AutoFormer: Searching Transformers for Visual Recognition

Minghao Chen[1,*], Houwen Peng[2,*,†], Jianlong Fu[2], Haibin Ling[1]
[1]Stony Brook University [2]Microsoft Research Asia

Abstract

Recently, pure transformer-based models have shown great potentials for vision tasks such as image classification and detection. However, the design of transformer networks is challenging. It has been observed that the depth, embedding dimension, and number of heads can largely affect the performance of vision transformers. Previous models configure these dimensions based upon manual crafting. In this work, we propose a new one-shot architecture search framework, namely AutoFormer, *dedicated to vision transformer search. AutoFormer entangles the weights of different blocks in the same layers during supernet training. Benefiting from the strategy, the trained supernet allows thousands of subnets to be very well-trained. Specifically, the performance of these subnets with weights inherited from the supernet is comparable to those retrained from scratch. Besides, the searched models, which we refer to AutoFormers, surpass the recent state-of-the-arts such as ViT and DeiT. In particular, AutoFormer-tiny/small/base achieve 74.7%/81.7%/82.4% top-1 accuracy on ImageNet with 5.7M/22.9M/53.7M parameters, respectively. Lastly, we verify the transferability of AutoFormer by providing the performance on downstream benchmarks and distillation experiments. Code and models are available at https://github.com/microsoft/Cream.*

1. Introduction

Vision transformer recently has drawn significant attention in computer vision because of its high model capability and superior potentials in capturing long-range dependencies. Building on top of transformers [52], modern state-of-the-art models, such as ViT [13] and DeiT [50], are able to learn powerful visual representations from images and achieve very competitive performance compared to previous convolutional neural network models [17, 25].

However, the design of transformer neural architectures is nontrivial. For example, how to choose the best network depth, embedding dimension and/or head number? These factors are all critical for elevating model capacity, yet finding a good combination of them is difficult. As seen in Fig. 2, increasing the depth, head number and MLP ratio (the ratio of hidden dimension to the embedding dimension in the multi-layer perceptron) of transformers helps achieve higher accuracy at first but get overfit after hitting the peak value. Scaling up the embedding dimension can improve model capability, but the accuracy gain plateaus for larger models. These phenomenons demonstrate the challenge of designing optimal transformer architectures.

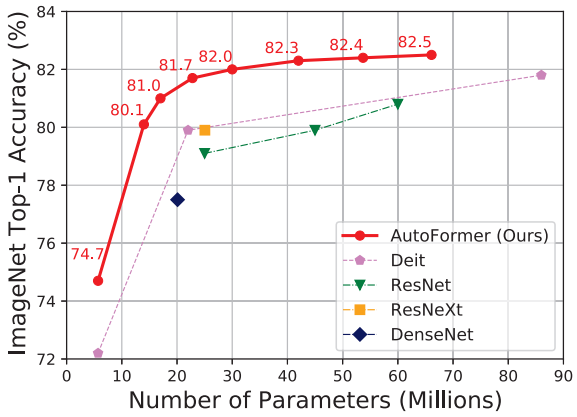

Figure 1. The comparison between AutoFormers and transformer-based, convolution-based and architecture-searched models, such as DeiT [50], and ResNet [18].

Previous works on designing vision transformers are based upon manual crafting, which heavily relies on human expertise and typically requires a deal of trial-and-error [13, 50, 67]. There are a few works on automating transformer design using neural architecture search (NAS) [45, 55]. However, they are all concentrated on natural language tasks, such as machine translation, which are quite different from computer vision tasks. As a result, it is hard to generalize prior automatic search algorithms to find effective vision transformer architectures.

In this work, we present a new architecture search algorithm, named *AutoFormer*, dedicated to finding pure vision transformer models. Our approach mainly addresses two challenges in transformer search. 1) How to strike a good

*Equal contributions. Work performed when Minghao is an intern of MSRA. † Corresponding author: houwen.peng@microsoft.com.

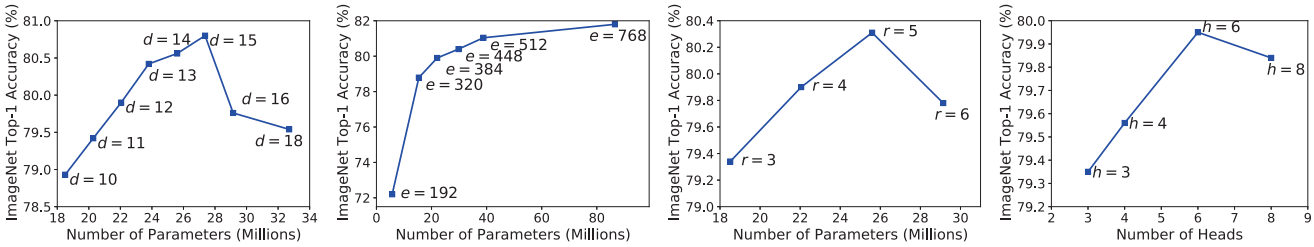

Figure 2. Adjust a baseline model with different embedding dimension (e), depth (d), MLP ratio (r), and number of heads (h) coefficients under the same training recipe, where MLP ratio(the ratio of hidden dimension to the embedding dimension in the multi-layer perceptron). We set the baseline model with $d = 12, r = 4, e = 384, h = 6$. **Note:** number of heads does not affect the model size and complexity if we fix the Q-K-V dimension.

combination of the key factors in transformers, such as network depth, embedding dimension and head number? 2) How to efficiently find out various transformer models that fit different resource constraints and application scenarios?

To tackle the challenges, we construct a large search space covering the main changeable dimensions of transformer, including embedding dimension, number of heads, query/key/value dimension, MLP ratio, and network depth. This space contains a vast number of transformers with diverse structures and model complexities. In particular, it allows the construction of transformers to use different structures of building blocks, thus breaking the convention that all blocks share an identical structure in transformer design.

To address the efficiency issue, inspired by BigNAS [65] and slimmable networks [65, 66], we propose a supernet training strategy called *weight entanglement* dedicated to transformer architecture. The central idea is to enable different transformer blocks to share weights for their common parts in each layer. An update of weights in one block will affect all other ones as a whole, such that the weights of different blocks are maximally entangled during training. This strategy is different from most one-shot NAS methods [16, 8, 59], in which the weights of different blocks are independent for the same layer, as visualized in Fig. 5.

We observe a surprising phenomenon when using the proposed weight entanglement for transformer supernet training: *it allows a large number of subnets in the supernet to be very well-trained, such that the performance of these subnets with weights inherited from the supernet are comparable to those retrained from scratch.* This advantage allows our method to obtain thousands of architectures that can meet different resource constraints while maintaining the same level of accuracy as training from scratch independently. We give a detailed discussion in Section 3.4 exploring the underlying reasons of weight entanglement.

We perform a evolutionary search with a model size constraint over the well-trained supernets to find promising transformers. Experiments on ImageNet [11] demonstrate that our method achieves superior performance to the handcrafted state-of-the-art transformer models. For instance, as shown in Fig. 1, with 22.9M parameters, Autoformer achieves a top-1 accuracy of 81.7%, being 1.8% and 2.9% better than DeiT-S [50] and ViT-S/16 [13], respectively. In addition, when transferred to downstream vision classification datasets, our AutoFormer also performs well with fewer parameters, achieving better or comparable results to the best convolutional models, such as EfficientNet [49].

In summary, we make three major contributions in this paper. 1) To our best knowledge, this work is the first effort to design an automatic search algorithm for finding vision transformer models. 2) We propose a simple yet effective framework for efficient training of transformer supernets. Without extra finetuning or retraining, the trained supernet is able to produce thousands of high quality transformers by inheriting weights from it directly. Such merit allows our method to search diverse models to fit different resource constraints. 3) Our searched models, *i.e.*, AutoFormers, achieve the state-of-the-art results on ImageNet among the vision transformers, and demonstrate promising transferability on downstream tasks.

2. Background

Before presenting our method, we first briefly review the background of the vision transformer and one-shot NAS.

2.1. Vision Transformer

Transformer is originally designed for natural language tasks [52, 28, 12]. Recent works, such as ViT and DeiT [13, 50], show its great potential for visual recognition. In the following, we give the basic pipeline of the vision transformer, which serves as a base architecture of AutoFormer.

Given a 2D image, we first uniformly split it into a sequence of 2D patches just like tokens in natural language processing. We then flatten and transform the patches to D-dimension vectors, named patch embeddings, by either linear projection [13] or several CNN layers [67]. A learnable [class] embedding is injected into the head of the sequence to represent the whole picture. Position embeddings are added to the patch embeddings to retain positional information. The combined embeddings are then fed to a *transformer encoder* described below. At last, a linear layer is

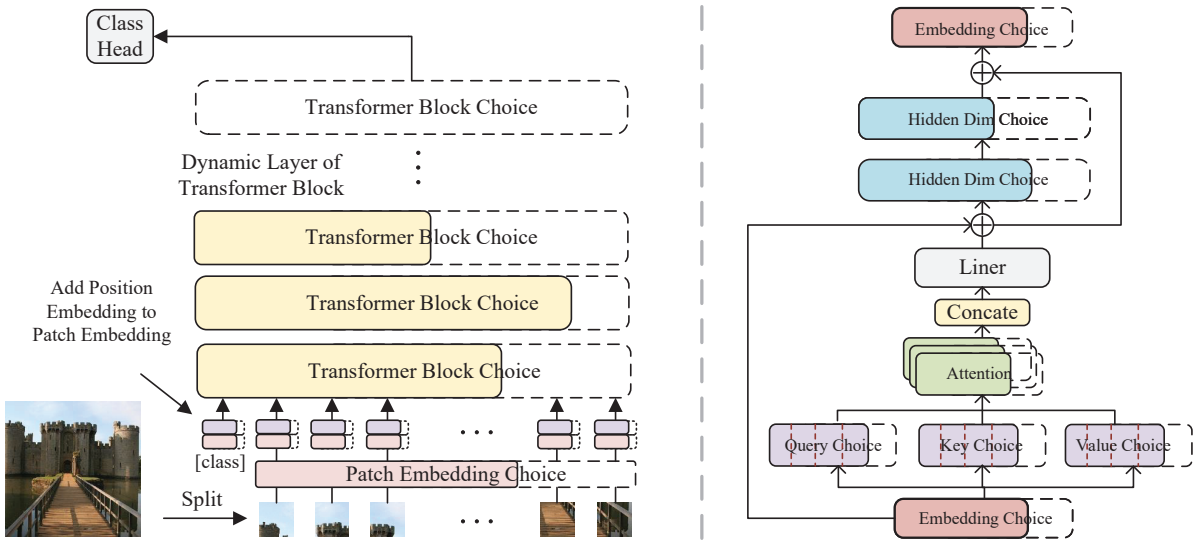

Figure 3. **Left:** The overall architecture of the AutoFormer supernet. Note that transformer blocks in each layer and depth are dynamic. The parts in solid lines mean they are chosen while those in dashed lines are not. **Right:** The detailed transformer block in an AutoFormer. We search for the best block of optimal embedding dimension, number of heads, MLP ratio, Q-K-V dim in a layer. For more details about the search space, please refer to section 3.2.

used for the final classification.

A transformer encoder consists of alternating blocks of *multihead self-attention* (MSA) and *multi-layer perceptron* (MLP) blocks. LayerNorm (LN) [2] is applied before every block, and residual connections after every block. The details of MSA and MLP are given below.

Multihead Self-Attention (MSA). In a standard self-attention module, the input sequence $z \in \mathbb{R}^{N\times D}$ will be first linearly transformed to queries $Q \in \mathbb{R}^{N\times D_h}$, keys $K \in \mathbb{R}^{N\times D_h}$ and values $V \in \mathbb{R}^{N\times D_h}$, where N is the number of tokens, D is the embedding dimension, D_h is the Q-K-V dimension. Then we compute the weighted sum over all values for each element in the sequence. The weights or attention are based on the pairwise similarity between two elements of the sequence:

$$\text{Attention}(Q, K, V) = \text{softmax}\Big(\frac{QK^T}{\sqrt{d_h}}\Big)V, \qquad (1)$$

where $\frac{1}{\sqrt{d_h}}$ is the scaling factor. Lastly, a fully connected layer is applied. Multihead self-attention splits the queries, keys and values into different heads and performs self-attention in parallel and projects their concatenated outputs.

Multi-Layer Perceptron (MLP). The MLP block consists two fully connected layers with an activation function, usually GELU [19]. In this work, we focus on finding optimal choices of the MLP ratios in each layer.

2.2. One-Shot NAS

One-shot NAS typically adopts a weight sharing strategy to avoid training each subnet from scratch [16, 39]. The architecture search space $\mathcal{A}$ is encoded in a supernet, denoted as $\mathcal{N}(\mathcal{A}, W)$, where W is the weight of the supernet. W is shared across all the architecture candidates, *i.e.*, subnets $\alpha \in \mathcal{A}$ in $\mathcal{N}$. The search of the optimal architecture α^* in one-shot NAS is usually formulated as a two-stage optimization problem. The first-stage is to optimize the weight W by

$$W_{\mathcal{A}} = \arg\min_{W} \mathcal{L}_{train}(\mathcal{N}(\mathcal{A}, W)), \qquad (2)$$

where $\mathcal{L}_{train}$ represents the loss function on the training dataset. To reduce memory usage, one-shot methods usually sample subnets from $\mathcal{N}$ for optimization. The second-stage is to search architectures via ranking the performance of subnets $\alpha \in \mathcal{A}$ based on the learned weights in $W_{\mathcal{A}}$:

$$\alpha^* = \arg\max_{\alpha\in\mathcal{A}} \ \text{Acc}_{val}\left(\mathcal{N}(\alpha, w)\right), \qquad (3)$$

where the sampled subnet α inherits weight w from $W_{\mathcal{A}}$, and Acc_{val} indicates the top-1 accuracy of the architecture α on the validation dataset. Since it is impossible to enumerate all the architectures $\alpha \in \mathcal{A}$ for evaluation, prior works resort to random search [34, 4], evolution algorithms [43, 16] or reinforcement learning [40, 48] to find the most promising one.

3. AutoFormer

In this section, we first demonstrate that it is impractical to directly apply one-shot NAS for transformer search following classical weight sharing strategy [16], using different weights for different blocks in each layer, because of the slow convergence and unsatisfactory performance. Then we propose the weight entanglement strategy for vision transformer to address the issues. Finally, we present the search space and search pipeline.

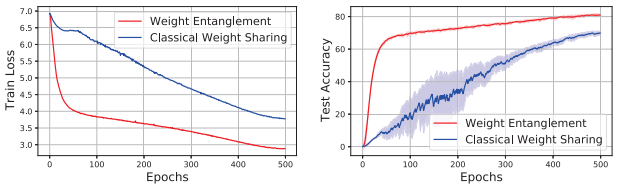

Figure 4. **Left:** Comparison of training loss of supernet between weight entanglement and classical weight sharing on ImageNet. **Right:** Comparison of Top-1 Accuracy on ImageNet of subnets between weight entanglement and classical weight sharing during supernet training.

3.1. One-Shot NAS with Weight Entanglement

Prior one-shot NAS methods commonly share weights across architectures during supernet training while decoupling the weights of different operators at the same layer. Such strategy performs well when used to search architectures over convolutional neural networks space [16, 6, 8, 21, 49]. However, in transformer search space, this classical strategy encounters difficulties. 1) Slow convergence. As shown in the Fig. 4 (left), the training loss of the supernet converges slowly. The reason might be that the independent training of transformer blocks results in the weights being updated by limited times. 2) Low performance. The performances of subnets inheriting weights from the one-shot supernet, trained under classical weight sharing strategy, are far below their true performances of training from scratch (see the right part of Fig. 4). This limits the ranking capacities of supernet. Furthermore, after the search, it is still necessary to perform additional retraining for the searched architectures since the weights are not fully optimized. Inspried by BigNAS [65] and slimmable networks [66, 64], we propose the weight entanglement training strategy dedicated to vision transformer architecture search. The central idea is to enable different transformer blocks to share weights for their common parts in each layer. More concretely, for a subnet $\alpha \in \mathcal{A}$ with a stack of l layers, we represent its structure and weights as

$$\begin{cases} \alpha = (\alpha^{(1)}, ...\alpha^{(i)}, ...\alpha^{(l)}), \\ w = (w^{(1)}, ...w^{(i)}, ...w^{(l)}), \end{cases} \tag{4}$$

where $\alpha^{(i)}$ denotes the sampled block in the i-th layer and $w^{(i)}$ is the block weights. During architecture search, there are multiple choices of blocks in each layer. Hence, $\alpha^{(i)}$ and $w^{(i)}$ are actually selected from a set of n block candidates belonging to the search space, which is formulated as

$$\begin{cases} \alpha^{(i)} \in \{b_1^{(i)}, ...b_j^{(i)}, ...b_n^{(i)}\}, \\ w^{(i)} \in \{w_1^{(i)}, ...w_j^{(i)}, ...w_n^{(i)}\}, \end{cases} \tag{5}$$

where $b_j^{(i)}$ is a candidate block in the search space and $w_j^{(i)}$ is its weights. The weight entanglement strategy enforces

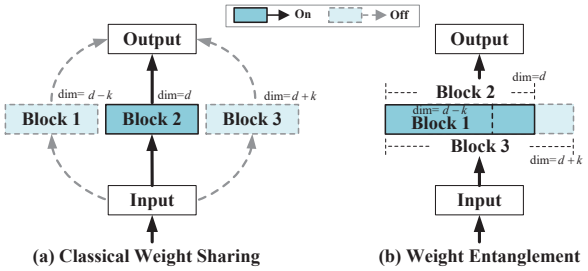

Figure 5. Classical weight sharing versus weight entanglement.

that different candidate blocks in the same layer to share as many weights as possible. This requires that, for any two blocks $b_j^{(i)}$ and $b_k^{(i)}$ in the same layer, we have

$$w_j^{(i)} \subseteq w_k^{(i)} \quad \text{or} \quad w_k^{(i)} \subseteq w_j^{(i)}. \tag{6}$$

Such within layer weight sharing makes the weight updates of $w_j^{(i)}$ and $w_k^{(i)}$ entangled with each other. The training of any block will affect the weights of others for their intersected portion, as demonstrated in Fig. 5. This is different from the classical weight sharing strategy in one-shot NAS, where the building blocks in the same layer are isolated. In other words, in classical weight sharing, for any two blocks $b_j^{(i)}$ and $b_k^{(i)}$, we have $w_j^{(i)} \cap w_k^{(i)} = \emptyset$.

Note that the proposed weight entanglement strategy is dedicated to work on homogeneous building blocks, such as self-attention modules with different numbers of heads, and multi-layer perceptron with different hidden dimensions. The underlying reason is that homogeneous blocks are structurally compatible, such that the weights can share with each other. During implementation, for each layer, we need to store only the weights of the largest block among the n homogeneous candidates. The remaining smaller building blocks can directly extract weights from the largest one.

Equipped with weight entanglement, one-shot NAS is capable of searching transformer architectures in an efficient and effective fashion, as demonstrated in Fig. 4.

Compared with classical weight sharing methods, our weight entanglement strategy has three advantages. 1) *Faster convergence*. Weight entanglement allows each block to be updated more times than the previous independent training strategy. 2) *Low memory cost*. We now only need to store the largest building blocks' parameters for each layer, instead of all the candidates in the space. 3) *Better subnets performance*. We found that the subnets trained with weight entanglement could achieve performance on par with those of training from scratch.

3.2. Search Space

We design a large transformer search space that includes five variable factors in transformer building blocks: embedding dimension, Q-K-V dimension, number of heads, MLP ratio, and network depth, as detailed in Tab. 1 and Fig. 3. All

	Supernet-tiny	Supernet-small	Supernet-base
Embed Dim	(192, 240, 24)	(320, 448, 64)	(528, 624, 48)
Q-K-V Dim	(192, 256, 64)	(320, 448, 64)	(512, 640, 64)
MLP Ratio	(3.5, 4, 0.5)	(3, 4, 0.5)	(3, 4, 0.5)
Head Num	(3, 4, 1)	(5, 7, 1)	(8, 10, 1)
Depth Num	(12, 14, 1)	(12, 14, 1)	(14, 16, 1)
Params Range	4 – 9M	14 – 34M	42 – 75M

Table 1. The search space of AutoFormer. We set up three supernets to satisfy different resource constraints. Tuples of three values in parentheses represent the lowest value, highest, and steps. **Note:** the Q-K-V dimensions, numbers of head and MLP ratios are varied across layers.

these factors are important for model capacities. For example, in attention layers, different heads are used to capture various dependencies. However, recent works [36, 53, 9] show that many heads are redundant. We thereby make the attention head number elastic so that each attention module can decide its necessary number of heads. On the other hand, since different layers have different capacities on feature representation, the varying hidden dimensions in layers might be better than the fixed sizes when used for constructing new models. Moreover, AutoFormer adds new Q-K-V dimension into the search space and fixes the ratio of the Q-K-V dimension to the number of heads in each block. This setting makes the scaling factor $\frac{1}{\sqrt{d_h}}$ in attention calculation invariant to the number of heads, stabilizing the gradients, and decouples the meaning of different heads.

Following one-shot NAS methods, we encode the search space into a supernet. That is, every model in the space is a part/subset of the supernet. All subnets share the weights of their common parts. The supernet is the largest model in the space, and its architecture is shown in Fig. 3. In particular, the supernet stacks the maximum number of transformer blocks with the largest embedding dimension, Q-K-V dimension and MLP ratio as defined in the space. During training, all possible subnets are uniformly sampled, and the corresponding weights are updated.

According to the constraints on model parameters, we partition the large-scale search space in to three parts and encode them into three independent supernets, as elaborated in Tab. 1. Such partition allows the search algorithm to concentrate on finding models within a specific parameter range, which can be specialized by users according to their available resources and application requirements.

Overall, our supernets contains more than 1.7×10^{16} candidate architectures covering a wide range of model size.

3.3. Search Pipeline

Our search pipeline includes two sequential phases.

Phase 1: Supernet Training with Weight Entanglement. In each training iteration, we uniformly sample a subnet $\alpha = (\alpha^{(1)}, ...\alpha^{(i)}, ...\alpha^{(l)})$ from the per-defined search space and update its corresponding weights $w = (w^{(1)}, ...w^{(i)}, ...w^{(l)})$ in the supernet's weight $W_{\mathcal{A}}$ while freezing the rest. Detailed algorithm is given in supplementary materials, Appendix A.

Phase 2: Evolution Search under Resource Constraints. After obtaining the trained supernet, we perform an evolution search on it to obtain the optimal subnets. Subnets are evaluated and picked according to the manager of the evolution algorithm. Our objective here is to maximize the classification accuracy while minimizing the model size. At the beginning of the evolution search, we pick N random architecture as seeds. The top k architectures are picked as parents to generate the next generation by crossover and mutation. For a crossover, two randomly selected candidates are picked and crossed to produce a new one during each generation. For mutation, a candidate mutates its depth with probability P_d first. Then it mutates each block with a probability of P_m to produce a new architecture.

3.4. Discussion

Why does weight entanglement work? We conjecture that there are two underlying reasons. 1) Regularization in training. Different from convolution neural networks, transformer has no convolution operations at all. Its two basic components, MSA and MLP, employ only fully connected layers. Weight entanglement could be viewed a regularization training strategy for transformer, to some extent, similar to the effects of dropout [47, 54, 30]. When sampling the small subnets, corresponding units cannot rely on other hidden units for classification, which hence reduces the reliance of units. 2) Optimization of deep thin subnets. Recent works [3, 57] show that deep transformer is hard to train, which coincides with our observation in Fig. 2. This is because the gradients might explode or vanish in deep thin networks during backpropagation. Increasing the width of or "overparameterizing" the network will help the optimization [33, 14, 1, 71]. Our weight entanglement training strategy helps to optimize the thin subnets in a similar way. The gradients backwarded by wide subnets will help to update the weights of thin subnets. Besides, the elastic depth severs similar effects to stochastics depth [23] and deep supervision [29], which supervise the shallow layers as well.

4. Experiments

In this section, we first present the implementation details and evolution search settings. We then analyze the proposed weight entanglement strategy and provide a large number of well-trained subnets sampled from supernets to demonstrate its efficacy. At last, we present the performance of AutoFormer evaluated on several benchmarks with comparisons with state-of-the-art models designed manually or automatically.

Search Method	Inherited	Retrain	Params
Random Search	-	79.4%	23.0M
Classical Weight Sharing + Random Search	69.7%	80.1%	22.9M
Weight Entanglement + Random Search	81.3%	81.4%	22.8M
Classical Weight Sharing + Evolution Search (SPOS[16])	71.5%	80.4%	22.9M
Weight Entanglement + Evolution Search (Ours)	81.7%	81.7%	22.9M

Table 2. Comparison of different search methods. The supernets are trained for 500 epochs, while the subnets are retrained for 300 epochs. Random search are performed three times and the best performance is reported.

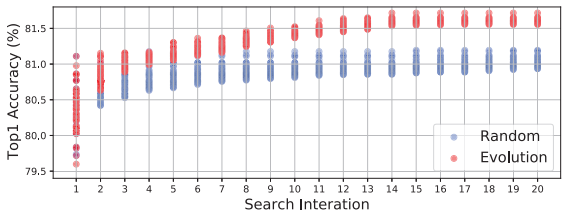

Figure 6. The performance of subnets inheriting weights from supernet during search. Top 50 candidates until the current iteration are depicted at each search iteration.

Epochs	Optimizer	Batch Size	LR	LR scheduler
500	AdamW	1024	1e-3	cosine
Weight Decay	Warmup Epochs	Label Smoothing	Drop Path	Repeated Augmentation
5e-2	20	0.1	0.1	✗

Table 3. Supernet training settings. LR refers to learning rate.

Model	Model Size	Inherited	Finetune	Retrain
AutoFormer-T	5.7M	74.7%	74.9%	74.9%
AutoFormer-S	22.9M	81.7%	81.8%	81.7%
AutoFormer-B	53.7M	82.4%	82.6%	82.6%

Table 4. Comparison of subnets with inherited weights, fine-tuned (40 epochs) and trained from-scratch (300 epochs).

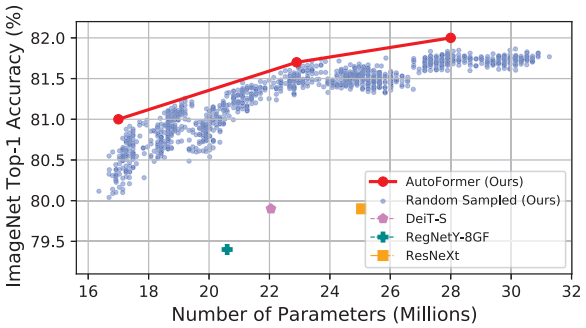

Figure 7. Top-1 accuracy on ImageNet of AutoFormer and 1000 sampled high-performing architectures from the supernet-small with weight inherited from the supernet.

4.1. Implementation Details

Supernet Training. We train the supernets using a similar recipe as DeiT [50]. The details are presented in Tab. 3. Data augmentation techniques, including RandAugment [10], Cutmix [68], Mixup [69] and random erasing, are adopted with the same hyperparameters as in DeiT [50] except the repeated augmentation [20]. Images are split into patches of size 16x16. All the models are implemented using PyTorch 1.7 and trained on Nvidia Tesla V100 GPUs.

Evolutionary Search. The implementation of evolution search follows the same protocol as in SPOS [16]. For a fair comparison, we reserve the ImageNet validation set for *testing* and subsample 10,000 training examples (100 images per class) as the validation dataset. We set the population size to 50 and number of generations to 20. Each generation we pick the top 10 architectures as the parents to generate child networks by mutation and crossover. The mutation probability P_d and P_m are set to 0.2 and 0.4.

4.2. Ablation Study and Analysis

The Efficacy of Weight Entanglement. We compare AutoFormer with random search and SPOS [16] (classical weight sharing) baselines to demonstrate the effectiveness of weight entanglement. For random search, we randomly pick up architectures from the search space to meet the model size constraints. For SPOS [16], we adapt it to the transformer search space defined in Tab. 1 and keep the remaining configurations to be consistent with the original method. In other words, in each layer of the SPOS supernet, the transformer blocks with different architecture arguments do not share weights. For example, in an MLP block, there are multiple choices of hidden dimensions. Each MLP choice has its own weights, being independent of each other. After training the SPOS supernet, we apply the same evolution process to find the most promising architecture candidate and retrain it using the same setting as our AutoFormer.

Table 2 presents the comparisons on ImageNet. We can observe that: 1) After retraining,random search and SPOS are 2.3% and 1.3% inferior to our methods indicating the superiority of the proposed methods. 2) Without retraining, *i.e.*, inheriting weights directly from the supernet, the weight entanglement training strategy can produce significantly better results than the classical weight sharing. The entangled weight can produce well-trained subnets, which are very close to the ones retrained from scratch. We conjecture the relatively inferior performance of SPOS in transformer space is mainly due to insufficient training. We also observe that if we train the supernet in SPOS for more epochs, the performance can be slowly improved. However, its training cost is largely higher than our proposed weight entanglement strategy. Fig. 6 plots the accuracy over the number of architectures sampled from the trained supernet during search. Top 50 candidates are depicted at each generation. It is clear that the evolution search on the supernet is more effective than the random search baseline.

Table 5. AutoFormer performance on ImageNet with comparisons to state-of-the-arts. We group the models according to their parameter sizes. Our AutoFormer consistently outperforms existing transformer-based visual models, being comparable to CNN models.†: reported by [50], ⋆: reported by [63].

Models	Top-1 Acc.	Top-5 Acc.	#Parameters	FLOPs	Resolution	Model Type	Design Type
MobileNetV3$_{Large1.0}$ [21]	75.2%	-	5.4M	0.22G	224^2	CNN	Auto
EfficietNet-B0[49]	77.1%	93.3%	5.4M	0.39G	224^2	CNN	Auto
DeiT-tiny [50]	72.2%	91.1%	5.7M	1.2G	224^2	Transformer	Manual
AutoFormer-tiny (Ours)	**74.7%**	**92.6%**	**5.7M**	**1.3G**	224^2	Transformer	Auto
ResNet50⋆ [18]	79.1%	-	25.5M	4.1G	224^2	CNN	Manual
RegNetY-4GF† [41]	80.0%	-	21.4M	4.0G	224^2	CNN	Auto
EfficietNet-B4 [49]	82.9%	95.7%	19.3M	4.2G	380^2	CNN	Auto
BoTNet-S1-59 [46]	81.7%	95.8%	33.5M	7.3G	224^2	CNN + Trans	Manual
T2T-ViT-14 [67]	81.7%	-	21.5M	6.1G	224^2	Transformer	Manual
DeiT-S [50]	79.9%	95.0%	22.1M	4.7G	224^2	Transformer	Manual
ViT-S/16 [13]	78.8%	-	22.1M	4.7G	384^2	Transformer	Manual
AutoFormer-small (Ours)	**81.7%**	**95.7%**	**22.9M**	**5.1G**	224^2	Transformer	Auto
ResNet152⋆ [18]	80.8%	-	60M	11G	224^2	CNN	Manual
EfficietNet-B7 [49]	84.3%	97.0%	66M	37G	600^2	CNN	Auto
ViT-B/16 [13]	79.7%	-	86M	18G	384^2	Transformer	Manual
Deit-B [50]	81.8%	95.6%	86M	18G	224^2	Transformer	Manual
AutoFormer-base (Ours)	**82.4%**	**95.7%**	**54M**	**11G**	224^2	Transformer	Auto

Subnet Performance without Retraining. We surprisingly observe that *there are a large number of subnets achieving very good performance when inheriting weights from the supernets, without extra finetuning or retraining*. The blue points shown in Fig. 7 represents the 1000 high-performing subsets sampled from the supernet-S. All these subsets can achieve top-1 accuracies ranging from 80.1% to 82.0%, exceeding the recent DeiT [50] and RegNetY [41]. Such results amply demonstrate the effectiveness of the proposed weight entanglement strategy for one-shot supernet training. Tab. 4 shows that if we further finetune or retrain the searched subnets on ImageNet, the performance gains are very small, even negligible. This phenomenon illustrates the weight entanglement strategy allows the subsets to be well-trained in supernets, leading to the facts that searched transformers do not require any retraining or finetuning and the supernet itself serves good indicator of subnets' ranking.

4.3. Results on ImageNet

We perform the search of AutoFormer on ImageNet and find multiple transformer models with diverse parameter sizes. All these models inherit weights from supernets directly, without extra retraining and other postprocessing. The performance are reported in Tab. 5 and Fig. 1. It is clear that our AutoFormer model families achieve higher accuracies than the recent handcrafted state-of-the-art transformer models such as ViT [13] and DeiT [50]. In particular, using ∼23M parameters, our small model, *i.e.* AutoFormer-S, achieves a top-1 accuracy of 81.7%, being 1.8% and 2.9% better than DeiT-S and ViT-S/16, respectively.

Compared to vanilla CNN models, AutoFormer is also competitive. As visualized in Fig. 1, our AutoFormers perform better than the manually-designed ResNet [18], ResNeXt [62] and DenseNet [22], demonstrating the potentials of pure transformer models for visual representation.

However, transformer-based vision models, including AutoFormer, now are still inferior to the models based on inverted residual blocks [44], such as MobileNetV3 [21] and EfficientNet [49]. The reason is that inverted residuals are optimized for edge devices, so the model sizes and FLOPs are much smaller than vision transformers.

4.4. Transfer Learning Results

Classification. We transfer Autoformer to a list of commonly used recognition datasets: 1) general classification: CIFAR-10 and CIFAR-100 [27]; 2) fine-grained classification: Stanford Car [26], FLowers [37] and Oxford-III Pets [38]. We follow the same training settings as DeiT [50], which take ImageNet pretrained checkpoints and finetune on new datasets. Tab. 6 shows the results in terms of top-1 accuracy: 1) Compared to state-of-the-art ConvNets, AutoFormer is close to the best results with a negligible gap with fewer parameters; 2) Compared to transformer-based models, AutoFormer achieves better or comparable results on all datasets, with much fewer parameters (∼4x).

Distillation. AutoFormer is also orthogonal to knowledge distillation (KD) since we focus on searching for an efficient architecture while KD focuses on better training a given architecture. Combining KD with AutoFormers by distilling hard labels from a RegNetY-32GF [41] teacher could further improve the performance from 74.7%/81.7%/82.4% to 75.7%/82.4%/82.9%, respectively.

Table 6. AutoFormer results on downstream classification datasets. ↑ 384 denotes fine-tuning with 384×384 resolution.

Model	#Param	FLOPs	ImageNet	CIFAR-10	CIFAR-100	Flowers	Cars	Pets	Model Type	Design Type
Grafit ResNet-50 [51]	25M	12.1G	79.6	-	-	98.2	92.5	-	CNN	Manual
Grafit RegNetY-8GF [51]	39M	23.4G	79.6	-	-	99.0	94.0	-	CNN	Manual
EfficientNet-B5 [49]	30M	9.5G	83.6	98.7	91.1	98.5	-	-	CNN	Auto
ViT-B/16 [13]	86M	55.4G	77.9	98.1	87.1	89.5	-	93.8	Trans	Manual
DeiT-B ↑ 384 [50]	86M	55.4G	83.1	99.1	90.8	98.5	93.3	-	Trans	Manual
AutoFormer-S ↑ 384	23M	16.5G	83.4	99.1	91.1	98.8	93.4	94.9	Trans	Auto

5. Related Work

Vision Transformer. Transformer is originally proposed for language modeling [52], and recently applied in computer vision. It has shown promising potentials on a variety of tasks [7, 13, 35]. A straightforward approach for using transformer in vision is to combine convolutional layers with the self-attention module [52, 58]. There has been progress in this direction, such as [42, 70, 56, 24].

Most recently, Dosovitskiy *et al.* introduce Vision Transformer (ViT) [13], a pure transformer architecture for visual recognition. It presents promising results when trained with an extensive image dataset (JFT-300M, 300 million images) that is not publicly available. The most recent DeiT [50, 67] models verify that large-scale data is not required. Using only Imagenet can also produce a competitive convolution-free transformer. However, existing visions of transformer models are all built upon manual design, which is engineering-expensive and error-prone. In this work, we present the first effort on automating the design of vision transformer with neural architecture search.

Neural Architecture Search. There has been an increasing interest in NAS for automating network design [15, 25]. Early approaches search a network using either reinforcement learning [73, 72] or evolution algorithms [61, 43]. Most recent works resort to the one-shot weight sharing strategy to amortize the searching cost [34, 40, 5, 16]. The key idea is to train a over-parameterized supernet model, and then share the weights across subnets. However, most weight-sharing methods need an additional *retraining* step after the best architecture is identified [16, 31, 59].

Recent works, OFA [6], BigNAS [65] and slimmable networks [66, 64] alleviate this issue by training a once-for-all supernet. Despite the fact that AutoFormer shares similarities with these methods in the concept of training a one-for-all supernet, these methods are designed to search for convolutional networks rather than vision transformers. Specifically, AutoFormer considers the design of multi-head self-attention and MLP, which is unique to transformer models, and gives dedicated design of search dimensions as elaborated in Sec. 3.2. Moreover, BigNAS adopts several well-crafted techniques, such as sandwich training, inplace distillation, regularization, *etc*. OFA proposes a progressively shrinking approach by progressively distilling the full network to obtain the smaller subnets. By contrast, AutoFormer is simple and efficient, achieving once-for-all training without these techniques.

For transformers, there are few studies applying NAS to improve their architectures [45, 55]. These approaches mainly focus on natural language processing tasks. Among them, the most similar one to us is HAT [55]. In addition to the difference between tasks, HAT requires an additional *retraining* or *finetuning* step after the search, while AutoFormer does not, which is the key difference. Another difference is the search space. HAT searches for an encoder-decoder Transformer structure, while ours is a pure encoder one. There are two concurrent works, *i.e.*, BossNAS [32] and CvT [60], exploring different search space from ours. BossNAS searches for CNN-transformer hybrids, while ours for pure transformers. CvT proposes a new architecture family and searches for the strides and kernel size of them. Due to the difference of search space, we do not compare them in this work.

6. Conclusion

In this work, we propose a new one-shot architecture search method, AutoFormer, dedicated to transformer search. AutoFormer is equipped with the training strategy, *Weight Entanglement*. Under this strategy, the subnets in the search space are almost fully trained. Extensive experiments demonstrate the proposed algorithm can improve the training of supernet and find promising architectures. Our searched AutoFormers achieve state-of-the-art results on ImageNet among vision transformers. Moreover, AutoFormers transfer well to several downstream classification tasks and could be further improved by distillation. In future work, we are interested in further enriching the search space by including convolutions as new candidate operators. Applying weight entanglement to convolution network search or giving the theoretical analysis of the weight entanglement are other potential research directions.

Acknowledgement

We are very grateful to Xingxing Zhang's valuable advice and discussion.

References

[1] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. A convergence theory for deep learning via over-parameterization. In *ICML*, 2019. 5

[2] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *NeurIPS*, 2016. 3

[3] Ankur Bapna, Mia Xu Chen, Orhan Firat, Yuan Cao, and Yonghui Wu. Training deeper neural machine translation models with transparent attention. In *EMNLP*, 2018. 5

[4] Gabriel Bender, Pieter-Jan Kindermans, Barret Zoph, Vijay Vasudevan, and Quoc Le. Understanding and simplifying one-shot architecture search. In *ICML*, 2018. 3

[5] Andrew Brock, Theodore Lim, James M Ritchie, and Nick Weston. Smash: one-shot model architecture search through hypernetworks. In *ICLR*, 2018. 8

[6] Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once for all: Train one network and specialize it for efficient deployment. In *ICLR*, 2020. 4, 8

[7] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*, 2020. 8

[8] Xiangxiang Chu, Bo Zhang, Ruijun Xu, and Jixiang Li. Fairnas: Rethinking evaluation fairness of weight sharing neural architecture search. *arXiv preprint arXiv:1907.01845*, 2019. 2, 4

[9] Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D Manning. What does bert look at? an analysis of bert's attention. *ACL*, 2019. 5

[10] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *CVPR Workshops*, 2020. 6

[11] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 2

[12] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, 2019. 2

[13] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*, 2021. 1, 2, 7, 8

[14] Simon S Du, Xiyu Zhai, Barnabas Poczos, and Aarti Singh. Gradient descent provably optimizes over-parameterized neural networks. In *ICLR*, 2018. 5

[15] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *JMLR*, 2019. 8

[16] Zichao Guo, Xiangyu Zhang, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, and Jian Sun. Single path one-shot neural architecture search with uniform sampling. *ECCV*, 2020. 2, 3, 4, 6, 8

[17] Kai Han, Yunhe Wang, Hanting Chen, Xinghao Chen, Jianyuan Guo, Zhenhua Liu, Yehui Tang, An Xiao, Chunjing Xu, Yixing Xu, et al. A survey on visual transformer. *arXiv preprint arXiv:2012.12556*, 2020. 1

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1, 7

[19] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016. 3

[20] Elad Hoffer, Tal Ben-Nun, Itay Hubara, Niv Giladi, Torsten Hoefler, and Daniel Soudry. Augment your batch: better training with larger batches. *CVPR*, 2020. 6

[21] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *ICCV*, 2019. 4, 7

[22] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *CVPR*, 2017. 7

[23] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *ECCV*, 2016. 5

[24] Zilong Huang, Xinggang Wang, Lichao Huang, Chang Huang, Yunchao Wei, and Wenyu Liu. Ccnet: Criss-cross attention for semantic segmentation. In *ICCV*, 2019. 8

[25] Salman Khan, Muzammal Naseer, Munawar Hayat, Syed Waqas Zamir, Fahad Shahbaz Khan, and Mubarak Shah. Transformers in vision: A survey. *arXiv preprint arXiv:2101.01169*, 2021. 1, 8

[26] Jonathan Krause, Jia Deng, Michael Stark, and Li Fei-Fei. Collecting a large-scale dataset of fine-grained cars. 2013. 7

[27] Alex Krizhevsky et al. Learning multiple layers of features from tiny images. 2009. 7

[28] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. *ICLR*, 2020. 2

[29] Chen-Yu Lee, Saining Xie, Patrick Gallagher, Zhengyou Zhang, and Zhuowen Tu. Deeply-supervised nets. In *AISTATS*, 2015. 5

[30] Changlin Li, Jiefeng Peng, Liuchun Yuan, Guangrun Wang, Xiaodan Liang, Liang Lin, and Xiaojun Chang. Block-wisely supervised neural architecture search with knowledge distillation. In *CVPR*, 2020. 5

[31] Changlin Li, Jiefeng Peng, Liuchun Yuan, Guangrun Wang, Xiaodan Liang, Liang Lin, and Xiaojun Chang. Blockwisely supervised neural architecture search with knowledge distillation. In *CVPR*, 2020. 8

[32] Changlin Li, Tao Tang, Guangrun Wang, Jiefeng Peng, Bing Wang, Xiaodan Liang, and Xiaojun Chang. Bossnas: Exploring hybrid cnn-transformers with block-wisely self-supervised neural architecture search. *arXiv preprint arXiv:2103.12424*, 2021. 8

[33] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. *NeurIPS*, 2017. 5

[34] Liam Li and Ameet Talwalkar. Random search and reproducibility for neural architecture search. In *UAI*, 2019. 3, 8

[35] Justin Liang, Namdar Homayounfar, Wei-Chiu Ma, Yuwen Xiong, Rui Hu, and Raquel Urtasun. Polytransform: Deep

polygon transformer for instance segmentation. In *CVPR*, 2020. 8

[36] Paul Michel, Omer Levy, and Graham Neubig. Are sixteen heads really better than one? In *NeurIPS*, 2019. 5

[37] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *ICVGIP*, 2008. 7

[38] Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and CV Jawahar. Cats and dogs. In *CVPR*, 2012. 7

[39] Houwen Peng, Hao Du, Hongyuan Yu, Qi Li, Jing Liao, and Jianlong Fu. Cream of the crop: Distilling prioritized paths for one-shot neural architecture search. *NeurIPS*, 2020. 3

[40] Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. Efficient neural architecture search via parameters sharing. In *ICML*, 2018. 3, 8

[41] Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces. In *CVPR*, 2020. 7

[42] Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jonathon Shlens. Stand-alone self-attention in vision models. *arXiv preprint arXiv:1906.05909*, 2019. 8

[43] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In *AAAI*, 2019. 3, 8

[44] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*, 2018. 7

[45] David So, Quoc Le, and Chen Liang. The evolved transformer. In *International Conference on Machine Learning*. PMLR, 2019. 1, 8

[46] Aravind Srinivas, Tsung-Yi Lin, Niki Parmar, Jonathon Shlens, Pieter Abbeel, and Ashish Vaswani. Bottleneck transformers for visual recognition. *arXiv preprint arXiv:2101.11605*, 2021. 7

[47] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *JMLR*, 2014. 5

[48] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. Mnasnet: Platform-aware neural architecture search for mobile. In *CVPR*, 2019. 3

[49] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *ICML*, 2019. 2, 4, 7, 8

[50] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *ICML*, 2021. 1, 2, 6, 7, 8

[51] Hugo Touvron, Alexandre Sablayrolles, Matthijs Douze, Matthieu Cord, and Hervé Jégou. Grafit: Learning fine-grained image representations with coarse labels. *arXiv preprint arXiv:2011.12982*, 2020. 8

[52] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017. 1, 2, 8

[53] Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *ACL*, 2019. 5

[54] Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In *ICML*, 2013. 5

[55] Hanrui Wang, Zhanghao Wu, Zhijian Liu, Han Cai, Ligeng Zhu, Chuang Gan, and Song Han. Hat: Hardware-aware transformers for efficient natural language processing. *arXiv preprint arXiv:2005.14187*, 2020. 1, 8

[56] Huiyu Wang, Yukun Zhu, Bradley Green, Hartwig Adam, Alan Yuille, and Liang-Chieh Chen. Axial-deeplab: Stand-alone axial-attention for panoptic segmentation. In *ECCV*, 2020. 8

[57] Qiang Wang, Bei Li, Tong Xiao, Jingbo Zhu, Changliang Li, Derek F Wong, and Lidia S Chao. Learning deep transformer models for machine translation. In *ACL*, 2019. 5

[58] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *CVPR*, 2018. 8

[59] Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Peter Vajda, Yangqing Jia, and Kurt Keutzer. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In *CVPR*, 2019. 2, 8

[60] Haiping Wu, Bin Xiao, Noel Codella, Mengchen Liu, Xiyang Dai, Lu Yuan, and Lei Zhang. Cvt: Introducing convolutions to vision transformers. *arXiv preprint arXiv:2103.15808*, 2021. 8

[61] Lingxi Xie and Alan Yuille. Genetic cnn. In *ICCV*, 2017. 8

[62] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *CVPR*, 2017. 7

[63] Weijian Xu, Yifan Xu, Tyler Chang, and Zhuowen Tu. Co-scale conv-attentional image transformers, 2021. 7

[64] Jiahui Yu and Thomas S Huang. Universally slimmable networks and improved training techniques. In *ICCV*, 2019. 4, 8

[65] Jiahui Yu, Pengchong Jin, Hanxiao Liu, Gabriel Bender, Pieter-Jan Kindermans, Mingxing Tan, Thomas Huang, Xiaodan Song, Ruoming Pang, and Quoc Le. Bignas: Scaling up neural architecture search with big single-stage models. *NeurIPS*, 2020. 2, 4, 8

[66] Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas Huang. Slimmable neural networks. *ICLR*, 2019. 2, 4, 8

[67] Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Francis EH Tay, Jiashi Feng, and Shuicheng Yan. Tokens-to-token vit: Training vision transformers from scratch on imagenet. *arXiv preprint arXiv:2101.11986*, 2021. 1, 2, 7, 8

[68] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *ICCV*, 2019. 6

[69] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *ICLR*, 2018. 6

[70] Hengshuang Zhao, Jiaya Jia, and Vladlen Koltun. Exploring self-attention for image recognition. In *CVPR*, 2020. 8

[71] Denny Zhou, Mao Ye, Chen Chen, Tianjian Meng, Mingxing Tan, Xiaodan Song, Quoc Le, Qiang Liu, and Dale Schuurmans. Go wide, then narrow: Efficient training of deep thin networks. In *ICML*, 2020. 5

[72] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *ICLR*, 2016. 8

[73] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *CVPR*, 2018. 8