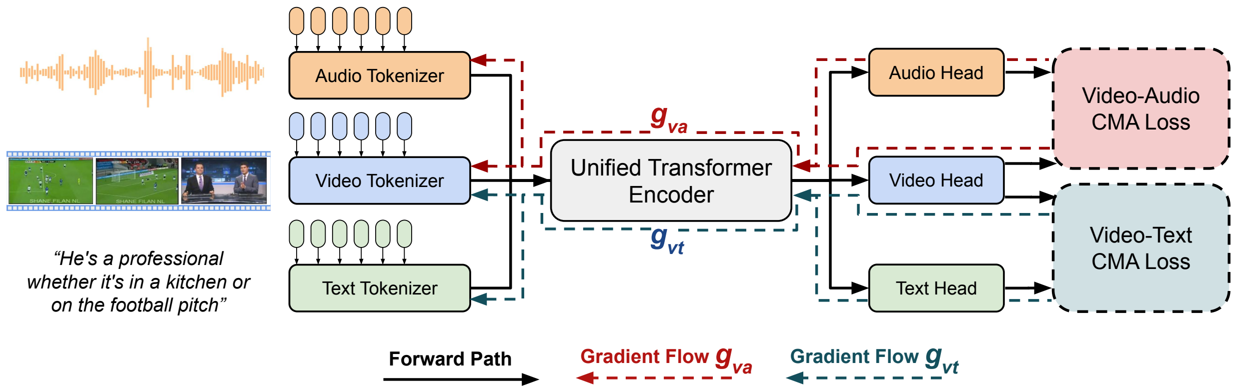


Figure 4.3: One main model used in this paper: the VATT model [6] with a modality-agnostic, single-backbone Transformer by sharing weights among the three modalities. VATT is trained with two contrastive losses between video-audio and video-text pairs, respectively. The positive cross-modality pairs to contrastive with are based on their temporal co-occurrence (e.g., the CMA assumption). We primarily study the conflicts between gradients g_{va} and g_{vt}, which indicate the often poor alignments, and discuss how to harmonize their conflicts. Reprinted with permission from [5].

While earlier self-supervised works used different modality-specific encoders, the latest multimodal pre-training sees a tendency to explore a versatile and "universal" model shared across modalities [118, 6]. It is easy to notice the resemblance between multimodal learning (especially with a unified backbone) and multi-task learning [135, 136]. The later often assumes multiple tasks to share transferable knowledge and can help each other learn. Yet in practice, different tasks can be heterogeneous in nature too, often leading to heavily misaligned gradient scales [137] or directions [129]. Such conflicting gradients from multiple losses would lead to several undesirable effects, including (i) strong bias towards learning some tasks with largest gradient scales; and (2) slow and unstable convergence. Similar problems are often faced by multimodal pre-training too.

As a result, several works have been developed to mitigate the gradient dilemma in multi-task learning. Methods such as gradient normalization [137] and adaptive loss weight [138, 139] address the gradient scale/learning speed imbalance issues. Gradient surgery [129] tries to project gradients with conflicting directions onto the normal plane of each other, to prevent each gradient's interfering component from being applied to the network. Later, Gradient vaccine [140] further generalizes this idea extending the gradient projection to arbitrary angle, and uses the exponential moving average of cosine similarity to measure the alignment between different tasks. When it comes to the multimodal learning, [141] investigates optimally blending modality-specific gradients by linearly re-scaling them. Meanwhile, to our best knowledge, there has been no prior study on modality-specific gradient directions.

4.3 Methodology

Pre-requisites: Without loss of generality, we focus on the SOTA multimodal contrastive learning model, VATT [6], as our default subject of study; and we follow its modality-agnostic single-backbone setting due to its compelling performance-efficiency trade-off, and due to the fact that it is the most intuitive setting to modify gradients (shared between paired modalities) w.r.t different end-point objectives.

In our modality-agnostic single-backbone setting of VATT, the goal of pre-training is to find the parameter θ of the model f_θ to learn meaningful features for all target modalities (video, audio, text), usually measured by a set of pairwise similarity metrics. VATT uses two pairwise contrastive losses for video-audio and video-text respectively to solve cross-modal alignment:

$$\min_{\theta} \ \text{CMA}_{va}(\theta) + \text{CMA}_{vt}(\theta) \tag{4.1}$$

where CMA_{va} and CMA_{vt} penalize the cross-modal alignment for video-audio and video-text pairs, using the Noise-Contrastive Estimation objective [142], respectively.

Figure 4.3 illustrates the VATT pipeline. First, video-audio-text triplets are sampled from random temporal locations, and then video-text and video-audio pairs are formed accordingly. Positive pairs are formed by selecting the two modalities at the same temporal locations from the same video clip; while the negative pairs are obtained by randomly sampling any video, audio, or text from other video clips. We follow the convention in [118, 6] to use the vanilla NCE loss for video-audio pairs, and to use the MIL-NCE loss proposed in [125] to align video-text pairs. Hence, CMA_{va} and CMA_{vt} are defined as follows:

$$\mathrm{CMA}_{va}(\theta) = -\log \left(\frac{\exp(z_v^\top z_a/\tau)}{\exp(z_v^\top z_a/\tau) + \sum_{z' \in \mathcal{N}} \exp({z'}_v^\top {z'}_a/\tau)} \right) \tag{4.2}$$

$$\mathrm{CMA}_{vt}(\theta) = -\log \left(\frac{\sum_{z_t \in \mathcal{P}_k(z_t)} \exp(z_v^\top z_t/\tau)}{\sum_{z_t \in \mathcal{P}_k(z_t)} \exp(z_v^\top z_t/\tau) + \sum_{z' \in \mathcal{N}} \exp({z'}_v^\top {z'}_t/\tau)} \right) \tag{4.3}$$

where $\mathcal{N}$ is the pool of negative pairs, $\mathcal{P}_k(z_t)$ denotes the k-neighbouring narrations surrounding the original text z_t. The gradients of two pairwise losses CMA_{va} and CMA_{vt}, denoted as g_{va} and g_{vt}, will be together applied (i.e. simple adding) to updating the model f_θ.

4.3.1 Observation of Conflicting Gradients

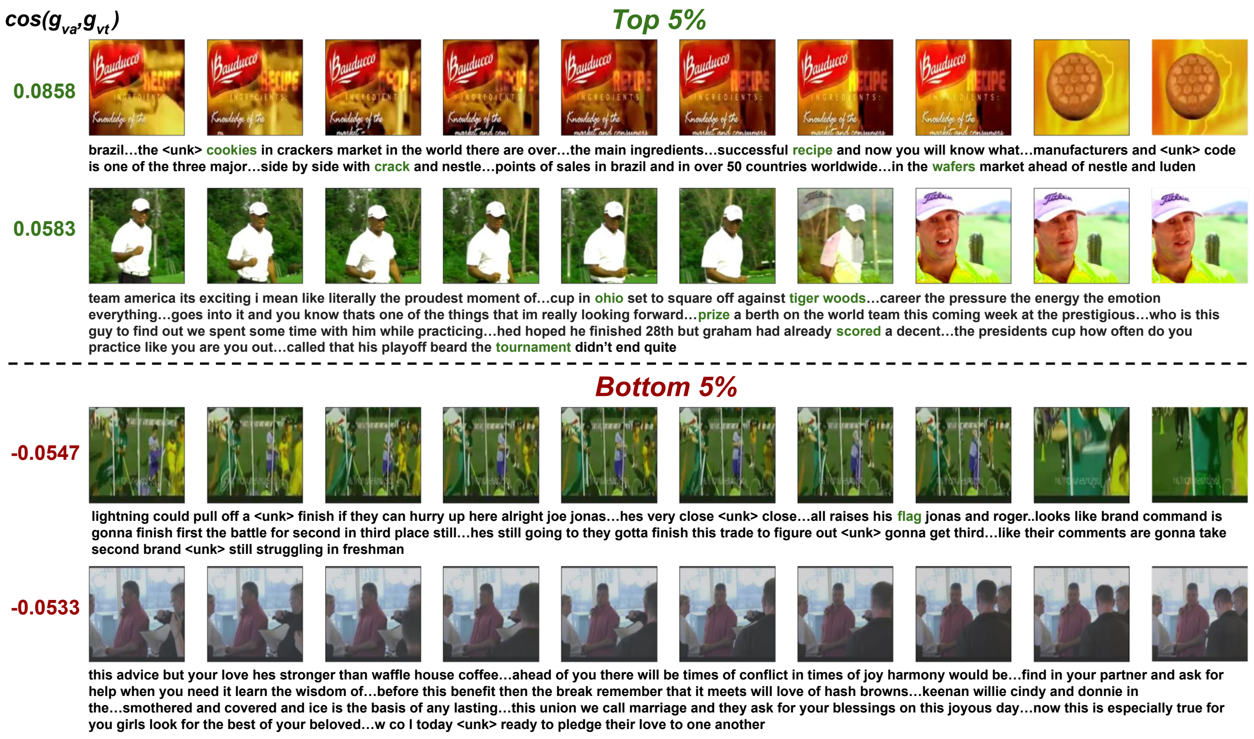
While qualitatively recognizing the cross-modal misalignment is easy, it is a highly challenging endeavor to quantify the misalignment and to automatically find misaligned triplets at scale, due to the vaguely defined criterion as well as the absence of labels [125, 126]. Therefore, the foremost need is to identify a *surrogate indicator* on how noisy the alignment of a sample triplet is, without making unrealistic assumptions nor incurring much computational overhead.

Our empirical choice of surrogate indicator is the directional alignability (i.e., vector angle) of gradients generated by different pairwise cross-modal losses, i.e., the **cosine similarity** between g_{va} and g_{vt} in the particular example of VATT, after flattening them into two vectors. The choice is firstly rationalized by the *coherent gradient hypothesis* [143], that healthy model training should witness the overall gradient stronger in certain directions along which the samples' gradients reinforce each other. Secondly, comparing the gradient directions between multiple pairwise losses could be considered as an ensemble or “cross-check": intuitively, if both video-audio and video-text pairwise consistency lead to the same update direction, then there is a good chance that those modalities are well aligned and the update direction is reliable; otherwise, at least one pair (video-audio, or video-text) might suffer from misalignment and provides noisy cross-modality guidance.

Overall, we conjecture that: *aligned video-text-audio triplets should have higher cosine simi-*

larity for g_{va} and g_{vt}, and vice versa.

To further validate the conjecture, we conduct a sanity check, where we started from a pre-trained VATT network and further optimize it for 200 iterations with a batch size of 64, on Youtube8M dataset. We then randomly sample video-text-audio triplets out of the samples with top 5% and bottom 5% most aligned gradient, measured by $cos(g'_{va}, g'_{va})$. In Figure 4.4, we visualize 10 frames within a 32-frame video clip and its corresponding text narration (including 8 neighbouring text narrations similar to [125]). We visually observed that the top 5% group triplets has more semantically aligned words in the corresponding text narration (highlighted in **green**) thus enjoy a better cross-modality alignment, while the bottom 5% group triplets has fewer semantically aligned words, therefore are much more noisy in their alignments. We include more visualization samples in Figure 4.5 and 4.6.



brazil…the <unk> cookies in crackers market in the world there are over…the main ingredients…successful recipe and now you will know what…manufacturers and <unk> code is one of the three major…side by side with crack and nestle…points of sales in brazil and in over 50 countries worldwide…in the wafers market ahead of nestle and luden

team america its exciting i mean like literally the proudest moment of…cup in ohio set to square off against tiger woods…career the pressure the energy the emotion everything…goes into it and you know thats one of the things that im really looking forward…prize a berth on the world team this coming week at the prestigious…who is this guy to find out we spent some time with him while practicing…hed hoped he finished 28th but graham had already scored a decent…the presidents cup how often do you practice like you are you out…called that his playoff beard the tournament didn't end quite

lightning could pull off a <unk> finish if they can hurry up here alright joe jonas…hes very close <unk> close…all raises his flag jonas and roger..looks like brand command is gonna finish first the battle for second in third place still…hes still going to they gotta finish this trade to figure out <unk> gonna get third…like their comments are gonna take second brand <unk> still struggling in freshman

this advice but your love hes stronger than waffle house coffee…ahead of you there will be times of conflict in times of joy harmony would be…find in your partner and ask for help when you need it learn the wisdom of…before this benefit then the break remember that it meets will love of hash browns…keenan willie cindy and donnie in the…smothered and covered and ice is the basis of any lasting…this union we call marriage and they ask for your blessings on this joyous day…now this is especially true for you girls look for the best of your beloved…w co l today <unk> ready to pledge their love to one another

Figure 4.4: Qualitative example of our proposed measure based on agreement of gradients on Youtube8M dataset, semantically aligned words are highlighted in **green**, $cos(g_{va}, g_{vt})$ reflect the agreement of between g_{va} and g_{vt}, by measuring their cosine similarity. Reprinted with permission from [5].

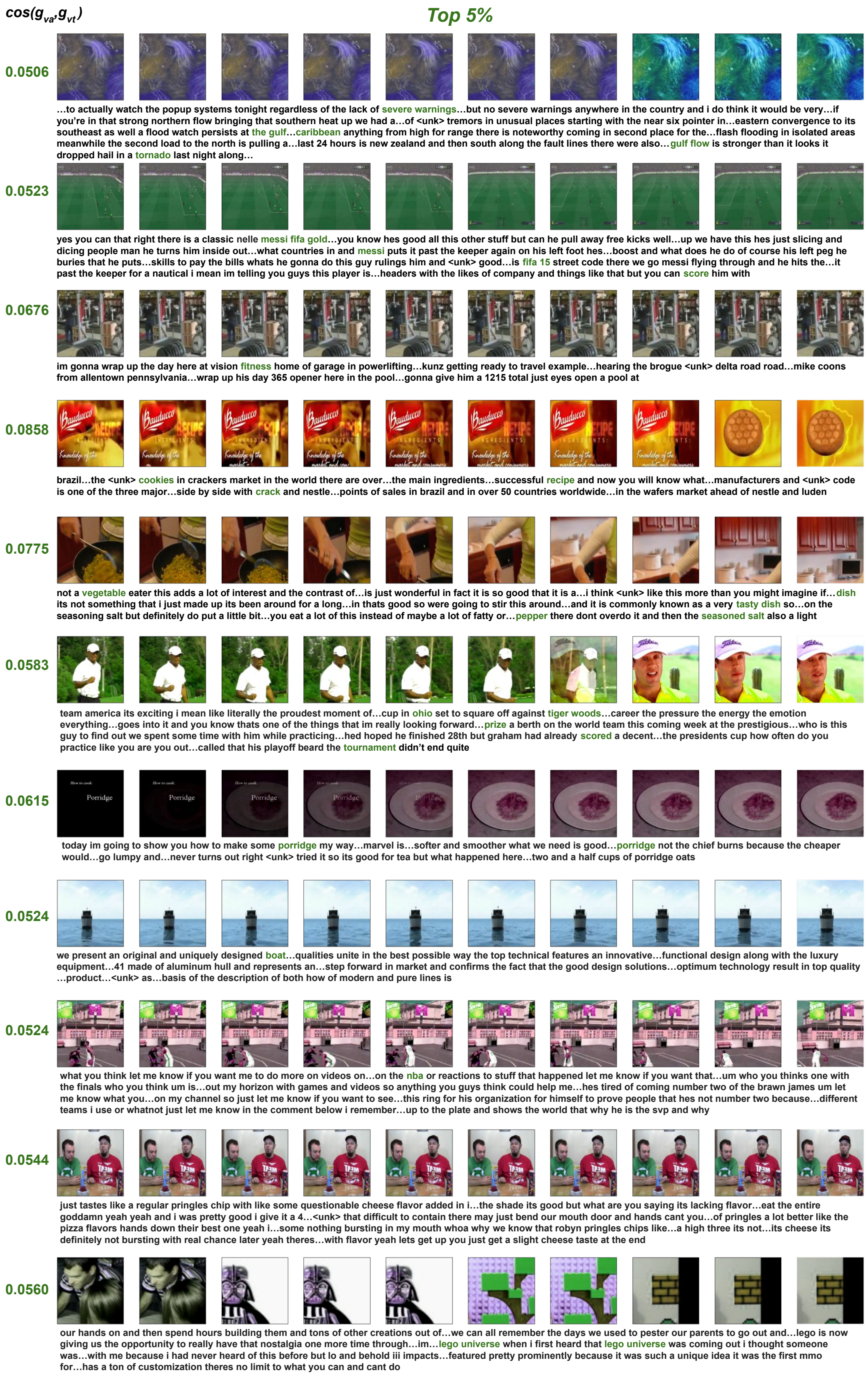
With the assumption that misalignment in gradients correlates to the noisiness of the sample's cross-modality alignment, we use the former as the computable surrogate indicator of the later, and examine the cosine similarity between gradient g_{va} and g_{vt} (over 500k iterations) of VATT on the HowTo100M dataset [124]. In Figure 4.7a, we plot the distribution of cosine similarities, $cos(g_{va}, g_{vt})$, across 500k iterations of pre-training. We observe that $cos(g_{va}, g_{vt})$ at any iteration resembles a normal distribution, and about half of the gradients g_{va} and g_{va} have misaligned directions (negative cosine similarities). The plot echos the empirical finding in [127] that around 50% of video-text pairs are not well-aligned on the HowTo100M dataset.

In Figure 4.8, we plot the distribution of cosine similarities, $cos(g_{va}, g_{va})$, across 500k iterations of pre-training, on the noisy Youtube8M dataset. We observe that $cos(g_{va}, g_{va})$ at any iteration also resembles a normal distribution, and about half of the gradients g_{va} and g_{va} have misaligned directions (negative cosine similarities). We also calculate the mean of $cos(g_{va}, g_{va})$ across all 500k iterations on Youtube8M dataset, and found it to be 30% smaller than that on Howto100M dataset, indicating stronger mis-alignment in gradient directions on Youtube8M, which further verify that non-narrative video sets such as Youtube8M have more severe misalignment than the narrative ones such as Howto100M, hence challenging the CMA assumption commonly used in multimodal pre-training[6, 125].

The existence of conflicting gradients not only makes the convergence harder, but also makes the learned representation heavily overfit some modalities. In our case, we observe the training to be biased towards video/audio modalities, which favors the downstream performance of video/audio classification task, yet sets back the performance of video-to-text retrieval task (verified in Table 4.1).

4.3.2 Harmonizing Gradients during Pre-training

We present two strategies to mitigate the conflicting gradients: a “hard" way to project gradient directions, and a “soft" way to progressively select sample triplets in curriculum.



0.0506

…to actually watch the popup systems tonight regardless of the lack of severe warnings…but no severe warnings anywhere in the country and i do think it would be very…if you're in that strong northern flow bringing that southern heat up we had a…of <unk> tremors in unusual places starting with the near six pointer in…eastern convergence to its southeast as well a flood watch persists at the gulf…caribbean anything from high for range there is noteworthy coming in second place for the…flash flooding in isolated areas meanwhile the second load to the north is pulling a…last 24 hours is new zealand and then south along the fault lines there were also…gulf flow is stronger than it looks it dropped hail in a tornado last night along…

0.0523

yes you can that right there is a classic nelle messi fifa gold…you know hes good all this other stuff but can he pull away free kicks well…up we have this hes just slicing and dicing people man he turns him inside out…what countries in and messi puts it past the keeper again on his left foot hes…boost and what does he do of course his left peg he buries that he puts…skills to pay the bills whats he gonna do this guy rulings him and <unk> good…is fifa 15 street code there we go messi flying through and he hits the…it past the keeper for a nautical i mean im telling you guys this player is…headers with the likes of company and things like that but you can score him with

0.0676

im gonna wrap up the day here at vision fitness home of garage in powerlifting…kunz getting ready to travel example…hearing the brogue <unk> delta road road…mike coons from allentown pennsylvania…wrap up his day 365 opener here in the pool…gonna give him a 1215 total just eyes open a pool at

0.0858

brazil…the <unk> cookies in crackers market in the world there are over…the main ingredients…successful recipe and now you will know what…manufacturers and <unk> code is one of the three major…side by side with crack and nestle…points of sales in brazil and in over 50 countries worldwide…in the wafers market ahead of nestle and luden

0.0775

not a vegetable eater this adds a lot of interest and the contrast of…is just wonderful in fact it is so good that it is a…i think <unk> like this more than you might imagine if…dish its not something that i just made up its been around for a long…in thats good so were going to stir this around…and it is commonly known as a very tasty dish so…on the seasoning salt but definitely do put a little bit…you eat a lot of this instead of maybe a lot of fatty or…pepper there dont overdo it and then the seasoned salt also a light

0.0583

team america its exciting i mean like literally the proudest moment of…cup in ohio set to square off against tiger woods…career the pressure the energy the emotion everything…goes into it and you know thats one of the things that im really looking forward…prize a berth on the world team this coming week at the prestigious…who is this guy to find out we spent some time with him while practicing…hed hoped he finished 28th but graham had already scored a decent…the presidents cup how often do you practice like you are you out…called that his playoff beard the tournament didn't end quite

0.0615

today im going to show you how to make some porridge my way…marvel is…softer and smoother what we need is good…porridge not the chief burns because the cheaper would…go lumpy and…never turns out right <unk> tried it so its good for tea but what happened here…two and a half cups of porridge oats

0.0524

we present an original and uniquely designed boat…qualities unite in the best possible way the top technical features an innovative…functional design along with the luxury equipment…41 made of aluminum hull and represents an…step forward in market and confirms the fact that the good design solutions…optimum technology result in top quality …product…<unk> as…basis of the description of both how of modern and pure lines is

0.0524

what you think let me know if you want me to do more on videos on…on the nba or reactions to stuff that happened let me know if you want that…um who you thinks one with the finals who you think um is…out my horizon with games and videos so anything you guys think could help me…hes tired of coming number two of the brawn james um let me know what you…on my channel so just let me know if you want to see…this ring for his organization for himself to prove people that hes not number two because…different teams i use or whatnot just let me know in the comment below i remember…up to the plate and shows the world that why he is the svp and why

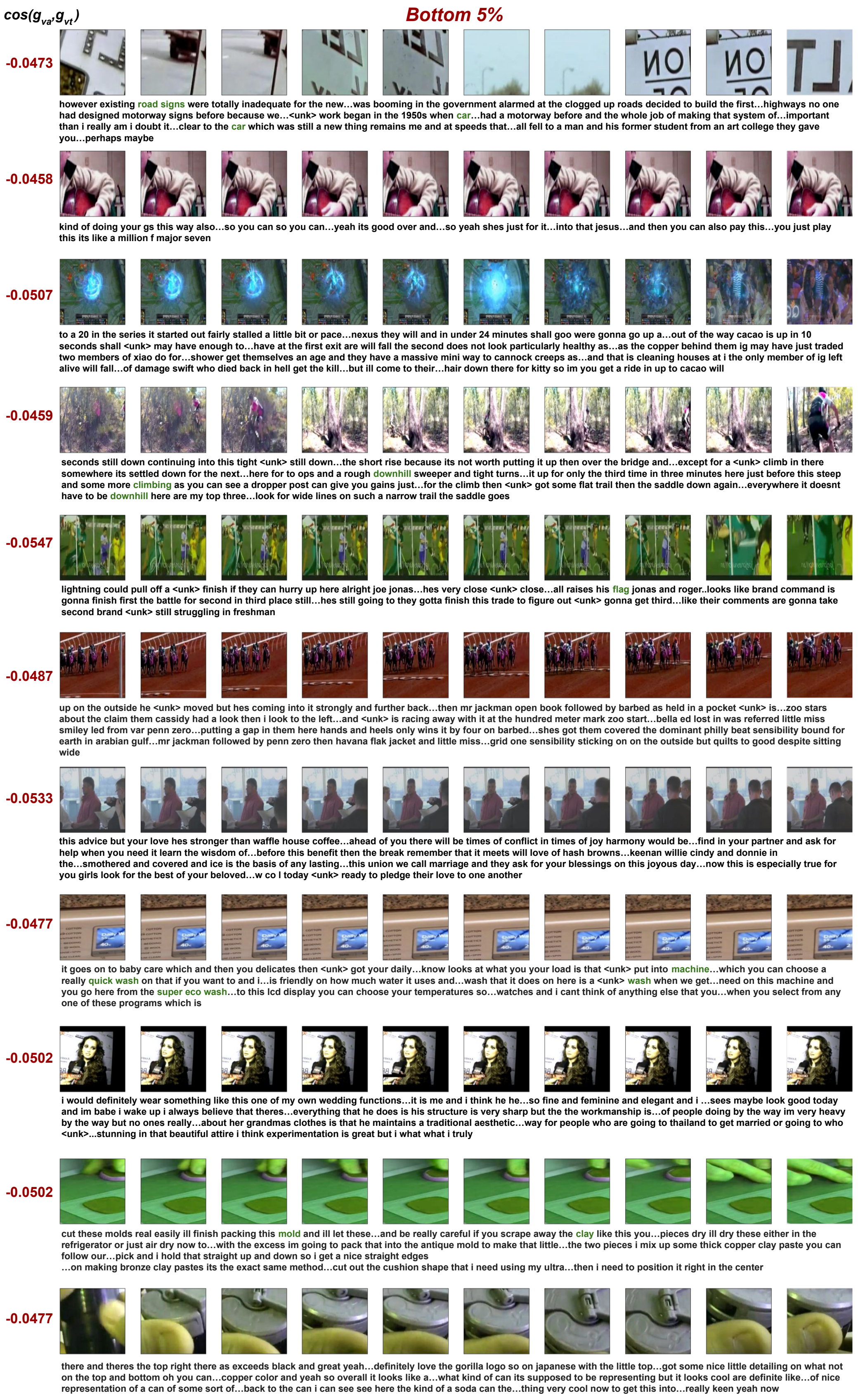
0.0544

just tastes like a regular pringles chip with like some questionable cheese flavor added in i…the shade its good but what are you saying its lacking flavor…eat the entire goddamn yeah yeah and i was pretty good i give it a 4…<unk> that difficult to contain there may just bend our mouth door and hands cant you…of pringles a lot better like the pizza flavors hands down their best one yeah i…some nothing bursting in my mouth whoa why we know that robyn pringles chips like…a high three its not…its cheese its definitely not bursting with real chance later yeah theres…with flavor yeah lets get up you just get a slight cheese taste at the end

0.0560

our hands on and then spend hours building them and tons of other creations out of…we can all remember the days we used to pester our parents to go out and…lego is now giving us the opportunity to really have that nostalgia one more time through…im…lego universe when i first heard that lego universe was coming out i thought someone was…with me because i had never heard of this before but lo and behold iii impacts…featured pretty prominently because it was such a unique idea it was the first mmo for…has a ton of customization theres no limit to what you can and cant do

Figure 4.5: Visualization of Top 5% examples measured by the agreement of gradients on Youtube8M dataset, semantically aligned text are highlighted in **green**, $cos(g_{va}, g_{vt})$ reflect the agreement of between g_{va} and g_{vt}, by measuring their cosine similarity. Reprinted with permission from [5].



-0.0473

however existing road signs were totally inadequate for the new…was booming in the government alarmed at the clogged up roads decided to build the first…highways no one had designed motorway signs before because we…<unk> work began in the 1950s when car…had a motorway before and the whole job of making that system of…important than i really am i doubt it…clear to the car which was still a new thing remains me and at speeds that…all fell to a man and his former student from an art college they gave you…perhaps maybe

-0.0458

kind of doing your gs this way also…so you can so you can…yeah its good over and…so yeah shes just for it…into that jesus…and then you can also pay this…you just play this its like a million f major seven

-0.0507

to a 20 in the series it started out fairly stalled a little bit or pace…nexus they will and in under 24 minutes shall goo were gonna go up a…out of the way cacao is up in 10 seconds shall <unk> may have enough to…have at the first exit are will fall the second does not look particularly healthy as…as the copper behind them ig may have just traded two members of xiao do for…shower get themselves an age and they have a massive mini way to cannock creeps as…and that is cleaning houses at i the only member of ig left alive will fall…of damage swift who died back in hell get the kill…but ill come to their…hair down there for kitty so im you get a ride in up to cacao will

-0.0459

seconds still down continuing into this tight <unk> still down…the short rise because its not worth putting it up then over the bridge and…except for a <unk> climb in there somewhere its settled down for the next…here for to ops and a rough downhill sweeper and tight turns…it up for only the third time in three minutes here just before this steep and some more climbing as you can see a dropper post can give you gains just…for the climb then <unk> got some flat trail then the saddle down again…everywhere it doesnt have to be downhill here are my top three…look for wide lines on such a narrow trail the saddle goes

-0.0547

lightning could pull off a <unk> finish if they can hurry up here alright joe jonas…hes very close <unk> close…all raises his flag jonas and roger..looks like brand command is gonna finish first the battle for second in third place still…hes still going to they gotta finish this trade to figure out <unk> gonna get third…like their comments are gonna take second brand <unk> still struggling in freshman

-0.0487

up on the outside he <unk> moved but hes coming into it strongly and further back…then mr jackman open book followed by barbed as held in a pocket <unk> is…zoo stars about the claim them cassidy had a look then i look to the left…and <unk> is racing away with it at the hundred meter mark zoo start…bella ed lost in was referred little miss smiley led from var penn zero…putting a gap in them here hands and heels only wins it by four on barbed…shes got them covered the dominant philly beat sensibility bound for earth in arabian gulf…mr jackman followed by penn zero then havana flak jacket and little miss…grid one sensibility sticking on on the outside but quilts to good despite sitting wide

-0.0533

this advice but your love hes stronger than waffle house coffee…ahead of you there will be times of conflict in times of joy harmony would be…find in your partner and ask for help when you need it learn the wisdom of…before this benefit then the break remember that it meets will love of hash browns…keenan willie cindy and donnie in the…smothered and covered and ice is the basis of any lasting…this union we call marriage and they ask for your blessings on this joyous day…now this is especially true for you girls look for the best of your beloved…w co l today <unk> ready to pledge their love to one another

-0.0477

it goes on to baby care which and then you delicates then <unk> got your daily…know looks at what you your load is that <unk> put into machine…which you can choose a really quick wash on that if you want to and i…is friendly on how much water it uses and…wash that it does on here is a <unk> wash when we get…need on this machine and you go here from the super eco wash…to this lcd display you can choose your temperatures so…watches and i cant think of anything else that you…when you select from any one of these programs which is

-0.0502

i would definitely wear something like this one of my own wedding functions…it is me and i think he he…so fine and feminine and elegant and i …sees maybe look good today and im babe i wake up i always believe that theres…everything that he does is his structure is very sharp but the the workmanship is…of people doing by the way im very heavy by the way but no ones really…about her grandmas clothes is that he maintains a traditional aesthetic…way for people who are going to thailand to get married or going to who <unk>...stunning in that beautiful attire i think experimentation is great but i what what i truly

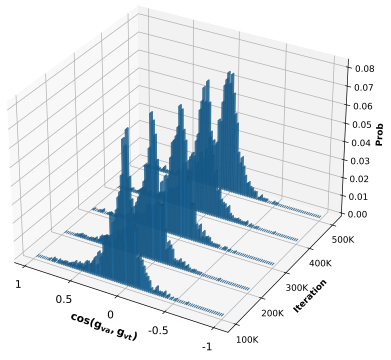
-0.0502

cut these molds real easily ill finish packing this mold and ill let these…and be really careful if you scrape away the clay like this you…pieces dry ill dry these either in the refrigerator or just air dry now to…with the excess im going to pack that into the antique mold to make that little…the two pieces i mix up some thick copper clay paste you can follow our…pick and i hold that straight up and down so i get a nice straight edges
…on making bronze clay pastes its the exact same method…cut out the cushion shape that i need using my ultra…then i need to position it right in the center

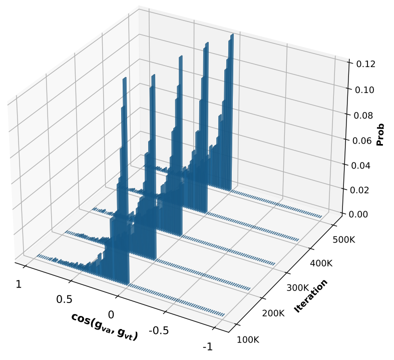
-0.0477

there and theres the top right there as exceeds black and great yeah…definitely love the gorilla logo so on japanese with the little top…got some nice little detailing on what not on the top and bottom oh you can…copper color and yeah so overall it looks like a…what kind of can its supposed to be representing but it looks cool are definite like…of nice representation of a can of some sort of…back to the can i can see see here the kind of a soda can the…thing very cool now to get this into…really keen yeah now

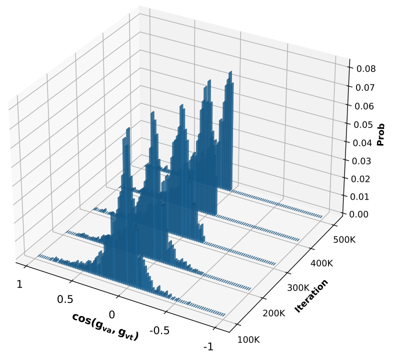
Figure 4.6: Visualization of Bottom 5% examples measured by the agreement of gradients on Youtube8M dataset, semantically aligned text are highlighted in **green**, $cos(g_{va}, g_{vt})$ reflect the agreement of between g_{va} and g_{vt}, by measuring their cosine similarity. Reprinted with permission from [5].



(a) Multimodal pre-training baseline

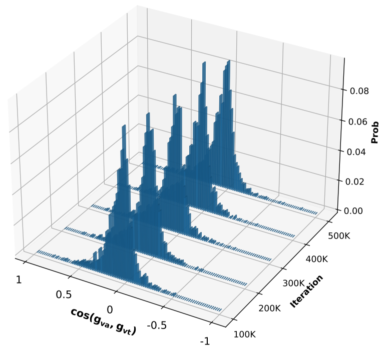


(b) w/ Cross-Modality Gradient Realignment

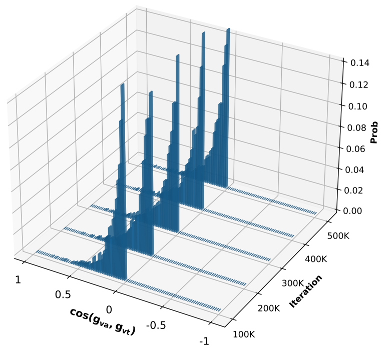


(c) w/ Gradient-based Curriculum Learning

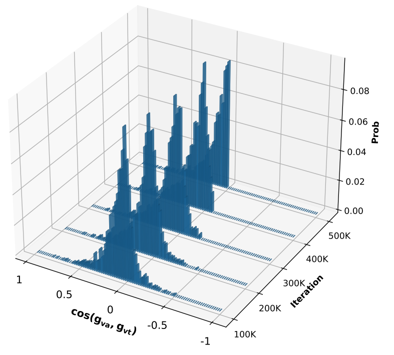
Figure 4.7: Visualization of cosine similarity between gradient g_{va} and g_{vt} across 500K Iterations on the HowTo100M dataset. (a) shows the baseline setting in multimodal pre-training; (b) with only cross-modality gradient realignment; (c) with only gradient-based curriculum learning. Reprinted with permission from [5].



(a) Multimodal pre-training baseline



(b) w/ Cross-Modality Gradient Realignment



(c) w/ Gradient-based Curriculum Learning

Figure 4.8: Visualization of cosine similarity between gradient g_{va} and g_{vt} across 500K Iterations on the Youtube8M dataset. (a) shows the baseline setting in multimodal pre-training; (b) with only cross-modality gradient realignment; (c) with only gradient-based curriculum learning. Reprinted with permission from [5].

Algorithm 1 Cross-Modality Gradient Realignment

```
**Require:** Model parameter $\theta$ , minibatchs $\mathcal{B}_{va}$, minibatchs $\mathcal{B}_{vt}$
  for $(b_{va}, b_{vt}) \in (\mathcal{B}_{va}, \mathcal{B}_{vt})$ do
      $g_{va} \leftarrow \nabla_\theta \text{CMA}_{va}(\theta)$, $g_{vt} \leftarrow \nabla_\theta \text{CMA}_{vt}(\theta)$
      $g_{va} \leftarrow \text{flatten}(g_{va})$, $g_{vt} \leftarrow \text{flatten}(g_{vt})$
      $\hat{g}_{va} \leftarrow g_{va}$, $\hat{g}_{vt} \leftarrow g_{vt}$
      if $g_{va} \cdot g_{vt} < 0$ then
          $\hat{g}_{va} \leftarrow \hat{g}_{va} - \frac{\hat{g}_{va} \cdot g_{vt}}{||g_{vt}||^2}$        ▷ projection $g_{vt} \leftarrow g_{va}$
          $\hat{g}_{vt} \leftarrow \hat{g}_{vt} - \frac{\hat{g}_{vt} \cdot g_{va}}{||g_{va}||^2}$        ▷ projection $g_{va} \leftarrow g_{vt}$
      end if
      $\hat{g}_{va} \leftarrow \text{reshape}(\hat{g}_{va})$, $\hat{g}_{vt} \leftarrow \text{reshape}(\hat{g}_{vt})$
      $\Delta\theta \leftarrow \hat{g}_{vt} + \hat{g}_{va}$        ▷ sum up gradients
      update $\theta$ with $\Delta\theta$        ▷ update parameter
  end for
```

4.3.2.1 Cross-Modality Gradient Realignment

Since g_{va} and g_{vt} are misaligned in terms of their directions, one natural idea is to re-align the cross-modality gradients, by enforcing or re-projecting their directions to be aligned. An intuitive choice is an effective remedy developed for multi-task learning named Gradient Surgery [129]. Here, instead of applying it to g_i and g_j where i, j are the two heterogeneous tasks, we apply it to g_{va} and g_{vt}, which refer to the gradients of the two pairwise losses CMA$_{va}$ and CMA$_{vt}$ w.r.t the weights in the model, following our rationale in Section 4.3.1.

If the gradients g_{va} and g_{vt} have negative cosine similarity, we alter them by projecting each onto the normal plane of the other. Otherwise, if g_{va} and g_{vt} have zero or positive similarity, we retain the original gradients. We refer to this particularly adapted form of gradient surgery as *Cross-Modality Gradient Realignment*. Its update procedure is outlined in Algorithm 1, and the overhead very cheap, mainly just computing inner products of two flattened gradients. We also tried another similar algorithm named Gradient Vaccine [140], and the results are almost identical to using Gradient Surgery [129]. We hence only report one for the sake of conciseness.

4.3.2.2 Gradient-based Curriculum Learning

We next propose a gradient-based curriculum learning strategy that gradually identifies and removes more misaligned samples based on the similarity indicator as the training goes on. Formally, given a video-audio pair b_{va} and video-text pair b_{vt}, we only update the parameter if $cos(g_{va}, g_{vt}) > \gamma$ and drop this sample triplet otherwise. This ensures us to gradually prioritize training with well-aligned samples while suppressing the noisy supervision from potentially misaligned samples. The drop rate here is controlled by a cut-off threshold γ. In general, γ is a negative value monotonically increasing during training, and there is an underlying rationale.

Algorithm 2 Gradient-based Curriculum Learning

```
Require: Model parameter $\theta$, minibatchs $\mathcal{B}_{va}$, minibatchs $\mathcal{B}_{vt}$, initial $\gamma_0$
  for $(b_{va}, b_{vt}) \in (\mathcal{B}_{va}, \mathcal{B}_{vt})$ do
      update $\gamma$                                              ▷ curriculumly update $\gamma$
      $g_{va} \leftarrow \nabla_\theta \text{CMA}_{va}(\theta)$, $g_{vt} \leftarrow \nabla_\theta \text{CMA}_{vt}(\theta)$
      $g_{va} \leftarrow \text{flatten}(g_{va})$, $g_{vt} \leftarrow \text{flatten}(g_{vt})$
      if $g_{va} \cdot g_{vt} > \gamma$ then
          $g_{va} \leftarrow \text{reshape}(g_{va})$, $g_{vt} \leftarrow \text{reshape}(g_{vt})$
          $\Delta\theta \leftarrow g_{va} + g_{vt}$                          ▷ sum up gradients
          update $\theta$ with $\Delta\theta$                               ▷ update parameter
      end if
  end for
```

Initially, the network is under-trained and the similarity between its learned features is not reliable to indicate a strong CMA; hence we are more conservative in excluding samples. As the network becomes more sufficiently trained, the features become more semantically aligned, allowing us to remove more misaligned samples in the later stage of training. The full procedure of Gradient-based Curriculum Learning is outlined in Algorithm 2. We empirically choice $\gamma_0 = -0.3$ at the beginning and linearly increase it to $\gamma_{500K} = 0$ at the end, we include an ablation on the adaptive choice of γ in the Table 4.2.

Similar classes of ideas, i.e, “gradually focusing a small training subset", have been explored by a prominent class of algorithms which rely on sample selection to robustify [144, 145, 146] or to accelerate [147] training. Most of them use the “small loss trick" wherein a fraction of samples with smallest loss values below a certain threshold are considered as reliable. However, they did not explore gradient conflict indicators, which targets solving our CMA problem here. Also note that our two steps can fully reuse the cosine similarity computed, so applying them together incurs no extra overhead.

4.4 Experiments

4.4.1 Pre-Training Datasets and Settings

In this paper, we focus on studying gradients in a modality-agnostic single-backbone setting, thus we do not consider multimodal pre-training baseline with modality-specific models, e.g. HERO[148], MMV[118], PerceiverIO[149], AVLnet[150]. Apart from VATT[6], there exist other concurrent work that use modality-agnostic setting[17, 151, 152], yet they all have their respective limitations. For example, UniVL[17] lacks audio modalities thus cannot constitute video-audio CMA loss, VLM[151] applies separate attention mask for each modality while PolyViT[152] use alternate training between modalities and tasks, thus eliminate the possibilities of any gradients conflicts. Therefore, this leave VATT[6] to be the most suitable baseline besides it SoTA performance.

Our pre-training adopts three settings for comprehensive comparison: **(1)** using HowTo100M as our training set, following [17, 124]; **(2)** combining HowTo100M and AudioSet as our training set following [118] and VATT [6]; and **(3)** additionally merging a noisy Youtube8M dataset alongside with HowTo100M and AudioSet, to create a much more compound and complicated training set than considered before, to stretch-test our proposal’s benefits to scale up pre-training in uncurated and poorly aligned data.

For each setting, we i.i.d sample video-audio-text clips from the mixture of all candidate sets in every batch. (HowTo100M, Audioset or Youtube8M, if applicable). For all setting, we only use a subset of HowTo100M, AudioSet, Youtube8M, Kinetics400 and Youcook2 dataset in compliance with Youtube's wipe-out policies.

HowTo100M[1][124] is a large-scale dataset of narrated videos with an emphasis on instructional videos that have well synchronized modalities. The training set consists of 127 Millions video clips with the corresponding close-captioned text from automatic speech recognition (ASR).

AudioSet[1][134] is a large-scale audio-visual dataset originally intended for audio event classification. It consists of 1.78 Million 10-second clips sampled from Youtube Videos that contains a variety of audio tracks, such as musical instruments, animals or mechanical sounds.

Youtube8M[1][4] is a large-scale video classification dataset based on Youtube videos. It consists of a diverse vocabulary of 3862 visual entities and a total number of 8 Million videos. To the best of our knowledge, we are the first to scale contrastive multimodal pre-training beyond instructional videos, onto this noisy Youtube8M videos that more faithfully represent videos the wild. For data pre-processing, we split each video into 5s clips and use ASR close-captions to generate the corresponding text, resulting in 165 Millions video-audio-text clips.

4.4.2 Pre-Training Settings

Network Architecture: For all of our experiments, we use modality-agnostic Transformer variant in [6], VATT-MA-Medium. Specifically, we use a 12-layer Transformer, in which each layer has a feed-forward MLP with hidden size of 4096, and 16 attention heads with a hidden size of 1024.

Pre-training Hyperparameter: We strictly follow the setting in [6], pre-training VATT from scratch with Adam optimizer with an initial learning rate of 1e-4, 10k warmup steps, 500k steps in total, a batch size of 2048 and using a cosine learning rate scheduler to anneal the learning rate from 1e-4 to 5e-5. Our framework is implemented in Tensorflow 2.8, and train with 256 TPUV3s, it took a total of 3 days to train our models.

4.4.3 Downstream Tasks for Evaluation

We evaluate the performance of our pre-trained representations when tuned towards various downstream tasks, including a variety of video/audio and cross modalities tasks.

Video Action Recognition: Following [118], we evaluate on UCF101[153] (101 action classes, 13,320 videos) and the HMDB51[154] (51 classes, 6,766 videos) benchmarks. For both settings, we directly attach a linear classifier on the learned representations while freezing the transformer backbone. We also evaluate on Kinetics400[1][155] dataset (400 classes, 234,584 video), where we fine-tune the transformer backbone.

Audio Event Classification: Following [118], we used ESC50[156] (50 classes, 2000 audio clips) as the benchmark for audio modalities. Similarly, we directly attach and tune a linear classifier on top of the frozen transformer backbone.

Zero-shot Video-Text Retrieval: To evaluate our performance on cross-modality task, we follow [118] using YouCook2[1][127] (3.1K video-text clips) and MSR-VTT[1][157] (1K video-text clips) as our evaluation datasets. We directly use the embedding from the pre-trained model without any fine-tuning, and report the recall at 10 (R@10) and Median Rank as described in [118].

4.4.4 Result Analysis on HowTo100M and AudioSet Pre-training

Gradient Re-weighting We set up a simple gradient re-weighting baseline similar to [141], where we simply scale up the gradient magnitude of g_{va} by 2.5 times and scale down the gradient magnitude of its counterpart g_{vt} by 0.5 times, we denote this baseline as Gradient Re-weighting (VA), we also did it vice versa and denoted it as Gradient Re-weighting (VT). As shown in Table 4.1, we did not observe much performance boost with gradient re-weighting, on the contrary, there are significant performance drop in Text-Video Retrieval task on both YouCook2 and MSRVTT datasets.

Cross-Modality Gradient Realignment Compared to Gradient Re-weighting, our Cross-Modality Gradient Realignment mitigates the gradient conflicts by modifying gradients direction. In Table 4.1, we can see significant performance gains in video-text retrieval task (e.g. 58% gain in rank metric on YouCook2, 46% gain in R@10 metric on MSR-VTT), while maintaining a comparable performance in video/audio tasks. Since video-text retrieval task heavily hinge on cross-modality

[1]It is important to note that Howto100M, Youtube8M, AudioSet, Kinetics400, MSRVTT and YouCook2 videos are sourced from YouTube. Many of the videos have been made explicitly unavailable [158], hence we only train and evaluate over the subset of data that is publicly available at the time.

Dataset	Tasks	Video Action Cls						Text-Video Retrieval				Audio Cls	
	Dataset	UCF101		HMDB51		Kinetics400		YouCook2		MSRVTT		ESC50	
	Metric	Top1↑	Top5↑	Top1↑	Top5↑	Top1↑	Top5↑	Rank↓	R@10↑	Rank↓	R@10↑	Top1↑	Top5↑
HT100M	VATT [6]	78.53	95.24	57.36	86.07	74.71	92.69	93.50	17.10	73.00	16.70	71.50	91.75
	+ RW (VA)	78.24	95.01	**58.74**	85.39	70.55	90.41	168.90	9.24	100.20	13.56	71.56	93.02
	+ RW (VT)	78.03	95.35	58.23	**86.80**	75.66	92.80	90.35	19.26	62.50	18.02	71.29	91.38
	+ GR	78.44	95.37	54.38	83.64	**76.72**	92.72	47.00	24.94	68.00	19.20	69.00	93.00
	+ CL	**79.10**	**96.16**	56.41	86.06	**77.25**	**93.38**	**40.00**	**26.01**	**56.00**	**23.90**	**72.50**	**94.25**
	+ Both	**79.24**	**96.58**	**58.24**	**87.37**	76.59	**93.26**	**42.00**	**25.87**	**54.00**	**24.50**	**72.05**	**94.16**
HT100M + AudioSet	VATT [6]	84.40	-	63.10	-	79.23	94.30	34.00	29.00	**67.00**	**23.60**	81.20	-
	+ RW (VA)	83.44	97.28	59.55	88.02	76.56	93.52	238.50	6.80	147.00	12.30	81.75	97.25
	+ RW (VT)	84.42	97.49	62.30	86.61	78.59	94.17	76.00	18.72	120.00	15.20	80.75	96.75
	+ GR	84.77	97.38	62.30	**90.38**	**79.29**	94.32	**29.00**	**31.65**	70.00	**21.40**	81.50	**97.00**
	+ CL	**86.04**	**97.75**	**65.45**	88.94	**79.89**	**94.71**	33.00	29.17	**65.50**	20.97	**82.00**	**97.25**
	+ Both	**85.46**	**97.58**	**65.52**	**89.74**	79.26	**94.48**	**31.50**	**30.26**	69.50	19.96	**82.00**	**97.00**
HT100M + AudioSet + YT8M	VATT [6]	88.28	**98.73**	**65.84**	91.43	79.39	94.56	**29.00**	29.66	56.00	26.90	80.75	97.00
	+ RW (VA)	86.97	98.09	61.06	89.66	77.70	93.83	99.00	14.25	75.50	19.70	83.50	97.25
	+ RW (VT)	88.19	97.96	61.13	90.51	78.43	94.38	**27.00**	31.07	48.50	27.70	82.25	96.75
	+ GR	87.49	98.10	60.99	88.35	**79.73**	94.57	32.00	29.56	60.00	27.20	**85.00**	**98.00**
	+ CL	**89.02**	98.33	**65.77**	**92.15**	79.70	**94.80**	31.00	**31.34**	**48.50**	**28.70**	83.50	**97.75**
	+ Both	**89.70**	**98.35**	64.35	**92.08**	**80.01**	**94.69**	**29.00**	**31.86**	**45.00**	**29.10**	**84.50**	**98.00**

Table 4.1: Results of pre-training on HowTo100M, AudioSet and Youtube8M. *Best* results are highlighted in **blue**, *second best* results are highlighted in **light blue**. RW(VA): Gradient Re-weighting (Scale up Video-Audio), RW(VT): Gradient Re-weighting (Scale up Video-Text), GR: Cross-Modality Gradient Realignment, CL: Gradient-based Curriculum Learning (best view in color). Reprinted with permission from [5].

	Hyperparameter	Video Action Cls				Text-Video Retrieval				Audio Cls	
	Dataset	UCF101		HMDB51		YouCook2		MSRVTT		ESC50	
	Metric	Top1↑	Top5↑	Top1↑	Top5↑	Rank↓	R@10↑	Rank↓	R@10↑	Top1↑	Top5↑
VATT [6]		78.53	95.24	**57.36**	86.07	93.50	17.10	73.00	16.70	71.50	91.75
+ CL	$(\gamma_0, \gamma_{500K}) = (-0.3, 0.0)$	**79.10**	**96.16**	56.41	86.06	**40.00**	**26.01**	**56.00**	**23.90**	**72.50**	94.25
	$(\gamma_0, \gamma_{500K}) = (-0.2, 0.0)$	77.75	94.94	56.87	**87.18**	53.00	22.21	57.00	20.80	70.75	**94.75**
	$(\gamma_0, \gamma_{500K}) = (-0.1, 0.0)$	44.49	75.42	33.44	68.78	81.00	17.13	68.50	19.10	70.25	93.25
	$(\gamma_0, \gamma_{500K}) = (0.0, 0.0)$	10.05	24.88	7.31	23.76	457.50	5.02	153.50	11.20	58.50	87.50

Table 4.2: Ablation on the choice of γ in Gradient-based Curriculum Learning on Howto100M. We linearly decay γ during training, γ_0, γ_{500K} denote γ at the beginning and at the end of the training, respectively. *Best* results are highlighted in **blue**. CL: Gradient-based Curriculum Learning. Reprinted with permission from [5].

alignment (CMA), this improvement verify the benefits of re-aligning the cross-modality gradients.

Gradient-based Curriculum Learning Compared to Cross-Modality Gradient Realignment, Gradient-based Curriculum Learning inherents its performance gains in video-text retrieval task, yet enjoy an additional performance boost in audio classification (e.g. 1.40% gain ESC50 dataset in Table 4.1) and video classification (e.g. 1.94% gains UCF101 dataset in Table 4.1). We also perform an ablation on the adaptive choice of γ on Howto100M dataset in Table 4.2. We can see that curriculumly evolving γ (i.e.$(\gamma_0, \gamma_{500K}) = (-0.3, 0.0)$) yields the best overall results, while maintaining a fixed γ (i.e. $(\gamma_0, \gamma_{500K}) = (0.0, 0.0)$) yields the worst results, this verify the effectiveness of curriculumly removing mis-aligned samples during training.

Cross-Modality Gradient Realignment + Gradient-based Curriculum Learning We further verify the effectiveness with the combination of both techniques. Specifically, given a video-audio pair and video-text pair, (a) if $cos(g_{va}, g_{vt}) \leq \gamma$, we drop this sample triplet; (b) if $\gamma < cos(g_{va}, g_{vt}) < 0$, we apply gradient projection in Algorithm 1; (c) if $cos(g_{va}, g_{vt}) \geq 0$, we retain the original gradients. We found they are complimentary to each other and the combination of both lead to better performance in the majority of downstream tasks.

4.4.5 Scaling Pre-training to YouTube8M

As shown in Table 4.1, adding YouTube8M dataset brings the video classification and video-text retrieval performance to a new level (e.g. 88.28% Top1 on UCF101, 56.0 rank on MSRVTT)), yet it set back the performance in audio classification task, we conjecture this is largely due to the noisy misaligned samples introduced by Youtube8M dataset.

However, by leveraging our proposed techniques, we further yields a new state-of-the-art performance in the modality-agnostic setting, without any modification in architecture, striking 89.70% Top1 on UCF101, 92.15% Top5 on HMDB51, 80.01% Top1 on Kinetics400 and 85.00% Top1 on ESC50 dataset, demonstrating the effectiveness of our method when scaling to noisy Youtub8M dataset.

4.4.6 Feature Visualization

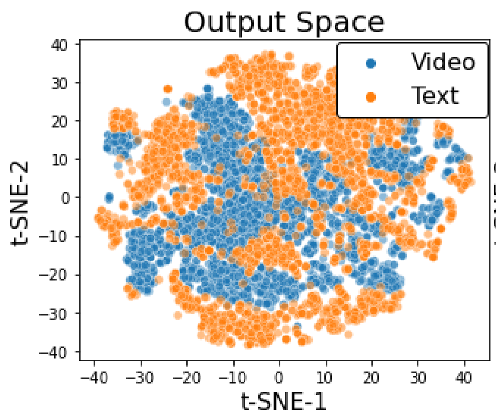
To better illustrate *why gradient harmonization techniques works*, in Figure 4.9, we follow the visualization used in VATT[6], which take YouCook2 Video-Text Retrieval dataset, showing the output space visualization of video and text, respectively. We can see the initial video and text feature representation of VATT baseline are mixed together (which aligned with the findings in [6]), however, after using gradient realignment, curriculum learning or applying both, the video and text representations become disentangled, this indicates our gradient harmonization techniques adds to cross-modality diversity, which makes video/text more discriminative from each other, making the model more discriminative and better in generalization.

4.4.7 Extending Gradient Realignment to Modality-Specific VATT

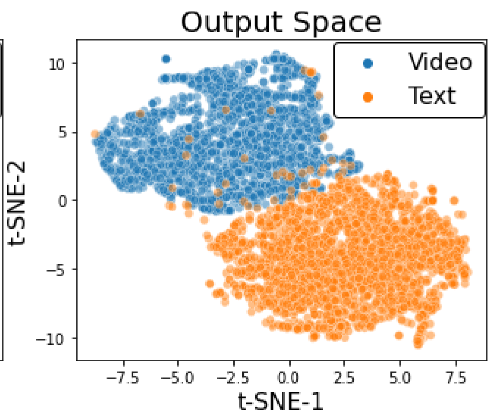
So far, our techniques were built upon the modality-agnostic setting of VATT with a shared backbone. We also try to apply the same technique to the modality-specific setting, given there is still a shared video head where we can harmonization the gradients. In Table 4.3, we observe that applying Gradient Realignment (GR) to modality-specific setting has some gain over uni-modality tasks (e.g. Video/Audio Classfication), yet set back on cross-modality tasks (e.g. Text-Video Retrieval). We hypothesize this is due the lack of cross-modality signals: since there is no longer shared backbone across modalities, the only shared video head would provide bias signala towards the video modality, for cross-modality alignment. We leave the improvement of modality-specific VATT for future work.

Methods	Video Action Cls				Text-Video Retrieval				Audio Cls	
Dataset	UCF101		HMDB51		YouCook2		MSRVTT		ESC50	
Metric	Top1↑	Top5↑	Top1↑	Top5↑	Rank↓	R@10↑	Rank↓	R@10↑	Top1↑	Top5↑
Modality-Specific VATT [6]	82.36	97.35	58.51	89.32	19.00	38.84	27.00	32.38	74.72	93.75
+ GR	82.06	97.35	61.45	90.45	22.00	36.82	40.00	27.32	76.50	93.00

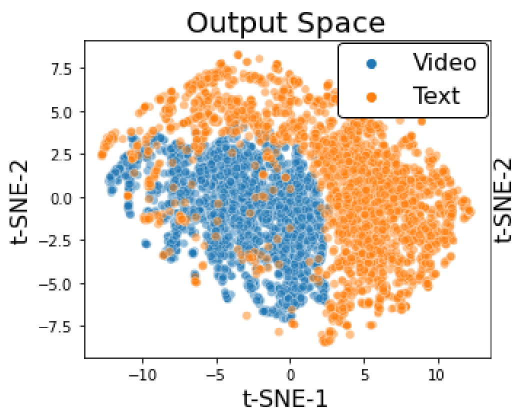
Table 4.3: Extended experiments on applying Gradient Realignment to the modality-specific VATT. Reprinted with permission from [5].



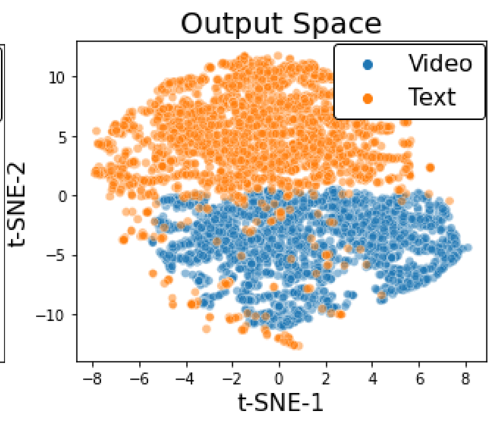
(a) VATT Baseline



(b) w/ Gradient Realignment



(c) w/ Curriculum Learning



(d) w/ Both Techniques

Figure 4.9: t-SNE visualization of the VATT output space on YouCook2 dataset. Reprinted with permission from [5].

4.5 Conclusion and Discussions

We take a deep dive into the common CMA assumption used in multimodal pre-training in the setting of Neural Architecture Unification. Using the gradient directions between pairwise losses as the surrogate indicator, we observe ubiquitous gradient conflicts even in relatively well-aligned narrative datasets. We then propose two plug-and-play techniques to harmonize the gradients during training. They are demonstrated to substantially enhance VATT pre-training across a number of downstream tasks, and further scale it up to pre-training with even more complicated and poorly aligned data. Our findings underline the (often overlooked) importance to carefully revisit the basic assumption and training dynamics of multimodal learning, besides advancing the model architectures. However, our cross-modality gradient harmonization technique still has room to generalize better to the modality-specific setting. Additionally, we did not fully explore other regularization techniques in multi-tasking learning literature, such as [137, 138, 139]. The potential negative societal impacts include the malicious use of multimodal models, such as unintended use of its downstream visual/audio recognition capability.

5. CONCLUSION

In this dissertation, we address Efficient Deep Learning from three distinct perspectives, namely (a) Neural Network Compression, (b) Neural Architecture Search, and (c) Neural Architecture Unification.

Firstly, we proposed a neural network compression algorithm called Deep k-Means for deep convolutional neural networks (CNNs). The algorithm involves applying k-means clustering on the weights of the CNN, which allows for weight-sharing and reduces the number of parameters needed. A novel regularization technique called spectrally relaxed k-means is introduced to make hard assignments of convolutional layer weights to learned cluster centers during re-training. The performance of Deep k-Means is evaluated against several state-of-the-art compression techniques and is shown to consistently achieve higher accuracy at the same compression ratio. Additionally, energy-aware metrics are developed to estimate energy consumption of CNN hardware implementations, and Deep k-Means is shown to have favorable energy efficiency as well.

Secondly, We proposed a novel framework for predictor-based neural architecture search (NAS) called WeakNAS. The approach utilizes a series of weak predictors to progressively shrink the search space towards the subspace where good architectures reside, rather than modeling the entire space with a strong predictor. By co-evolving the sampling stage and learning stage, WeakNAS can greatly simplify the learning task of each predictor and is both sample-efficient and robust to various combinations of predictors and architecture encoding means. Extensive experiments on popular NAS benchmarks demonstrate the effectiveness of WeakNAS in finding high-performing architectures while reducing computational costs.

Finally, we discuss the success of Neural Architecture Unification in self-supervised pre-training on large-scale multimodal data and how state-of-the-art contrastive learning methods enforce feature consistency from cross-modality inputs. However, cross-modality alignment can be weak and noisy supervision, which can negatively impact the performance of multimodal pre-training models. To address this issue, we proposes two plug-and-play techniques for cross-modality gradient harmonization during training. These techniques are demonstrated to substantially enhance VATT pre-training across a number of downstream tasks and further scale it up to pre-training with even more complicated and poorly aligned data. We underline the importance of carefully revisiting the basic assumption and training dynamics of multimodal learning, besides advancing the model architectures.

We strongly believe in the potential of Efficient Deep Learning for the future, and we believe that the three distinct perspectives mentioned can be further combined and integrated to achieve even greater efficiency and performance gains.

REFERENCES

[1] R. Abbasi-Asl and B. Yu, “Structural compression of convolutional neural networks based on greedy filter pruning,” *arXiv preprint arXiv:1705.07356*, 2017.

[2] J. Wu, Y. Wang, Z. Wu, Z. Wang, A. Veeraraghavan, and Y. Lin, “Deep k-means: Re-training and parameter sharing with harder cluster assignments for compressing deep convolutions,” in *International Conference on Machine Learning*, pp. 5363–5372, PMLR, 2018.

[3] J. Wu, X. Dai, D. Chen, Y. Chen, M. Liu, Y. Yu, Z. Wang, Z. Liu, M. Chen, and L. Yuan, “Stronger nas with weaker predictors,” *Advances in Neural Information Processing Systems*, vol. 34, pp. 28904–28918, 2021.

[4] S. Abu-El-Haija, N. Kothari, J. Lee, P. Natsev, G. Toderici, B. Varadarajan, and S. Vijayanarasimhan, “Youtube-8m: A large-scale video classification benchmark,” *arXiv preprint arXiv:1609.08675*, 2016.

[5] J. Wu, Y. Liang, H. Akbari, Z. Wang, C. Yu, *et al.*, “Scaling multimodal pre-training via cross-modality gradient harmonization,” *Advances in Neural Information Processing Systems*, vol. 35, pp. 36161–36173, 2022.

[6] H. Akbari, L. Yuan, R. Qian, W.-H. Chuang, S.-F. Chang, Y. Cui, and B. Gong, “Vatt: Transformers for multimodal self-supervised learning from raw video, audio and text,” *arXiv preprint arXiv:2104.11178*, 2021.

[7] T. Garipov, D. Podoprikhin, A. Novikov, and D. Vetrov, “Ultimate tensorization: compressing convolutional and fc layers alike,” *arXiv preprint arXiv:1611.03214*, 2016.

[8] W. Chen, J. Wilson, S. Tyree, K. Q. Weinberger, and Y. Chen, “Compressing convolutional neural networks in the frequency domain,” in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1475–1484, ACM, 2016.

[9] K. Ullrich, E. Meeds, and M. Welling, “Soft weight-sharing for neural network compression,” *arXiv preprint arXiv:1702.04008*, 2017.

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in Neural Information Processing Systems 25* (F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), pp. 1097–1105, Curran Associates, Inc., 2012.

[11] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” vol. abs/1311.2524, 2013.

[12] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, “Deepface: Closing the gap to human-level performance in face verification,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1701–1708, 2014.

[13] 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.

[14] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, “Exploring the limits of transfer learning with a unified text-to-text transformer,” *arXiv preprint arXiv:1910.10683*, 2019.

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

[16] C. Sun, A. Myers, C. Vondrick, K. Murphy, and C. Schmid, “Videobert: A joint model for video and language representation learning,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 7464–7473, 2019.

[17] H. Luo, L. Ji, B. Shi, H. Huang, N. Duan, T. Li, J. Li, T. Bharti, and M. Zhou, “Univl: A unified video and language pre-training model for multimodal understanding and generation,” *arXiv preprint arXiv:2002.06353*, 2020.

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

[19] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, “Edge computing: Vision and challenges,” vol. 3, pp. 637–646, Oct 2016.

[20] Y. Lin, C. Sakr, Y. Kim, and N. Shanbhag, “Predictivenet: An energy-efficient convolutional neural network via zero prediction,” in *Proceedings of ISCAS*, 2017.

[21] S. Han, H. Mao, and W. J. Dally, “Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding,” in *Proceedings of ICLR*, 2016.

[22] T.-J. Yang, Y.-H. Chen, and V. Sze, “Designing energy-efficient convolutional neural networks using energy-aware pruning,” *arXiv preprint*, 2017.

[23] S. Bhattacharya and N. D. Lane, “Sparsification and separation of deep learning layers for constrained resource inference on wearables,” in *Proceedings of SenSys*, 2016.

[24] N. D. Lane, S. Bhattacharya, P. Georgiev, C. Forlivesi, L. Jiao, L. Qendro, and F. Kawsar, “Deepx: A software accelerator for low-power deep learning inference on mobile devices,” in *Proceedings of IPSN*, 2016.

[25] S. Changpinyo, M. Sandler, and A. Zhmoginov, “The power of sparsity in convolutional neural networks,” *arXiv preprint arXiv:1702.06257*, 2017.

[26] 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.

[27] Z. Wang, J. Yang, H. Jin, E. Shechtman, A. Agarwala, J. Brandt, and T. S. Huang, “Deepfont: Identify your font from an image,” in *Proceedings of the 23rd ACM international conference on Multimedia*, pp. 451–459, ACM, 2015.

[28] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size," *arXiv preprint arXiv:1602.07360*, 2016.

[29] M. Lin, Q. Chen, and S. Yan, "Network in network," in *Proceedings of ICLR*, 2014.

[30] Y. H. Chen, J. Emer, and V. Sze, "Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks," in *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, pp. 367–379, June 2016.

[31] C. Sakr, Y. Kim, and N. Shanbhag, "Analytical guarantees on numerical precision of deep neural networks," in *Proceedings of the 34th International Conference on Machine Learning* (D. Precup and Y. W. Teh, eds.), vol. 70 of *Proceedings of Machine Learning Research*, (International Convention Centre, Sydney, Australia), pp. 3007–3016, PMLR, 06–11 Aug 2017.

[32] S. J. Hanson and L. Y. Pratt, "Comparing biases for minimal network construction with back-propagation," in *Advances in Neural Information Processing Systems 1* (D. S. Touretzky, ed.), pp. 177–185, Morgan-Kaufmann, 1989.

[33] Y. L. Cun, J. S. Denker, and S. A. Solla, "Advances in neural information processing systems 2," ch. Optimal Brain Damage, pp. 598–605, 1990.

[34] B. Hassibi and D. G. Stork, "Second order derivatives for network pruning: Optimal brain surgeon," in *Advances in Neural Information Processing Systems 5* (S. J. Hanson, J. D. Cowan, and C. L. Giles, eds.), pp. 164–171, Morgan-Kaufmann, 1993.

[35] S. Srinivas and R. V. Babu, "Data-free parameter pruning for deep neural networks," *CoRR*, vol. abs/1507.06149, 2015.

[36] Y. Gong, L. Liu, M. Yang, and L. D. Bourdev, "Compressing deep convolutional networks using vector quantization," *CoRR*, vol. abs/1412.6115, 2014.

[37] W. Chen, J. T. Wilson, S. Tyree, K. Q. Weinberger, and Y. Chen, "Compressing neural networks with the hashing trick," *CoRR*, vol. abs/1504.04788, 2015.

[38] T. He, Y. Fan, Y. Qian, T. Tan, and K. Yu, “Reshaping deep neural network for fast decoding by node-pruning,” in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 245–249, May 2014.

[39] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, “Pruning filters for efficient convnets,” *CoRR*, vol. abs/1608.08710, 2016.

[40] Y. Gong, L. Liu, M. Yang, and L. Bourdev, “Compressing deep convolutional networks using vector quantization,” *arXiv preprint arXiv:1412.6115*, 2014.

[41] H. Zha, X. He, C. Ding, M. Gu, and H. D. Simon, “Spectral relaxation for k-means clustering,” in *Advances in neural information processing systems*, pp. 1057–1064, 2002.

[42] Y. Lin, S. Zhang, and N. R. Shanbhag, “Variation-tolerant architectures for convolutional neural networks in the near threshold voltage regime,” in *2016 IEEE International Workshop on Signal Processing Systems (SiPS)*, pp. 17–22, Oct 2016.

[43] S. Han, J. Pool, J. Tran, and W. J. Dally, “Learning both weights and connections for efficient neural networks,” *CoRR*, vol. abs/1506.02626, 2015.

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

[45] A. Krizhevsky and G. Hinton, “Learning multiple layers of features from tiny images,” 2009.

[46] Y. Cheng, D. Wang, P. Zhou, and T. Zhang, “A survey of model compression and acceleration for deep neural networks,” *arXiv preprint arXiv:1710.09282*, 2017.

[47] M. Denil, B. Shakibi, L. Dinh, N. De Freitas, *et al.*, “Predicting parameters in deep learning,” in *Advances in neural information processing systems*, pp. 2148–2156, 2013.

[48] A. Parashar, M. Rhu, A. Mukkara, A. Puglielli, R. Venkatesan, B. Khailany, J. Emer, S. W. Keckler, and W. J. Dally, “Scnn: An accelerator for compressed-sparse convolutional neural networks,” in *Proceedings of the 44th Annual International Symposium on Computer Architecture*, pp. 27–40, ACM, 2017.

[49] S. Basu, A. Banerjee, and R. Mooney, “Semi-supervised clustering by seeding,” in *In Proceedings of 19th International Conference on Machine Learning (ICML-2002*, Citeseer, 2002.

[50] 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*, pp. 1–9, 2015.

[51] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, *et al.*, “Imagenet large scale visual recognition challenge,” *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[52] Y.-D. Kim, E. Park, S. Yoo, T. Choi, L. Yang, and D. Shin, “Compression of deep convolutional neural networks for fast and low power mobile applications,” *arXiv preprint arXiv:1511.06530*, 2015.

[53] C. Tai, T. Xiao, Y. Zhang, X. Wang, *et al.*, “Convolutional neural networks with low-rank regularization,” *arXiv preprint arXiv:1511.06067*, 2015.

[54] H. Liu, K. Simonyan, and Y. Yang, “Darts: Differentiable architecture search,” *arXiv preprint arXiv:1806.09055*, 2018.

[55] R. Luo, F. Tian, T. Qin, E. Chen, and T.-Y. Liu, “Neural architecture optimization,” in *Advances in neural information processing systems*, pp. 7816–7827, 2018.

[56] 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 Conference on Computer Vision and Pattern Recognition*, pp. 10734–10742, 2019.

[57] 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 International Conference on Computer Vision*, 2019.

[58] X. Ning, Y. Zheng, T. Zhao, Y. Wang, and H. Yang, "A generic graph-based neural architecture encoding scheme for predictor-based nas," *arXiv preprint arXiv:2004.01899*, 2020.

[59] C. Wei, C. Niu, Y. Tang, and J. Liang, "Npenas: Neural predictor guided evolution for neural architecture search," *arXiv preprint arXiv:2003.12857*, 2020.

[60] W. Wen, H. Liu, H. Li, Y. Chen, G. Bender, and P.-J. Kindermans, "Neural predictor for neural architecture search," *arXiv preprint arXiv:1912.00848*, 2019.

[61] L. Dudziak, T. Chau, M. Abdelfattah, R. Lee, H. Kim, and N. Lane, "Brp-nas: Prediction-based nas using gcns," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[62] R. Luo, X. Tan, R. Wang, T. Qin, E. Chen, and T.-Y. Liu, "Neural architecture search with gbdt," *arXiv preprint arXiv:2007.04785*, 2020.

[63] Y. Wang, Y. Xu, and D. Tao, "Dc-nas: Divide-and-conquer neural architecture search," *arXiv preprint arXiv:2005.14456*, 2020.

[64] X. Dai, D. Chen, M. Liu, Y. Chen, and L. Yuan, "Da-nas: Data adapted pruning for efficient neural architecture search," *ECCV 2020*, 2020.

[65] Z. Yang, Y. Wang, X. Chen, J. Guo, W. Zhang, C. Xu, C. Xu, D. Tao, and C. Xu, "Hournas: Extremely fast neural architecture search through an hourglass lens," *arXiv preprint arXiv:2005.14446*, 2020.

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

[67] 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*, vol. 33, pp. 4780–4789, 2019.

[68] Z. Yang, Y. Wang, X. Chen, B. Shi, C. Xu, C. Xu, Q. Tian, and C. Xu, "Cars: Continuous evolution for efficient neural architecture search," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1829–1838, 2020.

[69] W. Hong, G. Li, W. Zhang, R. Tang, Y. Wang, Z. Li, and Y. Yu, "Dropnas: Grouped operation dropout for differentiable architecture search," in *International Joint Conference on Artificial Intelligence*, 2020.

[70] Y. Xu, Y. Wang, K. Han, S. Jui, C. Xu, Q. Tian, and C. Xu, "Renas: Relativistic evaluation of neural architecture search," *arXiv preprint arXiv:1910.01523*, 2019.

[71] Y. Li, M. Dong, Y. Wang, and C. Xu, "Neural architecture search in a proxy validation loss landscape," in *International Conference on Machine Learning*, pp. 5853–5862, PMLR, 2020.

[72] H. Shi, R. Pi, H. Xu, Z. Li, J. Kwok, and T. Zhang, "Bridging the gap between sample-based and one-shot neural architecture search with bonas," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[73] R. Luo, X. Tan, R. Wang, T. Qin, E. Chen, and T.-Y. Liu, "Semi-supervised neural architecture search," *arXiv preprint arXiv:2002.10389*, 2020.

[74] L. Wang, S. Xie, T. Li, R. Fonseca, and Y. Tian, "Sample-efficient neural architecture search by learning actions for monte carlo tree search," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[75] R. Tappenden, P. Richtárik, and J. Gondzio, "Inexact coordinate descent: complexity and preconditioning," *Journal of Optimization Theory and Applications*, vol. 170, no. 1, pp. 144–176, 2016.

[76] M. Schmidt, N. L. Roux, and F. Bach, "Convergence rates of inexact proximal-gradient methods for convex optimization," *arXiv preprint arXiv:1109.2415*, 2011.

[77] W. W. Hager and H. Zhang, "Convergence rates for an inexact admm applied to separable convex optimization," *Computational Optimization and Applications*, 2020.

[78] Z.-H. Zhou, *Ensemble methods: foundations and algorithms*. CRC press, 2012.

[79] I. Radosavovic, R. P. Kosaraju, R. Girshick, K. He, and P. Dollár, "Designing network design spaces," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10428–10436, 2020.

[80] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[81] J. Siems, L. Zimmer, A. Zela, J. Lukasik, M. Keuper, and F. Hutter, "Nas-bench-301 and the case for surrogate benchmarks for neural architecture search," *arXiv preprint arXiv:2008.09777*, 2020.

[82] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785–794, 2016.

[83] 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," *arXiv preprint arXiv:2004.13431*, 2020.

[84] X. Dong and Y. Yang, "Nas-bench-201: Extending the scope of reproducible neural architecture search," in *International Conference on Learning Representations*, 2020.

[85] 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*, pp. 7105–7114, 2019.

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

[87] 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*, pp. 8697–8710, 2018.

[88] H. Cai, C. Gan, T. Wang, Z. Zhang, and S. Han, "Once-for-all: Train one network and specialize it for efficient deployment," *arXiv preprint arXiv:1908.09791*, 2019.

[89] 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*, pp. 248–255, Ieee, 2009.

[90] 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," *arXiv preprint arXiv:1904.00420*, 2019.

[91] R. Wightman, "Pytorch image models." `https://github.com/rwightman/pytorch-image-models`, 2019.

[92] R. Luo, X. Tan, R. Wang, T. Qin, E. Chen, and T.-Y. Liu, "Accuracy prediction with non-neural model for neural architecture search," *arXiv preprint arXiv:2007.04785*, 2020.

[93] X. Dai, A. Wan, P. Zhang, B. Wu, Z. He, Z. Wei, K. Chen, Y. Tian, M. Yu, P. Vajda, *et al.*, "Fbnetv3: Joint architecture-recipe search using neural acquisition function," *arXiv preprint arXiv:2006.02049*, 2020.

[94] S. Yan, Y. Zheng, W. Ao, X. Zeng, and M. Zhang, "Does unsupervised architecture representation learning help neural architecture search?," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[95] S. Yan, K. Song, F. Liu, and M. Zhang, "Cate: Computation-aware neural architecture encoding with transformers," *arXiv preprint arXiv:2102.07108*, 2021.

[96] L. Wang, Y. Zhao, Y. Jinnai, Y. Tian, and R. Fonseca, "Alphax: exploring neural architectures with deep neural networks and monte carlo tree search," *arXiv preprint arXiv:1903.11059*, 2019.

[97] Y. Tang, Y. Wang, Y. Xu, H. Chen, B. Shi, C. Xu, C. Xu, Q. Tian, and C. Xu, "A semi-supervised assessor of neural architectures," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1810–1819, 2020.

[98] C. White, W. Neiswanger, and Y. Savani, "Bananas: Bayesian optimization with neural architectures for neural architecture search," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, pp. 10293–10301, 2021.

[99] B. Ru, X. Wan, X. Dong, and M. Osborne, “Interpretable neural architecture search via bayesian optimisation with weisfeiler-lehman kernels,” in *International Conference on Learning Representations*, 2021.

[100] 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)*, pp. 19–34, 2018.

[101] 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*, pp. 702–717, Springer, 2020.

[102] B. Lakshminarayanan, A. Pritzel, and C. Blundell, “Simple and scalable predictive uncertainty estimation using deep ensembles,” *arXiv preprint arXiv:1612.01474*, 2016.

[103] D. R. Jones, M. Schonlau, and W. J. Welch, “Efficient global optimization of expensive black-box functions,” *Journal of Global optimization*, vol. 13, no. 4, pp. 455–492, 1998.

[104] S. Xie, H. Zheng, C. Liu, and L. Lin, “Snas: stochastic neural architecture search,” *arXiv preprint arXiv:1812.09926*, 2018.

[105] 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 International Conference on Computer Vision*, pp. 1294–1303, 2019.

[106] Y. Xu, L. Xie, X. Zhang, X. Chen, G.-J. Qi, Q. Tian, and H. Xiong, “Pc-darts: Partial channel connections for memory-efficient differentiable architecture search,” *arXiv preprint arXiv:1907.05737*, 2019.

[107] H. Cai, L. Zhu, and S. Han, “Proxylessnas: Direct neural architecture search on target task and hardware,” *arXiv preprint arXiv:1812.00332*, 2018.

[108] S. Ji, W. Xu, M. Yang, and K. Yu, “3d convolutional neural networks for human action recognition,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 1, pp. 221–231, 2012.

[109] 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*, pp. 4489–4497, 2015.

[110] 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)*, pp. 305–321, 2018.

[111] S. Hershey, S. Chaudhuri, D. P. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, *et al.*, "Cnn architectures for large-scale audio classification," in *2017 ieee international conference on acoustics, speech and signal processing (icassp)*, pp. 131–135, IEEE, 2017.

[112] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[113] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, pp. 5998–6008, 2017.

[114] 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," *arXiv preprint arXiv:2010.11929*, 2020.

[115] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou, "Training data-efficient image transformers & distillation through attention," in *International Conference on Machine Learning*, pp. 10347–10357, PMLR, 2021.

[116] A. Arnab, M. Dehghani, G. Heigold, C. Sun, M. Lučić, and C. Schmid, "Vivit: A video vision transformer," *arXiv preprint arXiv:2103.15691*, 2021.

[117] G. Bertasius, H. Wang, and L. Torresani, "Is space-time attention all you need for video understanding," *arXiv preprint arXiv:2102.05095*, vol. 2, no. 3, p. 4, 2021.

[118] J.-B. Alayrac, A. Recasens, R. Schneider, R. Arandjelovic, J. Ramapuram, J. De Fauw, L. Smaira, S. Dieleman, and A. Zisserman, "Self-supervised multimodal versatile networks.," *NeurIPS*, vol. 2, no. 6, p. 7, 2020.

[119] R. Arandjelovic and A. Zisserman, "Look, listen and learn," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 609–617, 2017.

[120] R. Arandjelovic and A. Zisserman, "Objects that sound," in *Proceedings of the European conference on computer vision (ECCV)*, pp. 435–451, 2018.

[121] B. Korbar, D. Tran, and L. Torresani, "Cooperative learning of audio and video models from self-supervised synchronization," *arXiv preprint arXiv:1807.00230*, 2018.

[122] 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*, pp. 1450–1459, 2021.

[123] P. Morgado, N. Vasconcelos, and I. Misra, "Audio-visual instance discrimination with cross-modal agreement," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12475–12486, 2021.

[124] A. Miech, D. Zhukov, J.-B. Alayrac, M. Tapaswi, I. Laptev, and J. Sivic, "Howto100m: Learning a text-video embedding by watching hundred million narrated video clips," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 2630–2640, 2019.

[125] A. Miech, J.-B. Alayrac, L. Smaira, I. Laptev, J. Sivic, and A. Zisserman, "End-to-end learning of visual representations from uncurated instructional videos," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9879–9889, 2020.

[126] J. Wang, H. Wang, J. Deng, W. Wu, and D. Zhang, "Efficientclip: Efficient cross-modal pre-training by ensemble confident learning and language modeling," *arXiv preprint arXiv:2109.04699*, 2021.

[127] L. Zhou, C. Xu, and J. J. Corso, “Towards automatic learning of procedures from web instructional videos,” in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[128] Y. Huang, J. Lin, C. Zhou, H. Yang, and L. Huang, “Modality competition: What makes joint training of multi-modal network fail in deep learning?(provably),” *arXiv preprint arXiv:2203.12221*, 2022.

[129] T. Yu, S. Kumar, A. Gupta, S. Levine, K. Hausman, and C. Finn, “Gradient surgery for multi-task learning,” *arXiv preprint arXiv:2001.06782*, 2020.

[130] H. Alwassel, D. Mahajan, B. Korbar, L. Torresani, B. Ghanem, and D. Tran, “Self-supervised learning by cross-modal audio-video clustering,” *arXiv preprint arXiv:1911.12667*, 2019.

[131] A. Recasens, P. Luc, J.-B. Alayrac, L. Wang, F. Strub, C. Tallec, M. Malinowski, V. Patraucean, F. Altché, M. Valko, *et al.*, “Broaden your views for self-supervised video learning,” *arXiv preprint arXiv:2103.16559*, 2021.

[132] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, “A simple framework for contrastive learning of visual representations,” in *International conference on machine learning*, pp. 1597–1607, PMLR, 2020.

[133] X. Chen, H. Fan, R. Girshick, and K. He, “Improved baselines with momentum contrastive learning,” *arXiv preprint arXiv:2003.04297*, 2020.

[134] J. F. Gemmeke, D. P. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, “Audio set: An ontology and human-labeled dataset for audio events,” in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 776–780, IEEE, 2017.

[135] Y. Zhang and Q. Yang, “A survey on multi-task learning,” *arXiv preprint arXiv:1707.08114*, 2017.

[136] C. Fifty, E. Amid, Z. Zhao, T. Yu, R. Anil, and C. Finn, “Efficiently identifying task groupings for multi-task learning,” *arXiv preprint arXiv:2109.04617*, 2021.

[137] Z. Chen, V. Badrinarayanan, C.-Y. Lee, and A. Rabinovich, “Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks,” in *International Conference on Machine Learning*, pp. 794–803, PMLR, 2018.

[138] O. Sener and V. Koltun, “Multi-task learning as multi-objective optimization,” *arXiv preprint arXiv:1810.04650*, 2018.

[139] A. Kendall, Y. Gal, and R. Cipolla, “Multi-task learning using uncertainty to weigh losses for scene geometry and semantics,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7482–7491, 2018.

[140] Z. Wang, Y. Tsvetkov, O. Firat, and Y. Cao, “Gradient vaccine: Investigating and improving multi-task optimization in massively multilingual models,” *arXiv preprint arXiv:2010.05874*, 2020.

[141] W. Wang, D. Tran, and M. Feiszli, “What makes training multi-modal classification networks hard?,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12695–12705, 2020.

[142] M. Gutmann and A. Hyvärinen, “Noise-contrastive estimation: A new estimation principle for unnormalized statistical models,” in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 297–304, JMLR Workshop and Conference Proceedings, 2010.

[143] S. Chatterjee, “Coherent gradients: An approach to understanding generalization in gradient descent-based optimization,” *arXiv preprint arXiv:2002.10657*, 2020.

[144] B. Han, Q. Yao, X. Yu, G. Niu, M. Xu, W. Hu, I. W. Tsang, and M. Sugiyama, “Co-teaching: robust training of deep neural networks with extremely noisy labels,” in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pp. 8536–8546, 2018.

[145] L. Jiang, Z. Zhou, T. Leung, L.-J. Li, and L. Fei-Fei, “Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels,” in *International Conference on Machine Learning*, pp. 2304–2313, PMLR, 2018.

[146] T. Zhou, S. Wang, and J. Bilmes, “Robust curriculum learning: From clean label detection to noisy label self-correction,” in *International Conference on Learning Representations*, 2021.

[147] A. H. Jiang, D. L.-K. Wong, G. Zhou, D. G. Andersen, J. Dean, G. R. Ganger, G. Joshi, M. Kaminksy, M. Kozuch, Z. C. Lipton, *et al.*, “Accelerating deep learning by focusing on the biggest losers,” *arXiv preprint arXiv:1910.00762*, 2019.

[148] L. Li, Y.-C. Chen, Y. Cheng, Z. Gan, L. Yu, and J. Liu, “Hero: Hierarchical encoder for video+ language omni-representation pre-training,” *arXiv preprint arXiv:2005.00200*, 2020.

[149] A. Jaegle, S. Borgeaud, J.-B. Alayrac, C. Doersch, C. Ionescu, D. Ding, S. Koppula, D. Zoran, A. Brock, E. Shelhamer, *et al.*, “Perceiver io: A general architecture for structured inputs & outputs,” *arXiv preprint arXiv:2107.14795*, 2021.

[150] A. Rouditchenko, A. Boggust, D. Harwath, B. Chen, D. Joshi, S. Thomas, K. Audhkhasi, H. Kuehne, R. Panda, R. Feris, *et al.*, “Avlnet: Learning audio-visual language representations from instructional videos,” *arXiv preprint arXiv:2006.09199*, 2020.

[151] H. Xu, G. Ghosh, P.-Y. Huang, P. Arora, M. Aminzadeh, C. Feichtenhofer, F. Metze, and L. Zettlemoyer, “Vlm: Task-agnostic video-language model pre-training for video understanding,” *arXiv preprint arXiv:2105.09996*, 2021.

[152] V. Likhosherstov, A. Arnab, K. Choromanski, M. Lucic, Y. Tay, A. Weller, and M. Dehghani, “Polyvit: Co-training vision transformers on images, videos and audio,” *arXiv preprint arXiv:2111.12993*, 2021.

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

[154] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre, "Hmdb: a large video database for human motion recognition," in *2011 International conference on computer vision*, pp. 2556–2563, IEEE, 2011.

[155] 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.

[156] K. J. Piczak, "Esc: Dataset for environmental sound classification," in *Proceedings of the 23rd ACM international conference on Multimedia*, pp. 1015–1018, 2015.

[157] J. Xu, T. Mei, T. Yao, and Y. Rui, "Msr-vtt: A large video description dataset for bridging video and language," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5288–5296, 2016.

[158] L. Smaira, J. Carreira, E. Noland, E. Clancy, A. Wu, and A. Zisserman, "A short note on the kinetics-700-2020 human action dataset," *arXiv preprint arXiv:2010.10864*, 2020.