ShrinkNAS : Single-Path One-Shot Operator Exploratory Training for Transformer with Dynamic Space Shrinking

Anonymous ACL submission

Abstract

Neural Architecture Search (NAS) for Transformer has shown its growing capabilities in exploiting the benefits of various Transformer architecture configurations. Recent studies envision the diverse potential of introducing unprecedented Transformer operators (OPs, such as Convolution) to its structure, yet the existing methods of doing so are all time-consuming. Traditionally, Single-Path One-Shot (SPOS) models enable efficient search over a vast set of OPs. However, existing SPOS methods on Transformer focus only on dimensional configurations of the vanilla Transformer OP (e.g., Multi-head Attention), and did not consider introducing other OPs. This paper explores the possibility of including OPs in the Transformer-based SPOS architecture search to discover better Transformer structures with the high efficiency facilitated in the SPOS category. To achieve that, we propose Dynamic Space Shrinking (DSS), a novel method that resolves problems brought from newly added OPs by dynamically keeping the current sample space containing subnets with good configurations and performance. We implemented DSS in ShrinkNAS, the first SPOS one-shot inter-OP model for Transformer. Our evaluation shows that ShrinkNAS is of much higher elasticity by finding a better structure beating the human-designed ones under tight constraint (<10M parameters), while existing intra-OP SPOS methods are not even close.

1 Introduction

Transformer (Vaswani et al., 2017) has been widely applied in modern Neural Language Processing (NLP) tasks. Although being powerful enough, neural architectures derived from Transformer, either human-designed (Devlin et al., 2018; Dai et al., 2019) or discovered by Neural Architecture Search (NAS, Zoph and Le, 2016) (So et al., 2019; Fan et al., 2020), prove that further altering the structure of Transformer is necessary to fully exploit its potential. Among NAS methods for further exploring potential structures of Transformer, the single-path one-shot (SPOS) NAS (Guo et al., 2020) is popular for its efficiency and simplicity. It only requires designing a one-shot model (the **supernet**) containing all possible candidate networks (the **subnets**) with shared weights for their common operators (OPs) in the same layer. Only one subnet (single-path) is sampled for every forward-backward pass to train the supernet. Eventually, subnets are evaluated with inherited weights from the converged supernet to discover the optimal structure. Previous works on SPOS NAS for Transformer (Wang et al., 2020; Chen et al., 2021) have all set up new SOTA on their respective tasks.

Surprisingly, unlike many SPOS NAS studies for convolutional networks that perform both intra-OP search (e.g., the number of output channels or strides in convolutional OPs) and inter-OP search (e.g., replacement of OPs or introducing new OPs) (Guo et al., 2020), recent SPOS NAS studies for Transformer focus solely on intra-OP search while keeping the vanilla Transformer OP (e.g., self-attention) unchanged. They succeeded in exploring intra-OP structures (e.g., the number of attention heads), but failed to take newly designed, promising OPs for Transformer (e.g., Lightweight Convolution, Dynamic Convolution (Wu et al., 2019a) and LSRA (Wu et al., 2019b)) into consideration. The potential of the inter-OP search for Transformer has been proved by many non-SPOS studies, showing a higher elasticity on different scales (So et al., 2019) and better performance on many tasks (Fan et al., 2020), compared to intra-OP methods. However, these non-SPOS studies are highly time-consuming, as they essentially need to train thousands of subnets searched out even with early stopping, sometimes costing 10^6 GPU Hours, while SPOS costs only 10^2 GPU Hours, as it only needs to train one supernet by sharing weights.

In this paper, we explore the possibility of involving inter-OP search in Transformer-based SPOS

NAS to discover a better neural structure with the high efficiency facilitated by the one-shot method. An intuitive way is to extend the supernet with more types of Transformer OPs. However, several problems raised from here and the first is the training step amortization. Precious training opportunities are amortized by the increasing number of candidate subnets, causing good structures underestimated due to insufficient train. What is worse, the structural characteristics of Transformer cause supernets like this almost untrainable. Instability from intra-OP and inter-OP altering at the same time is propagated and amplified layer-by-layer in Transformer's stacked structure, as observed in (Liu et al., 2020). According to our evaluation, output instability will increase by up to 50% if we naively include more OPs in the supernet, which significantly raises the training difficulty and affect the supernet's ability to identify promising subnets.

We believe the root reason for this severe divergence in the naive SPOS method is assigning equal training opportunities to all subnets. This is fundamentally controversial to the fact that good structures are rare in the search space. Our insight is that good structures usually share common characteristics, like good-performing OPs usually show up more frequently, while bad structures show more randomness and arbitrariness. We believe that bad structures are a major cause of divergence for their randomness. If we can locate a smaller sample space containing more promising subnets, we can solve the above two problems together.

Based on the insights, we present Dynamic Space Shrinking (DSS), a novel method that can dynamically evaluate and adjust the sampling space to make sure it contains more promising potential-good subnets, instead of the random and bad ones. This method helps the potential-good structures get more training while train-time divergence is also reduced. To realize the method, we encounter several challenges and embed an abundant bunch of unique solutions.

First, how to identify potential-good subnets with low cost? We identified that subnet performance at the early stage of training serves as a good indicator of its final performance. Therefore, we can adaptively filter over the search space by simply using the current supernet and collecting potential-good subnets by their current performance, forming a good search space for the next stage of training. The evaluation is lightweight as it only requires a forward batch for each subnet.

Second, how to efficiently explore as much search space as possible to reveal deeply covered architectures that do not appear during training? We utilize an iteratively evolutionary filtering process that explores the sampled subnets during train time and promising new subnets that were not sampled before by mutation and crossover on the collected good subnets.

Based on DSS, we perform SPOS with both inter-OP and intra-OP enabled and successfully find new structures that outperform the dedicatedly designed SOTA Lite Transformer (Wu et al., 2019b) under tight mobile constraints (<10M parameter size). We focus on small-scale models because small models are more sensitive to structural variation, therefore, serve as a better metric for comparing structures. In contrast, large models are more likely to be affected by hyperparameters or even initialization (Huang et al., 2020). Besides, it can also demonstrate extra elasticity empowered by inter-OP search. Our structure achieves at most 1.0 higher IWSLT'14 De-En BLEU score than SOTA under the same parameter size. To compare, no previous SPOS method can beat the human-designed SOTA under this constraint. Our contributions are:
(1) To our best knowledge, we are the first to identify and quantify the difficulties of inter-OP SPOS for Transformer. We attribute the previous limitations on achieving inter-OP SPOS to not addressing the increased subnet divergence during training.
(2) We propose DSS, a novel method that can greatly mitigate the supernet convergence difficulties. Our evaluation shows that DSS reduces subnet divergence during inter-OP supernet training by 60%, reaching the same magnitude as the intra-OP ones. Hence, DSS provides higher training stability and enables inter-OP SPOS search on Transformer; DSS takes the first step to realize the potential envisioned in recent works (So et al., 2019).
(3) Our inter-OP SPOS model, ShrinkNAS, shows much better searching results in tight constraints than the readily highly-optimized intra-OP SPOS works. Our evaluation justified that extending search space in inter-OPs can create higher searching elasticity for future Transformer NAS works. Moreover, DSS can be extended to other NAS areas, serving as a good search space filter for locating good search spaces at the early stage of training. Our source code and results are released on `github.com/acl22p2484/shrinknas`.

2 Background

Single-Path One-Shot NAS. Among many novel NAS algorithms, the one-shot NAS (Liu et al., 2020) is popular for its efficiency as well as relatively good accuracy.

We denote the search space containing all potential structures as $\mathcal{A}$. Typically, SPOS method trains a supernet $\mathcal{N}(\mathcal{A}, W)$ containing all potential subnets α_i as paths on it. The optimization target for the supernet is:

$$W_{\mathcal{A}} = \arg\min_{W} \mathcal{L}_{\text{train}}\left(\mathcal{N}(\mathcal{A}, W)\right)$$

To reach the target, SPOS will sample $\alpha \in \mathcal{A}$ at each training step, and perform a normal forward-backward pass to optimize the parameters of its OP on the sampled path. After the supernet is trained, the next target is to find the optimal subnet from the supernet$\alpha^\star$ with weights of its OP inherited from the supernet:

$$\alpha^\star = \arg\max_{\alpha \in \mathcal{A}} \text{Acc}_{val}(\mathcal{N}(\alpha, w))$$

After the two steps, $\alpha^\star$ is the searched structure.

3 Difficulties of Inter-OP SPOS Search on Transformer

In this section we will discuss why the naive inter-OP SPOS method (i.e. simply add more OPs to the supernet) is impractical for Transformer. We breakdown the problem to two factors: amortized training step for new OPs, and training instability from OP switching.

3.1 Expansion of the supernet amortize training steps for subnets

While SPOS is popular for its simplicity, requiring only one subnet to be sampled and trained every time, it's intuitive that adding more choices to each layer will amortize training steps for subnets and OPs. Good structures that don't get enough training steps could be underrated, which lead to sub-optimal searching results.

Weight Entanglement proposed in (Chen et al., 2021) can mitigate this problem in intra-OP search by merging the weights of OPs of same type in a layer together. However, it's not useful in the inter-OP SPOS case, as OPs of different types can't be merged together. Therefore, the amortization problem will still exist if we take the naive uniform sampling method in inter-OP SPOS for Transformer.

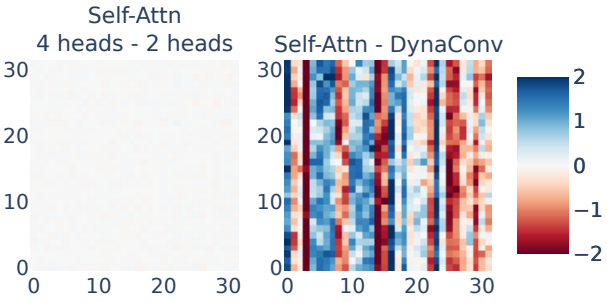


Figure 1: Visualization of module output divergence after normalization. The darker the color, the higher training instability it will cause. left (it is not empty): output divergence of intra-OP substitution (4-heads to 2-heads); right: output divergence of inter-OP substitution (self-attention to dynamic convolution).

3.2 OP switching causes fluctuation of network outputs

(Liu et al., 2020) discussed the difficulty of initializing weights for Transformer, showing that Transformer is sensitive to small weight perturbations. We use a simplified case of their proved theorem to show why inter-OP SPOS brings higher subnet divergence and how it affects training stability.

We refer the normalized output of each Transformer sub-layer (i.e. self-attention, attend-to-encoder attention or feed foward network (FFN)) as $\widehat{\mathbf{a}}_i = \mathcal{G}_i\left(\widehat{\mathbf{x}}_{i-1}, W_i\right)$ where $\widehat{\mathbf{x}}_{i-1}$ is the input of the i-th layer. Given residual connections in Transformer, the final output of a Transformer model is $\widehat{\mathbf{x}} = \mathcal{F}\left(\mathbf{x}_0, W\right) = \sum_{j \le N} \beta_{N,j} \widehat{\mathbf{a}}_j$ where $\beta_{i,j}$ is the layer normalized scaling factor of the j-th output in the i-th layer. We put the star (*) mark on the variable to indicate it's change because of modification. (Liu et al., 2020) proves that, the variance of output change due to modification is:

$$\text{Var}\left[\mathcal{F}\left(\mathbf{x}_0, W\right) - \mathcal{F}^*\left(\mathbf{x}_0, W^*\right)\right] \approx \sum_{i=1}^{N} \beta_{i,i}^2 \text{Var}[\mathcal{G}_i\left(\widehat{\mathbf{x}}^*_{i-1}, W_i\right) - \mathcal{G}^*_i\left(\widehat{\mathbf{x}}^*_{i-1}, W^*_i\right)]. \quad (1)$$

This indicates that effects of modification on layers will get propagated and aggregated at the model output due to residual connection and stacked structure in Transformer. With the same input ($\mathbf{x}_0$), output differs dramatically with each other on each run, which will for sure make the training unstable. Figure 1 visualizes the high divergence of OP output ($\mathcal{G}_i\left(\widehat{\mathbf{x}}^*_{i-1}, W_i\right) - \mathcal{G}^*_i\left(\widehat{\mathbf{x}}^*_{i-1}, W^*_i\right)$) in intra-OP search comparing to inter-OP search. Therefore, uniform sampling that is effective in intra-OP

SPOS for Transformer does not apply to inter-OP any more.

Previous techniques that help stabilize Transformer training, like pre-layer normalization (Xiong et al., 2020) and Adam (Kingma and Ba, 2014) is to a extent still effective. For example, pre-layer normalization reduces $\beta_{i,i}$ in eq 1 comparing to post-layer normalization, the normal practice (Liu et al., 2020), but it is not enough for the higher magnitude of divergence. Adam, on the other hand, provides high stability at the early stage of training where different OPs may hold common optimization target. But when the target of optimization diverges, i.e., gradients do not share a common direction at each step any more, Adam won't be effective either. In a word, previous techniques are helpful at mitigating training instability in inter-OP SPOS, but they don't resolve the root cause, i.e., the subnet divergence.

4 ShrinkNAS with DSS

In this section, we will first illustrate and explain DSS, the core strategy we propose to cope with the problems in the last section. Then we will demonstrate how the whole searching process works for ShrinkNAS, the inter-OP SPOS search for Transformer enabled by DSS. ShrinkNAS includes three components: (a)An inter-OP Supernet (b)A subnet validator and (c)A evolutionary subnet generator and sampler. An overview of ShrinkNAS is shown in Figure 2.

4.1 Overview of DSS

DSS is able to solve the training step amortization and subnet divergence problem together with two underlying techniques: **Train-time Sample Space Shrinking** helps reduce the divergence, gradually driving the sample space from big and random to relatively small but accurate, containing promising potential-good subnets. **Evolutionary Exploration** helps maintain a large space for structure search and exploration, and make the search more thorough and accurate. With these two techniques combined, we are able to reduce the divergence in sample space while give the promising structures more training opportunities.

4.2 Train-time Sample Space Shrinking

It's commonly believed that good-performing structures are rare in the whole sample space, especially when we further enlarge it with inter-OP search. Also, they should share some common characteristics, which also serves as the underlying guidance for human researchers to design new network structures. Therefore, most of the train-time divergence actually comes from the bad, random structures in stead of new good structures added when we extend the search space.

Ideally, if we let the supernet trains only on the subset of the training space that contains good structures, we will be able to reduce the divergence while evaluate thoroughly over potential-good subnets. To formalize the idea, we let $\mathcal{A}_{good} \subset \mathcal{A}$ to represent the set of globally good-performing structures, the ideal sample space we are looking for. Then the ideal training process can be formalized as:

$$W_{\mathcal{A}} = \arg\min_{W} \mathcal{L}_{\text{train}} \left(\mathcal{N}(\mathcal{A}_{good}, W)\right)$$

However, we don't have accurate *a priori knowledge* on $\mathcal{A}_{good}$. The best we can get is finding out a estimation of $\mathcal{A}_{good}$, the $\hat{\mathcal{A}}_{good}$. How to find an accurate $\hat{\mathcal{A}}_{good}$ during train time at a low cost?

It's a widely used and evaluated assumption that structures being able to perform well in the early stage of training is likely to be good structures globally. Many previous works (So et al., 2019; You et al., 2020; Hu et al., 2021) take advantage of this idea to reduce their training cost. Simply extending the idea, we assume that structures perform well in any stage of training are also more likely to be good eventually. To formalize the idea, $\mathcal{A}_{i\_good} \subset \mathcal{A}$ denotes structures performing good in training stage i. Then the above assumption can be described as: $P(\alpha \in \mathcal{A}_{good} | \alpha \in \mathcal{A}_{i\_good}) > P(\alpha \in \mathcal{A}_{good} | \alpha \notin \mathcal{A}_{i\_good})$.

Moreover, $\mathcal{A}_{i\_good}$ converge in probability to $\mathcal{A}_{good}$. As the training goes on, the supernet can give more accurate evaluation for the subnets, and its target is to be able to distinguish good subnets from bad ones, i.e.:

$$\forall \alpha \in \mathcal{A}_{good}, \lim_{n \to \infty} P(\alpha \notin \mathcal{A}_{n\_good}) = 0$$

Therefore, $\mathcal{A}_{i\_good}$ serves as a good estimation of $\mathcal{A}_{good}$ for its growing accuracy. Following the theory, space shrinking is as simple as keeping the set of subnets that's currently performing well as the sample space for the supernet in the next stage, $\mathcal{N}(\mathcal{A}_{i\_good}, W)$. In that way, we **shrink** the sample space from $\mathcal{A}$ to $\mathcal{A}_{i\_good}$.

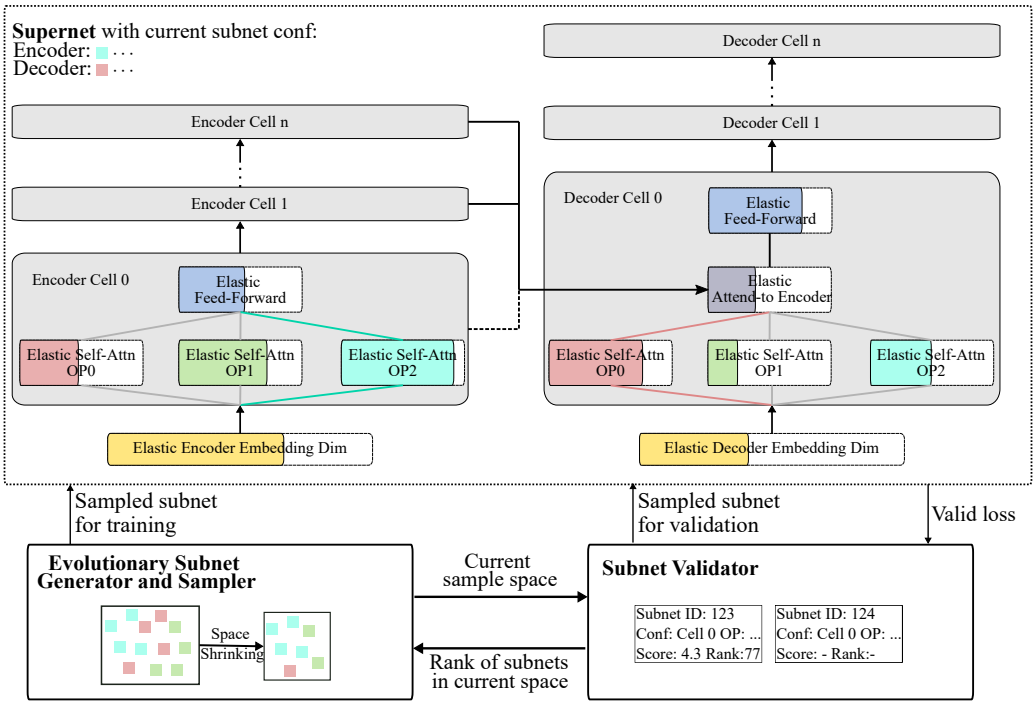


Figure 2: Overview of ShrinkNAS. At the training stage, ShrinkNAS samples subnets from the current sample space stored in subnet sampler. At the shrinking stage, a iterative evolutionary subnet search is performed based on the current sample space to find out a smaller space containing more potential-good structures.

4.3 Evolutionary Exploration

Taking $\mathcal{A}_{i\_good}$ as the $\hat{\mathcal{A}}_{good}$ has been proved to be an accurate estimation. However, the accurate $\mathcal{A}_{i\_good}$ is still not available for it involves too much cost for validating every subnet. During the train time, only a subset, $\mathcal{A}^*_{i\_good} \subset \mathcal{A}_{i\_good}$ is practically available. Therefore in practice, we are only able to keep a portion of subnets with high ranks we sampled for validation as the $\mathcal{A}^*_{i\_good}$ to shrink to. This raises a challenge on the accuracy of $\mathcal{A}^*_{i\_good}$.

What's more, as the ranking accuracy of the supernet is not high enough before converging, promising structures may be dropped at the early stage of training. We need to not only keep the good-performing subnets at the current stage but also find subnets with good *potential*, i.e. likely to perform well after more training.

To cope with the above dual requirements on both accuracy and depth, we introduce the evolutionary exploration method. An evolutionary exploration containing the following steps: First, we collect the sample subnets from the last training stage as the seed. Then we evaluate and rank the subnets according to their validation scores, keeping only the top k subnets. Then two operations are performed to supplement the sample space to its original size: For **crossover**, we sample 2 subnets from the top k and generate a new subnet by crossing the two parent subnets. For **mutation**, we sample a subnet and randomly change some of its components to other available choices by a pre-defined probability. We iteratively perform the above steps till reaching the designated iterations. Algorithm 1 shows the process of our evolutionary exploration.

The reason we choose the evolutionary method is two-fold: On the one hand, comparing to random sampling, the evolutionary method is proved to be more efficient to search for good subnets, and thus is a standard practice in the search stage of SPOS works (Guo et al., 2020). On the other hand, based on the previous insight on the distribution of good subnets, the evolutionary method is also capable of finding the subnets with good potential by slightly modifying the current good structures. Therefore, it reaches both requirements on accuracy and depth, and can generate a good sample space $\mathcal{A}^*_{i\_good}$ for the next stage of training.

4.4 Training and Searching Pipeline for ShrinkNAS

After introducing the process of DSS, we can now illustrate the training and searching pipeline for ShrinkNAS. During the train time, ShrinkNAS in-

Algorithm 1 Evolutionary Exploration for Sample Space Shrinking

Require: n: the target sample size;
$A^*_{i-1\_good}$: the current sampled subnets;
i: the amount of iterations;
p: ratio of good subnets to keep;
m: ratio of subnets geterated by mutation;
c: ratio of subnets generated by crossover;
Ensure: $p + m + c = 1.0$
$A^*_{i\_good} \leftarrow A^*_{i-1\_good}$
while $i \neq 0$ **do**
  $i \leftarrow i - 1$
  A_g = EVAL_SORTED($A^*_{i\_good}$)[:$p \times n$]
  A_m = MUTATION(A_g, $m \times n$)
  A_c = CROSSOVER(A_g, $c \times n$)
  $A^*_{i\_good} \leftarrow A_g + A_m + A_c$
end while
return $A^*_{i\_good}$

volves an iterative two-phased training process:

Phase 1: Given a sample space $\mathcal{A}$, we uniformly sample a subnet $\alpha_i \in \mathcal{A}$ and perform normal forward-backward pass to update its corresponding weights ω_i in the supernet $\mathcal{N}(\mathcal{A}_{good}, W)$. This phase is similar to other SPOS methods, but the $\mathcal{A}$ will be adaptively adjusted in the next phase.

When the training steps in phase 1 reach a pre-defined threshold, it will switch to phase 2 and performs sample space shrinking.

Phase 2: When the training reaches a pre-defined step n, Given a sample space $\mathcal{A}^*$ containing all the sampled subnets in phase 1, we perform the evolutionary exploration algorithm (Algorithm 1 on it to shrink our current sample space to the new one.

After phase 2, a new sample space is generated for the next stage of training, and the process goes back to phase 1. Following the alternate two phases, the training process goes on till it reaches the maximum training step.

We then perform the standard evolutionary search over the supernet to find the optimal structure. The searching process can involve extra penalties or constraints that guide the searching process towards a desired scope or follow a preset constraint. The evolutionary search process is similar to our evolutionary exploration, except for after the iterations, we return the best subnet with the highest score instead of a new search space.

5 Experiments

5.1 Dataset

We conduct experiments on the IWSLT'14 De-En Machine Translation Dataset. We follow the settings in (Wu et al., 2019b) for fair comparison and preprocess the dataset with 10K joint byte pair splitting (BPE) (Sennrich et al., 2016), splitting it into around 160K/7K/7K sentences for train/valid/test. The vocabulary size for German is 8848 while 6632 for English.

5.2 Baselines and setups

The human-designed baselines include the vanilla Transformer (Vaswani et al., 2017) and two modified Transformers with new substitutions for the self-attention operator: Lightweight/Dynamic Convolutions (Wu et al., 2019a) and Long-Short Range Attentions (LSRA) (Wu et al., 2019b). The NAS baselines includeHardware Aware Transformer (Wang et al., 2020). Following (Wu et al., 2019b), we set a 10M parameter size constraint for all human-designed models. For evaluation, following (Wang et al., 2020), we use beam 5 for sequence generation and use case-sensitive tokenized BLEU [1] to calculate the BLEU scores.

5.3 Implementation Details

Search Space Setups. Our supernet is heterogeneous and configurations can be different among cells. We have [160, 240, 320] for embedding dim, [1, 1.5, 2] for dimension expansion ratio of FFN (e.g., for a embedding dim of 160 and a expansion ratio of 1.5, FFN will raise the dimension to $160 \times 1.5 = 240$ and reduce back to 240), [2, 4] for head number in self-attention OPs and [1, 2, 3, 4, 5, 6] for decoder cell number. We also support the same arbitrary attend-to-encoder connection between encoder cell outputs and decoder's attend-to-encoder attentions in (Wang et al., 2020). For search space for the OPs, we support substitution of self-attention OPs and activation functions of FFNs. Candidates of self-attention OPs include: the vanilla self-attention (Vaswani et al., 2017), the lightweight convolution and dynamic convolution (Wu et al., 2019a) and the LSRA with lightweight/dynamic convolution (Wu et al., 2019b). Candidates of activation functions include: ReLU (Nair and Hinton, 2010), Leaky ReLU (Maas et al., 2013) and Swish (Ramachandran et al., 2017).

[1] https://github.com/moses-smt/mosesdecoder

Validator Setups. For the validator, we use validation batch size of 30,720 and validate each subnet with a single forward pass. Different sampling seeds are generated each validation run to avoid the sample space getting overfitted to a specific subset of the validation set. The validator returns the sum of validation loss and other score penalties as the final loss for a subnet.

Evolutionary Subnet Generator Setups. For the evolutionary subnet generator, we use a mutation rate of 0.4 to better explore the extended search space. At each iteration of the search, we keep the top 30% subnets according to their ranks in validation loss as the parents, then generate 40% of the original population with mutation and the remaining 30% with crossover to restore the original population and perform the next search iteration.

Training Settings. The hyperparameter settings are similar with (Wang et al., 2020) and (Wu et al., 2019b). We train the supernet for 50K steps with inverse square root LR scheduler. The LR grows linearly to $5e-4$ in the first 10,000 warm-up steps and then square-root annealed. The batch size for training is 4096 with no gradient aggregation. After training, the searched subnet inherits its weight from the supernet and continues to fine-tune for another 30k steps.

We perform DSS every 10 epochs for 5 times, and for the last time, we take the first subnet (with the lowest loss so far) of the returned subnets as the searched subnet. We report the score of the selected subnet after fine-tuning instead of retraining. We will discuss fine-tuning and retraining in the ablation study section.

6 Analysis and Discussion

In this section, we will analyze the final results and perform ablation study on important factors we propose in this paper.

6.1 Result Analysis

The final results of our proposed method are shown in Table 2. As is shown, our proposed method outperforms the previous state-of-the-art human-designed structure in both BLEU scores and the size of parameters. We are able to push the SOTA forward for at most 1.0 BLEU scores under the same parameter constraints or gain 22.1% parameter size reduction in exchange for a mere 0.1 BLEU score drop. Please note that we have chosen the tightest baselines, the Lite Transformer that has already been optimized for small parameter sizes to compare. Yet, ShrinkNAS still helps further optimize the structure of Transformer under this lightweight constraint.

Model	Search Type	Time Cost (h)
HAT	intra-OP	3.4
RankNAS	intra-OP*	2.3
Ours	**inter-OP**	3.2
Lite Trans.	Not applicable	2.6

Table 1: Search and retrain (fine-tuning) time costs for different NAS Models on IWSLT'14 De-En. $\star$ indicates RankNAS is a intra-OP method but doesn't involve training a supernet. All costs are normalized in RTX3090 GPU Hours. We list Lite Transformer's training cost for 80k steps for reference.

Compared with other NAS methods in Table 1, except for consistent better performance, ShrinkNAS is also capable of providing end-to-end structure search and training pipeline. To compare, training a model with the same amount of steps costs only 0.6 hours less, while with the extra 0.6 hours, ShrinkNAS can explore the inter-OP search space containing 10^{30} combinations, find a better structure and fully train it. For other NAS models for Transformer, a retrain is usually required for optimal performance. This helps ShrinkNAS save at least 20% GPU Hours compared to intra-OP methods. What's more, ShrinkNAS provides a much larger search space with inter-OP search, while other methods for comparison are all intra-OP. This indicates our method is able to explore the larger search space with even less search time, thus achieves higher exploration efficiency. Even compared to the recent SOTA, RankNAS (Hu et al., 2021), which dramatically reduces the search time, our method still achieving a comparable cost with it for RankNAS requires retraining as it doesn't involve training a one-shot supernet.

6.2 Ablation Study

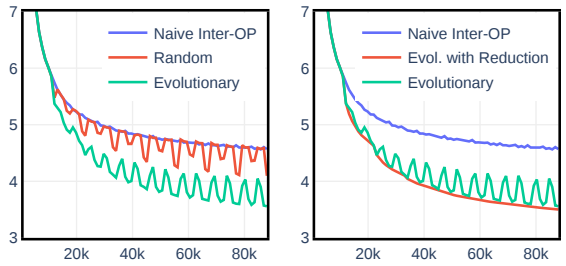
In this part, we will empirically evaluate the claims we make in this paper. We will also evaluate the two techniques proposed in DSS.

Reduction of Subnet Divergence. Figure 4 shows the distribution of subnet divergence before and after our sample space shrinking. We quantify subnet divergence in the same way as the analysis in Section 3, i.e. the variance of model output with the same input. We randomly sample 200 subnets each in the sample space before and after shrinking.

Model	#Parameters	BLEU	Δ BLEU
Transformer (Vaswani et al., 2017)	2.8M (1.8M)‡	27.8	-
LightConv (Wu et al., 2019a)	2.5M (1.5M)	28.5	+0.7
Lite Transformer (Wu et al., 2019b)	2.8M (1.8M)	30.9	+3.1
ShrinkNAS (Ours)	1.8M	**31.9**	**+4.1**
Transformer	5.7M (4.1M)	31.3	-
LightConv	5.1M (3.5M)	31.6	+0.3
Lite Transformer	5.4M (3.9M)	32.9	+1.6
ShrinkNAS (Ours)	**3.6M**	**33.3**	**+1.9**
Transformer	8.5M (6.4M)	32.7	-
LightConv	8.4M (6.3M)	32.9	+0.2
Lite Transformer	8.9M (6.8M)	33.6	+0.9
ShrinkNAS (Ours)	**5.9M**	**33.8**	**+1.1**
HAT (Wang et al., 2020)	16.8M	33.4	-

Table 2: Results on IWSLT'14 De-En. Our model (ShrinkNAS) outperforms both Lite Transformer, the human-designed SOTA and HAT, the NAS-Searched SOTA in comparable #Parameters ranges. Performance of the three baselines comes from (Wu et al., 2019b). ‡ indicates we find a flaw about parameter size in their paper, but we faithfully cite the original data and put the correct data in the parentheses

To serve as a baseline, we also sample 200 subnets from the intra-OP supernet. Naive inter-OP SPOS will on average have 2x divergence than intra-OP methods. While through an iteration of DSS, it can be reduced by up to 40%.



(a) Evolutionary exploration (b) Space reduction

Figure 3: The average training losses in different experiment setups. "Evolutionary" (green in a, b) and "Evol. with Reduction" (red in b) are DSS's results. Applying techniques including evolutionary exploration and space reduction can significantly improve training convergence.

Effects of Evolutionary Exploration. Figure 3(a) shows the effect of evolutionary exploration. Although shrinking to the top 5,000 subnets from the initially sampled subnets does help the training process get closer to convergence, adding iteratively evolutionary exploration to the shrinking process significantly boosts up the training and eventually lets the supernet converge. We can claim that evolutionary exploration help locates better sample space that contains good structures.

Effects of Space Reduction Figure 3(b) shows the effect of the Space Reduction, a variant of DSS. Space reduction keeps only the top subnets returned from evolutionary search as the next training sample space, showing higher stability than DSS. This proves the potential of further extending DSS.

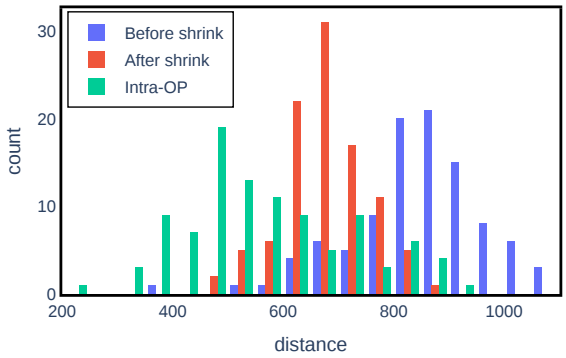


Figure 4: Distribution of subnet divergence before and after the first shrinking process. The left (lower distance between output feature maps) is better, indicating less subnet divergence in the supernet training, therefore easier to train. Different distributions between the red and blue are DSS's outcome.

7 Conclusion

In this paper, we present Dynamic Space Shrinking, a novel method to cope with the training step amortization and subnet divergence problems in inter-OP SPOS NAS by locating a smaller sample space containing promising structures. Enabled by DSS, we design the first inter-OP SPOS model for Transformer that efficiently finds network structures outperforming the human-designed SOTA model while no previous intra-OP method managed to do so. In the future, DSS can be extended to other fields of NAS search and empowering searching in larger spaces. Our source code and results are released on `github.com/acl22p2484/shrinknas`.

References

Minghao Chen, Houwen Peng, Jianlong Fu, and Haibin Ling. 2021. Autoformer: Searching transformers for visual recognition. *arXiv preprint arXiv:2107.00651*.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G Carbonell, Quoc Le, and Ruslan Salakhutdinov. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Yang Fan, Fei Tian, Yingce Xia, Tao Qin, Xiang-Yang Li, and Tie-Yan Liu. 2020. Searching better architectures for neural machine translation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28:1574–1585.

Zichao Guo, Xiangyu Zhang, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, and Jian Sun. 2020. Single path one-shot neural architecture search with uniform sampling. In *European Conference on Computer Vision*, pages 544–560. Springer.

Chi Hu, Chenglong Wang, Xiangnan Ma, Xia Meng, Yinqiao Li, Tong Xiao, Jingbo Zhu, and Changliang Li. 2021. RankNAS: Efficient neural architecture search by pairwise ranking. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 2469–2480, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Xiao Shi Huang, Felipe Perez, Jimmy Ba, and Maksims Volkovs. 2020. Improving transformer optimization through better initialization. In *International Conference on Machine Learning*, pages 4475–4483. PMLR.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Liyuan Liu, Xiaodong Liu, Jianfeng Gao, Weizhu Chen, and Jiawei Han. 2020. Understanding the difficulty of training transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5747–5763.

Andrew L Maas, Awni Y Hannun, Andrew Y Ng, et al. 2013. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3. Citeseer.

Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Icml*.

Prajit Ramachandran, Barret Zoph, and Quoc V. Le. 2017. Searching for activation functions.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725.

David So, Quoc Le, and Chen Liang. 2019. The evolved transformer. In *International Conference on Machine Learning*, pages 5877–5886. PMLR.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Hanrui Wang, Zhanghao Wu, Zhijian Liu, Han Cai, Ligeng Zhu, Chuang Gan, and Song Han. 2020. Hat: Hardware-aware transformers for efficient natural language processing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7675–7688.

Felix Wu, Angela Fan, Alexei Baevski, Yann Dauphin, and Michael Auli. 2019a. Pay less attention with lightweight and dynamic convolutions. In *International Conference on Learning Representations*.

Zhanghao Wu, Zhijian Liu, Ji Lin, Yujun Lin, and Song Han. 2019b. Lite transformer with long-short range attention. In *International Conference on Learning Representations*.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tieyan Liu. 2020. On layer normalization in the transformer architecture. In *International Conference on Machine Learning*, pages 10524–10533. PMLR.

Shan You, Tao Huang, Mingmin Yang, Fei Wang, Chen Qian, and Changshui Zhang. 2020. Greedynas: Towards fast one-shot nas with greedy supernet. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1999–2008.

Barret Zoph and Quoc V Le. 2016. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*.

A Appendix

We release the source code and evaluation results on `github.com/acl22p2484/shrinknas`