# AutoFormer: Searching Transformers for Visual Recognition
# —— Supplementary Material ——

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

**Algorithm 1** Supernet Training with Weight Entanglement

**Input:**
Training epochs $N$, search space $\mathcal{A}$, supernet $\mathcal{N}$, initial supernet weights $W_{\mathcal{A}}$, train dataset $D_{\text{train}}$, loss function *Loss*

**Output:** Well-trained supernet

**for** $i = 1$ to $N$ **do**
  **for** *data, labels* in $D_{\text{train}}$ **do**
    Random sample one transformer architecture $\alpha = (\alpha^{(1)}, ...\alpha^{(i)}, ...\alpha^{(l)})$ from the space $\mathcal{A}$
    Obtain the corresponding weights $w = (w^{(1)}, ...w^{(i)}, ...w^{(l)})$ from $W_{\mathcal{A}}$, where $l$ is the maximum depth
    Compute the gradients $\nabla w$ based on *Loss, data, labels*
    Update the corresponding part of $w$ in $W_{\mathcal{A}}$ while freeze the rest part of the superet $\mathcal{N}$
  **end for**
**end for**

**Algorithm 2** Evolution Search

**Input:**
Search space $\mathcal{A}$, supernet $\mathcal{N}$, supernet weights $W_{\mathcal{A}}$, population size $P$, resources constraints $C$, number of generation iteration $\mathcal{T}$, validation dataset $D_{\text{val}}$, mutation probability of depth $P_d$, mutation probability of each layer $P_m$

**Output:** The most promising transformer $\alpha^*$

$G_{(0)}$ := Random sample $P$ transformer architectures $\{\alpha_1, \alpha_2, \cdots \alpha_P\}$ from $\mathcal{A}$ with the constrain $C$
**while** search step $t \in (0, \mathcal{T})$ **do**
  **while** $\alpha_i \in G_{(t)}$ **do**
    Obtain the corresponding weight $W_{\alpha_i}$ from the supernet weights $W_{\mathcal{A}}$
    Obtain the accuracy of the subnet $\mathcal{N}(\alpha_i, W_{\alpha_i})$ on $D_{\text{val}}$
  **end while**
  $G_{\text{topk}}$ := the Top $K$ candidates by accuracy order;
  $G_{\text{crossover}} := \text{Crossover}(G_{\text{topk}}, S, C)$
  $G_{\text{mutation}} := \text{Mutation}(G_{\text{topk}}, P_d, P_m, S, C)$
  $G_{(t+1)} = G_{\text{crossover}} \cup G_{\text{mutation}}$
**end while**

# A. Appendix

## A.1. Search Pipeline

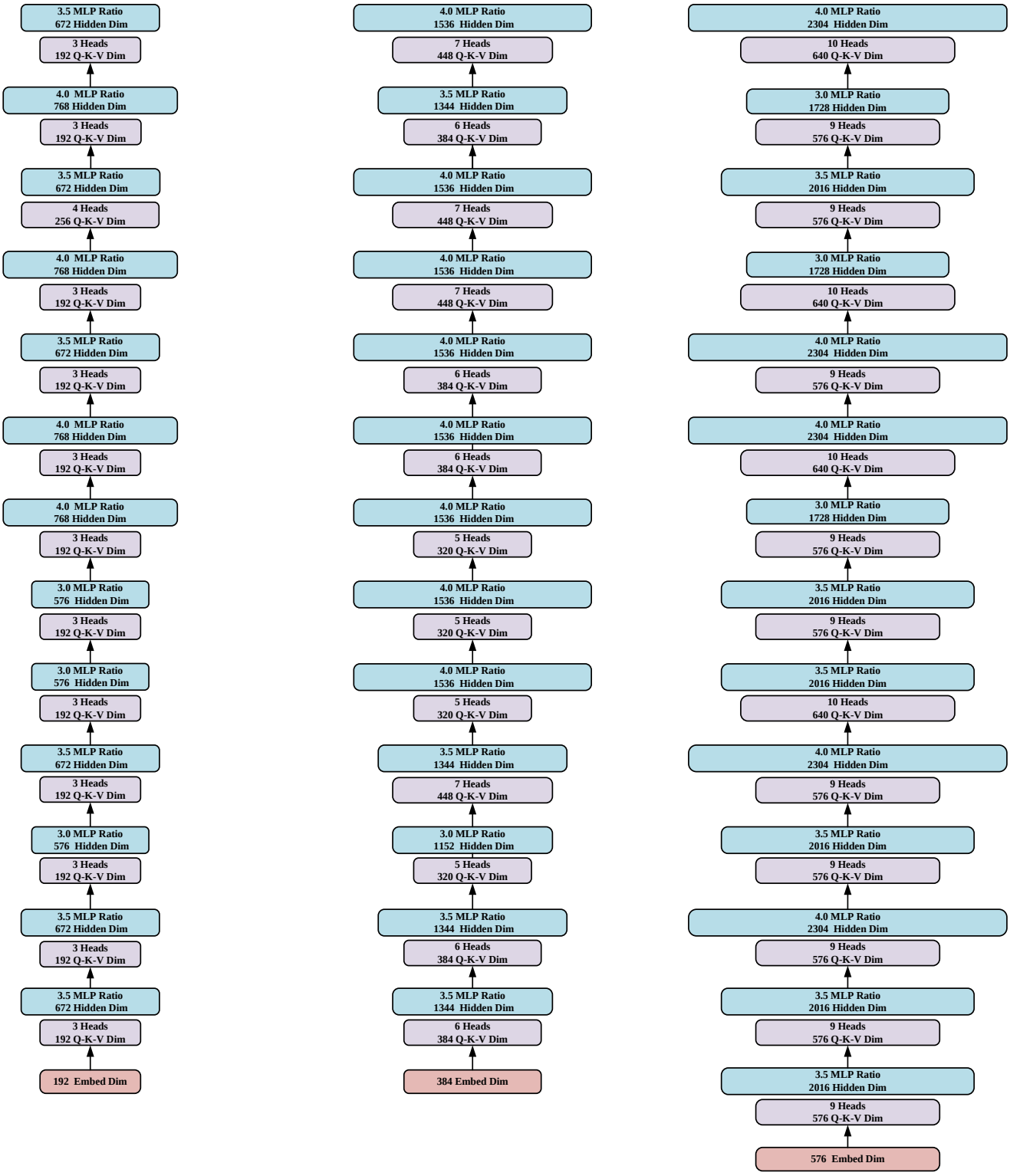
This section presents the details of supernet training and evolutionary algorithm. Alg. 1 elaborates the procedure of supernet training with weight entanglement. In each iteration, we will first randomly sample one transformer architecture $\alpha$. Then we obtain its weights from the supernet's weights $W_{\mathcal{A}}$ and compute loss using the subnet $\mathcal{N}(\alpha, w)$. At last, we update the corresponding weights in $W_{\mathcal{A}}$ while freezing the rest. Alg. 2 shows the evolution search in our method. For crossover, two randomly selected candidate architectures are picked from the top candidates first. Then we uniformly choose one block from them in each layer to generate a new architecture. 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. New produced architectures that do not satisfy the constraints will not be added to the next generation.

## A.2. Structures of AutoFormers

Fig. 1 plots the transformer architectures searched by our AutoFormer method. We observe an interesting phenomenon that the shallow transformer blocks prefer to large $Q$-$K$-$V$ dimensions and small MLP ratios, while the deep blocks tend to have small $Q$-$K$-$V$ dimensions with large MLP ratios.

## A.3. Additional Analysis and Details

*Ranking Correlation Analysis.* We conduct experiments to show the ranking capability of the supernet trained with weight entanglement. Tab. 1 shows that the performance of subnets with inherited weights and weights trained from-scratch are very close even comparable. More importantly, they also have the same relative performance order, which indicates the subnet with inherited weights is a good perfor-




**(a) AutoFormer-T** **(b) AutoFormer-S** **(c) AutoFormer-B**

Figure 1. Architectures of AutoFormer-T/S/B. Different widths of blocks sketch different selected choices. The blue and purple blocks represent MLP blocks and multi-head self-attention layers, respectively. Shortcuts are omitted between blocks.

mance proxy of real performance.

*Transfer Learning Details.* For the experiment on 384 $\times$384 resolution, we fine-tune the searched AutoFormer-S model using AdamW optimizer, initial learning rate $10^{-6}$, 30 epochs, batch size 64, minimal learning $10^{-7}$, cosine scheduler, weight decay $5 \times 10^{-2}$, no warmup and the same data augmentation as AutoFormer supernet training. For downstream tasks, similar to DeiT [7], we interpolate the images to 384 $\times$384 for Cars [2], Flowers [4] and Pets [5], while using 224 $\times$224 for CIFAR-10 and CIFAR-100 [3]. We use SGD optimizer with learning rate $10^{-2}$, batch size 64, weight decay $10^{-8}$, 1000 epochs for CIFAR-10, CIFAR-100 and Cars. We use AdamW optimizer, initial learning rate $5 \times 10^{-4}$, minimal learning $10^{-6}$, cosine scheduler, batch size 64, weight decay $10^{-8}$, 1000 epochs for Flowers and Pets.

*Comparison with HAT and BigNAS.* The table below shows the results when applying BigNAS [10] and HAT [8] to our search space. When compared with BigNAS, we find that the sandwich training and inplace distillation lead to poor convergence of the vision transformer supernet. AutoFormer without these strategies outperforms BigNAS by 5.0% or 1.1% top-1 accuracy using inherited or retrained weights, with less than half search cost in total. Similarly, AutoFormer surpasses HAT by 1.2% or 0.5%, while being 1.6$\times$ faster since it does not need any retraining.

| Search Method | Inherited (%) | Retrain (%) | Params (M) | FLOPs (G) | Total Cost (GPU days) |
|---|---|---|---|---|---|
| HAT | 80.5 | 81.2 | 22.8 | 5.0G | 50 |
| BigNAS | 76.7 | 80.6 | 22.9 | 4.9G | 72 |
| AutoFormer | 81.7 | 81.7 | 22.9 | 4.9G | 32 |

*Classification Head.* In the original ViT model [1], a [class] embedding is injected into the patch embeddings to learn the representation of an image. For classification, ViT only uses the output embeddings of [class] token as the input of the final multilayer perceptron, while leaving the output embeddings of image patches unused. Inspired by the widely used *Gloval Average Pooling* operation in convolutional neural networks, we use the average of embedding of all patches as the input of the classification head. We surprisingly find that using such a scheme can further enhance the performance of AutoFormer. In particular, under the same model size constraint ($\leq$23M), it has a 0.3% improvement from 81.4% to 81.7% in terms of top-1 accuracy on ImageNet with *Gloval Average Pooling*.

*Position Embedding.* ViT [1] explores different position embedding methods for vision transformer, including no position embedding, 1-dimensional position embedding, 2-dimensional position embedding and relative position embedding. The authors find that 1-dimensional position embedding is the best choice. By contrast, we implement another version of relative position embedding by extending [6] from 1-D to 2-D inputs. More specifically, two different sets of embeddings are learned for horizontal and vertical relation, respectively. Each of them has the same size as the $Q$-$K$-$V$ dimension ($D_h$). We find that such relative position embedding is able to improve the performance of subnets by around 0.6%. Recent study [9] also has similar observation suggesting that relative position embedding enhances the performance of vision transformer.

| Model | Model Size | Inherited | Rank | Retrain | Rank* |
|---|---|---|---|---|---|
| Subnet 1 | 22.1M | 80.2% | 1 | 80.4% | 1 |
| Subnet 2 | 23.1M | 80.7% | 2 | 80.8% | 2 |
| Subnet 3 | 21.4M | 81.1% | 3 | 81.1% | 3 |
| Subnet 4 | 24.4M | 81.5% | 4 | 81.6% | 4 |
| Subnet 5 | 22.9M | 81.7% | 5 | 81.7% | 5 |

Table 1. Performance of five randomly sampled architectures from the supernet-small space with similar model sizes. We rank their performance with the weights inherited from the supernet and trained from scratch. The columns 'Rank' and 'Rank*' mean the rankings of subnets with weights inherited from supernets and retraining, respecitively

## References


[1] 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. 3

[2] Jonathan Krause, Jia Deng, Michael Stark, and Li Fei-Fei. Collecting a large-scale dataset of fine-grained cars. 2013. 3

[3] Alex Krizhevsky et al. Learning multiple layers of features from tiny images. 2009. 3

[4] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *ICVGIP*, 2008. 3

[5] Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and CV Jawahar. Cats and dogs. In *CVPR*, 2012. 3

[6] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. In *NAACL-HLT (2)*, 2018. 3

[7] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *ICML*, 2021. 3

[8] 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. 3

[9] Kan Wu, Houwen Peng, Minghao Chen, Jianlong Fu, and Hongyang Chao. Rethinking and improving relative position encoding for vision transformer. In *ICCV*, 2021. 3

[10] 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. 3