# ShiftNAS: Improving One-shot NAS via Probability Shift Appendix

## 1. Training Settings

The training settings for ViT and CNN models are following **Table 1**.

| Model | Epochs | Batch size | Learning rate | Weight decay | Optimizer | Augmentation |
|---|---|---|---|---|---|---|
| CNN | 500 | 1024 | 5e-1 | 1e-5 | SGD | CropFlip+AutoAugment |
| ViT | 500 | 1024 | 1e-3 | 5e-2 | AdamW | CropFlip+RandAugment +Cutmix+Mixup+random erasing |

Table 1: Experimental configurations.

**Algorithm 1** The Pytorch-style algorithm of supernet training.

```
Require: Supernet architecture $S$ and weight $w$; Architecture generator $AG$; Sampling distribution $\mathbf{B}$; training dataset $(X, Y)$; criterion C; update frequency $q$; Optimizer for $w$ and $\alpha$, $optimizer$.
Ensure: Trained supernet weight $\mathbf{w}$.
  Initialize frequency count and checkpoint weight, $count$ = 0 and $w_{t-1}$=$w$;
  while not convergence do
    Clear gradients, $optimizer.zero\_grad()$;
    Sample a subnet computational resource$\mathbf{b}$, $\mathbf{b} \sim \ \mathbf{B}$;
    Sample a subnet architecture $a$, $a$ = $AG(\mathbf{b}|\alpha)$ ;
    Sample mini-batch of data, $(x, y) \leftarrow (X, Y)$;
    Compute loss, $loss$=C($S(x|w, a), y)$;
    Compute $\nabla_w$ $\nabla_\alpha$, $loss$.backward();
    Update $w$ and $\alpha$, $optimizer.step()$;
    $count$+=1;
    if $count$==$q$ then
      Update $w_t$=$w$ and $count$=0;
      Do Algorithm 2 with $w_t$ and $w_{t-1}$;
      Update $w_{t-1}$=$w_t$;
    end if
  end while
```

## 2. Training Algorithm

We provide a detailed account of the supernet training process in **Algorithm 1**. Unlike the uniform sampling approach, we propose a dynamic distribution $B$ to sample subnets in each iteration. The sampling distribution $B$ is updated every $q$ iterations as shown in **Algorithm 2**. We calculate $\nabla_B$ by performing two forward and backward passes on the current and former supernet weight $w_t$ and $w_{t-1}$. It should be noted that only a batch of data is used in each $B$ update, which keeps the time overhead at an acceptable level.

**Algorithm 2** The Pytorch-style algorithm of distribution update.

```
Require: Supernet architecture $S$; Checkpoint and current weights $w_{t-1}$, $w_t$; Architecture generator $AG$; Sampling distribution $\mathbf{B}$; validation dataset $(X', Y')$; criterion C; Optimizer for $B$, $optimizer\_B$.
Ensure: Updated distribution $\mathbf{B}$.
    Clear gradients, $optimizer\_B.zero\_grad()$;
    Sample a subnet computational resource, $\mathbf{b} \sim \mathbf{B}$;
    Sample a subnet architecture $a$, $a = AG(\mathbf{b}|\alpha)$ ;
    Sample mini-batch of data, $(x', y') \leftarrow (X', Y')$;
    Compute $loss_{t-1}$, $loss_{t-1}$ = C($S(x'|w_{t-1}, a)$,$y'$);
    Compute and save $\nabla_{B_{t-1}}$, $loss_{t-1}.backward()$;
    Clear gradients, $optimizer\_B.zero\_grad()$;
    Compute $loss_t$, $loss_t$ = C($S(x'|w_t, a)$,$y'$);
    Compute and save $\nabla_{B_t}$, $loss_t.backward()$;
    Compute $\nabla_B$=$\nabla_{B_{t-1}}$-$\nabla_{B_t}$;
    Update $B$, $optimizer\_B.step()$;
```

| Epoch | 30 | 60 | 90 | 120 |
|---|---|---|---|---|
| Kendall's tau | 0.24 | 0.63 | 0.75 | 0.72 |

Table 2: The Kendall's tau values in different training stages.

## 3. More Ablation Study

**Correlation between the gradient and the training sufficiency.** In ShiftNAS, we utilize the gradient of $\nabla_B$=$\nabla_{B_{t-1}}$-$\nabla_{B_t}$ to quantify the training sufficiency of subnets, and there is a curiosity about the relationship between the two. To address this, we conducted an experiment on ViT-tiny space to investigate their correlation. Specifically, we followed the steps: **1)** We trained a supernet with few epochs using a uniform sampling strategy. **2)** We randomly selected 30 subnets from the supernet and calculated their scores $\nabla_B$=$\nabla_{B_{t-1}}$-$\nabla_{B_t}$ based on the validation dataset. **3)** These subnets were independently finetuned for one epoch. **4)** After finetuning, the loss variations of the sampled subnets on the validation dataset were recorded.

The Kendall's tau values between the scores $\nabla_B$ and the loss variations are presented in **Table 2**. Our results demonstrate a strong correlation between the gradient and the training sufficiency after training the supernet for 60 epochs.

**Split steps of search space.** In ShiftNAS, the search space is divided into several parts based on computational complexity, e.g., FLOPs. The effect of steps on model performance is discussed here, using experiments carried out on ViT-tiny space where each supernet is trained under the same training setting. The search space is split from 1.3 GFLOPs to 1.9 GFLOPs with 0.2, 0.1, and 0.05 GFLOPs steps, respectively. As shown in **Figure 1 (a)**, it can be observed that the 0.1 step obtains the best performance in most cases. Empirically, a larger step leads to a smaller search space since the AG only needs to search the optimal subnets along the steps. Therefore, these subnets sampled from a smaller search space can be trained more sufficiently, which is also mentioned in [2]. However, a large step means that we cannot obtain a fine-grained optimal subnet. Therefore, 0.1 steps are chosen for ShiftNAS to balance performance and deployment.

**Update frequency of the sampling distribution vector.** To investigate the effect of different update frequency, we conducted experiments by setting the update frequency $q$ as 50, 100, 500, and 1000 iterations. The results, shown in **Figure 1 (b)**, indicate that the subnet performance decreases as the update frequency decreases, for different computational constraints. This phenomenon suggests that the optimal sampling distribution varies under different training stages, and frequent updates can help to better adapt to the changing training dynamics.

**The efficiency of architecture generator.** To validate the efficiency of the architecture generator (AG), we compared the time required for AG and without AG when sampling subnets of different FLOPs. Without AG, we randomly sample subnets until it finds one that meets the computational constraint. The experimental results, as shown in Figure 2, demonstrate that AG can directly infer architectures of any computational complexity, while random sample takes hundreds or thousands of times longer. For example, during searching ViT-tiny on ImageNet-1k, an additional 52 hours ($1250 iters \times 500 epochs \times 0.4s$) is required.

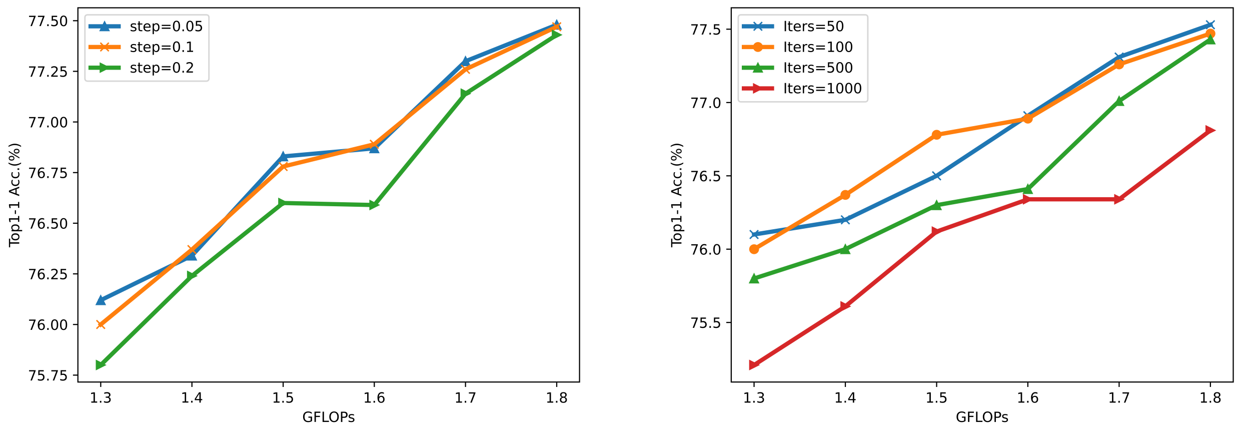

Figure 1: FLOPs/Accuracy tradeoffs of ShiftNAS with different (a) steps, (b) update frequency

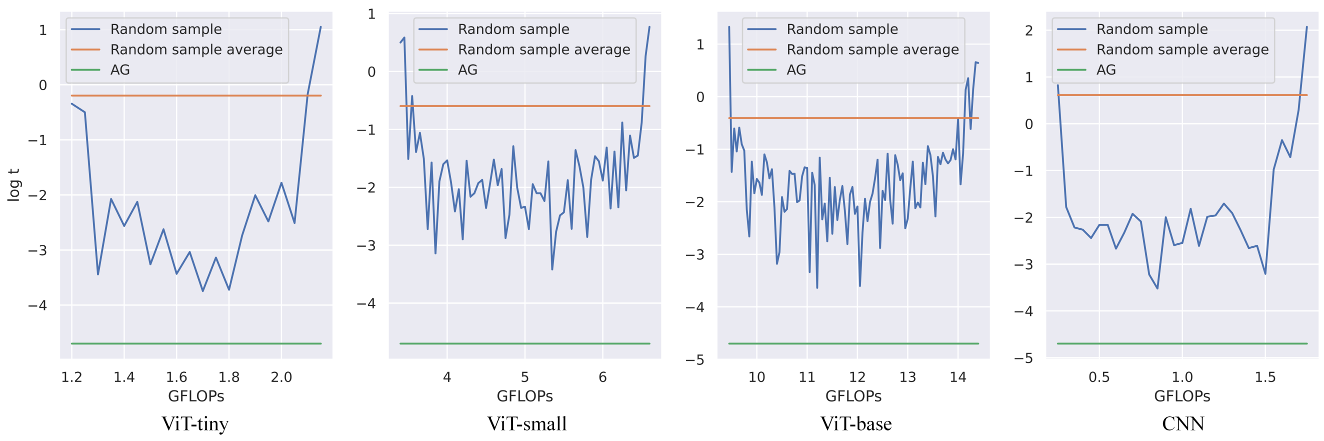

Figure 2: Sampling time comparison. Y label is $logt$.

## 4. Comparisons under AttentiveNAS search space

AttentiveNAS [3] has introduced a method to dynamically sample subnets during supernet training. However, this method employs a more comprehensive search space than ours, as illustrated in **Figure 9 in the AttentiveNAS Appendix**. To ensure a fair comparison, we have trained the supernet with **500 epochs on AttentiveNAS search space**. The comparative results are presented in **Table 3**. Notably, our ShiftCNN models can outperform the AttentiveNAS models under comparable FLOPs constraints. Additionally, it is worth mentioning that ShiftNAS consumes fewer training epochs than AttentiveNAS due to the utilization of the sandwich rule [4] in training the supernet.

## 5. Transfer for Segmentation Tasks

To assess the transferability of our proposed approach to other computer vision tasks, we have conducted experiments on segmentation using the ADE20k dataset. In this regard, we have employed SegViT [5] as our framework and have replaced its backbone with ShiftFormer-B. For fair comparison, the baseline backbone is ViT-Base [1], pre-trained on ImageNet1k. The experimental results are reported in **Table 4**.

| method | Acc.@1 | FLOPs(M) | Training epochs |
|---|---|---|---|
| **ShiftCNN-S** | **78.7** | **259** | **500** |
| AttentiveNAS-A1 | 78.4 | 279 | $360\times4$=1440 |
| **ShiftCNN-B** | **80.4** | **453** | **500** |
| AttentiveNAS-A4 | 79.8 | 444 | $360\times4$=1440 |

Table 3: Comparison of the effectiveness with AttentiveNAS.

| Backbone | Acc.@1 | FLOPs(G) |
|---|---|---|
| ViT-Base | 48.2 | 120.9 |
| ShiftFormer-B | **49.8** | **90.4** |

Table 4: Comparison of the effectiveness on the segmentaion task.

## 6. Visualization of the Searched Architectures

We show the searched architectures of ShiftNAS family models in **Figure 3**, including ShiftFormer-T, ShiftFormer-S, ShiftFormer-B, ShiftCNN-S and ShiftCNN-B.

## 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. *arXiv preprint arXiv:2010.11929*, 2020. 3

[2] Jing Liu, Jianfei Cai, and Bohan Zhuang. Focusformer: Focusing on what we need via architecture sampler. *arXiv preprint arXiv:2208.10861*, 2022. 2

[3] Dilin Wang, Meng Li, Chengyue Gong, and Vikas Chandra. Attentivenas: Improving neural architecture search via attentive sampling. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6418–6427, 2021. 3

[4] Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas Huang. Slimmable neural networks. In *International Conference on Learning Representations*, 2018. 3

[5] Bowen Zhang, Zhi Tian, Quan Tang, Xiangxiang Chu, Xiaolin Wei, Chunhua Shen, and Yifan Liu. Segvit: Semantic segmentation with plain vision transformers. *arXiv preprint arXiv:2210.05844*, 2022. 3

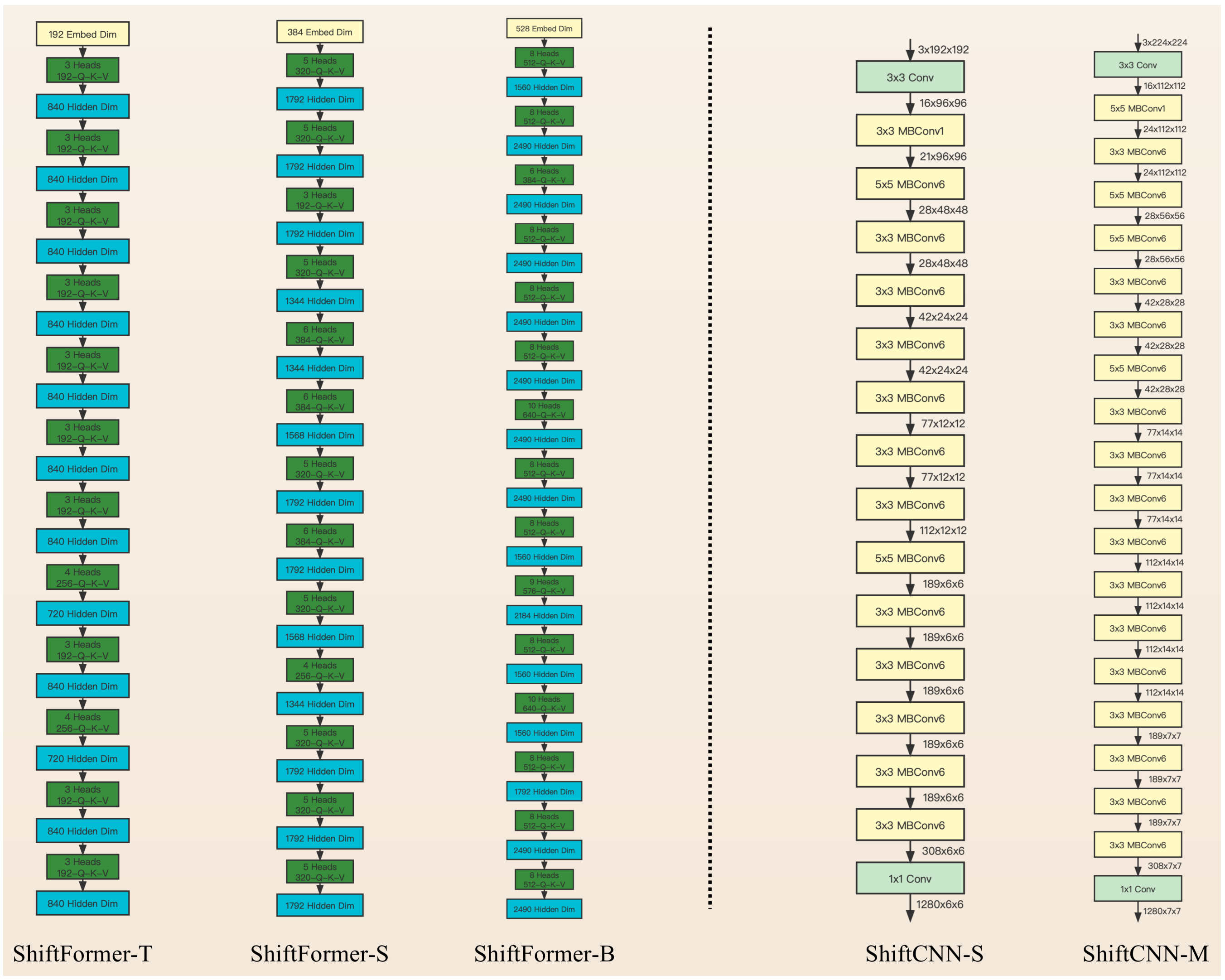

Figure 3: The searched architectures of ShiftNAS family models.