TAS-LoRA: Transformer Architecture Search with Mixture-of-LoRA Experts - Supplementary -

In this supplementary material, we provide more detailed analyses on the design of TAS-LoRA (Sec. 1), additional experiments for searching hyperparameters (Sec. 2), and limitations of our work and future directions (Sec. 3). We then summarize the training, and search procedures of TAS-LoRA (Sec. 4). Finally, we provide the computational resources used in our experiments (Sec. 5).

1. Analyses on design of TAS-LoRA

In this section, we provide a more detailed analysis of the design choices in TAS-LoRA.

1.1. Router architecture

We show in Table 1 the results of different router architectures in the AutoFormer-T search space. In all cases, the router takes as input the sequence of architectural attributes of the sampled subnet (e.g., MLP ratio, number of attention heads, and embedding dimension). The router then predicts routing weights for the MoLE experts for each block. We compare recurrent routers, which can capture sequential dependencies across blocks, with a non-recurrent MLP baseline that treats each block independently. We can see that recurrent routers, i.e., LSTM [1] and GRU [2], perform similarly, while the block-wise MLP performs worse. This indicates that modeling dependencies across transformer blocks is beneficial for expert routing.

1.2. MoLE

We show in Table 2 the comparison between using a single shared LoRA module for all subnets and the proposed MoLE. For a fair comparison, both variants are inserted into the same target layers, and are trained under the same search and optimization setting. We can see that replacing MoLE with a single shared LoRA degrades the performance from 75.7 to 75.0. This is because, since a single LoRA is shared across all subnets, which still results in the feature collapse problem. Our MoLE mitigates this issue by maintaining multiple experts and using the router to assign different experts to different subnets based on their architectural attributes, allowing for more specialized adaptation.

Table 1. Comparison of different router architectures in the AutoFormer-T search space. All routers use the same routing input and are trained with the same optimization setting.

Router	Top-1 (%)
LSTM	75.7
GRU	75.6
Block-wise MLP	75.4

Table 2. Comparison between a single shared LoRA and the proposed mixture-of-LoRA experts in the AutoFormer-T search space.

Method	Top-1 (%)
Single LoRA	75.0
TAS-LoRA (MoLE)	75.7

Table 3. Comparison of different grouping strategies for router initialization in the AutoFormer-T search space.

Grouping strategy	Top-1 (%)
Architectural similarity	75.7
Random grouping	75.5
Number of parameters	75.6

1.3. Group-wise initialization strategy

We show in Table 3 the results of different grouping strategies used in router initialization. In TAS-LoRA, we group blocks by architecture similarities, and assign each group to a different expert during the early stage of training. We compare this strategy with random grouping and grouping by the number of parameters. We can see that grouping by architectural similarity achieves the best performance. This is because grouping by architectural similarity reduces redundancy among experts and encourages diverse feature learning across experts early in training.

1.4. Architectural attributes for routing

We show in Table 4 the ablation results for choosing the architectural attributes used by the router. In TAS-LoRA, the router predicts expert weights for each block based on

Table 4. Ablation on the architectural attributes used for routing in the AutoFormer-T search space.

MLP ratio	#Head	Embed. dim.	Depth	Top-1 (%)
✓	✓	✓	✓	75.8
✓	✓	✓		75.7
	✓	✓		75.5
	✓			75.2

the architectural properties of the sampled subnet. To this end, we provide the router with block-level and subnet-level attributes, including the MLP ratio, the number of attention heads, the embedding dimension, and the depth index. Using a richer set of attributes allows the router to distinguish candidate subnets more precisely, but it also increases the number of distinct groups considered in the initialization stage and may require a larger number of experts.

In this ablation, we incrementally vary the set of routing attributes while keeping the rest of the framework unchanged. We first consider the full set of attributes, and then remove some of them to study whether a more compact routing input can still preserve the effectiveness of subnet-specific expert assignment.

We can see that using richer architectural attributes generally improves performance, while using only a limited subset leads to degraded results. In particular, using only the number of heads, or the number of heads together with the embedding dimension, does not provide sufficient information for the router to distinguish subnets effectively. We also observe that adding depth gives only a marginal gain over using the MLP ratio, number of heads, and embedding dimension. Since including depth increases the number of distinct groups in the initialization stage and thus enlarges the expert space, we exclude depth in the main experiments and use the remaining attributes as the default routing input.

2. Hyperparameters

In this section, we investigate the effect of several hyperparameters: (1) LoRA rank, (2) the β value in router initialization, (3) learning rates for MoLE and the router, and (4) the number of warm-up epochs. Notably, TAS-LoRA consistently outperforms AutoFormer [3] across all settings, demonstrating its effectiveness.

LoRA rank. We present in Table 5 the results of TAS-LoRA with varying LoRA rank r. We observe that TAS-LoRA achieves the best performance when the rank is set to 8 across all search spaces. We attribute this to the following: (1) a small rank may not provide sufficient capacity for capturing subnet-specific features, while (2) a large rank increases the number of trainable parameters, which can lead to overfitting. We also compare in Table 5 the number of trainable parameters of TAS-LoRA to that of full fine-tuning. We can see that when r is set to 2, 4, or 8, TAS-LoRA requires fewer parameters than full fine-tuning, while with $r = 16$, the parameter count exceeds that of full fine-tuning. Considering this trade-off between performance and parameter efficiency, we set the rank to 8 in all experiments.

Table 5. Top-1 accuracy and parameter ratio across LoRA ranks in AutoFormer-T/S/B on ImageNet [4].

Rank	AutoFormer-T		AutoFormer-S		AutoFormer-B	
	Acc. (%)	Params. (%)	Acc. (%)	Params. (%)	Acc. (%)	Params. (%)
2	75.5	23	81.7	32	82.4	11
4	75.6	35	81.8	48	82.5	18
8	75.7	59	81.9	79	82.6	33
16	75.6	107	81.7	142	82.5	63

Table 6. Top-1 accuracy of TAS-LoRA in AutoFormer-T with varying β values for router initialization.

β	1	3	10
Top-1 acc. (%)	75.5	75.7	75.2

β in router initialization. We present in Table 6 the results of TAS-LoRA with different β values in router initialization. We can see that TAS-LoRA achieves the best performance when β is set to 3. If β is too small, the router may fail to promote sufficient specialization among experts. Conversely, if β is too large, it may reduce the flexibility of expert selection, limiting the diversity of experts assigned to subnets. Based on this, we set $\beta = 3$ in all experiments.

Learning rates We show in Table 7 the results of TAS-LoRA with different learning rates for the MoLE and router. We observe that the best performance is obtained when the learning rate is set to 5e-4 for the MoLE and 1e-1 for the router, and use them in all experiments.

Warm-up epochs We present in Table 8 the results of TAS-LoRA with different numbers of warm-up epochs. TAS-LoRA achieves the best performance when the number of warm-up epochs is set to 5. A shorter warm-up limits the time for the experts to specialize in their assigned groups, while a longer warm-up may hinder the joint optimization of the router and MoLE. Based on the results, we set the number of warm-up epochs to 5 in all experiments.

3. Limitations and future work

The number of LoRA experts in TAS-LoRA is directly tied to the number of groups, which corresponds to the number of candidate values for each architectural attribute in the search

Table 7. Top-1 accuracy across learning rate combinations for MoLE (η_L) and router (η_R) in AutoFormer-T.

η_L \ η_R	2e-1	1e-1	5e-2
1e-3	75.7	75.6	75.7
5e-4	75.5	75.7	75.5
1e-4	75.3	75.3	75.2

Table 8. Top-1 accuracy of TAS-LoRA in AutoFormer-T with varying numbers of warm-up epochs.

Warm-up epochs	0	3	5	10
Top-1 acc. (%)	75.5	75.6	75.7	75.6

space. For example, if the number of attention heads varies across 4 candidate values, this results in 4 groups for that attribute. In large-scale search spaces with many attributes and extensive candidate sets, the total number of groups, and thus the number of experts can increase significantly, which may raise computational overhead. To address this, we plan to explore clustering techniques that merge similar candidates within each attribute (e.g., clustering attention head sizes with comparable capacities), thereby reducing the number of groups while ensuring subnet-specific feature learning.

In the current implementation, the number of experts and the rank of the LoRA modules are uniformly set across all layers. Recent studies on LoRA [5, 6] suggest that adaptively assigning ranks and expert counts based on the importance of each layer can further enhance performance. As such, future work will explore dynamic expert allocation and rank adaptation tailored to individual layers to improve both efficiency and accuracy.

4. Algorithms

We summarize in Algorithms 1 and 2 the training and search procedures of TAS-LoRA, respectively. For searching procedure, we use the evolutionary search algorithm, following the same procedure as in [3]. Specifically, we sample 10,000 training examples as the validation set. We set the population size P to 50, and number of iterations T to 20. For each iteration, we pick top-10 subnets based on the validation loss, and generate the child networks by mutating and crossing over the top-10 subnets, with probabilities of 0.2, and 0.4, respectively.

5. Computational resources

We conduct all experiments on NVIDIA A5000 and A6000 GPUs. For the AutoFormer-T space, we use 4 A5000 GPUs, while for AutoFormer-S and AutoFormer-B, we utilize 8 A5000 and 8 A6000 GPUs, respectively. During the searching process, we employ a single A5000 GPU. For transfer learning experiments, we use 4 A5000 GPUs.

Algorithm 1 Training of TAS-LoRA

Input: Pretrained supernet weights W, LoRA experts E, Router U, training dataset D_{train}, max iteration T, number of transformer blocks B, number of layers L
Initialize LoRA experts E and router U
Load pretrained supernet weights W
for $t = 1$ to T **do**
  Sample a mini-batch from D_{train} and a subnet architecture $\mathcal{N}_i$ from the supernet
  Extract subnet-level attributes (e, v) and block-level attributes $\{(n^{(b)}, m^{(b)})\}_{b=1}^{v}$
  Compute expert weights P with router U
  for each layer $l = 1$ to L **do**
    Merge weights: $W^l_{\text{merged}} = W^l + \sum_{k=1}^{K} p^l_k E^l_k$
    Compute output: $y^l = W^l_{\text{merged}} . x^l$
  end for
  Compute loss $\mathcal{L}_{\text{train}}$
  Update E and U using gradient descent on $\mathcal{L}_{\text{train}}$
end for
Return: Trained LoRA experts and router

Algorithm 2 Searching of TAS-LoRA

Input: Pretrained supernet weights θ, LoRA and router parameters ϕ, population size P, architecture constraints C, number of iterations T, validation dataset D_{val}, search space $\mathcal{N}$
Initialize population P_0 with P random subnets satisfying C
for $i = 1$ to T **do**
  $\mathcal{L}_{\text{val}} \leftarrow \emptyset$
  for each subnet $\mathcal{N}_s$ in population P_{i-1} **do**
    if $\mathcal{N}_s$ satisfies constraints C **then**
      Evaluate validation loss $\mathcal{L}_{\text{val}}(\mathcal{N}_s; \theta, \phi)$ on D_{val}
      Store $\mathcal{L}_{\text{val}}[\mathcal{N}_s] = \mathcal{L}_{\text{val}}(\mathcal{N}_s; \theta, \phi)$
    end if
  end for
  Select top-k subnets: $Top_k = \text{Select\_Topk}(P_{i-1}, \mathcal{L}_{\text{val}}, k)$
  Generate mutated subnets: $P_{\text{mutation}} = \text{Mutation}(Top_k, C)$
  Generate crossover subnets: $P_{\text{crossover}} = \text{Crossover}(Top_k, C)$
  Create next population $P_i = P_{\text{mutation}} \cup P_{\text{crossover}}$
  Fill remaining slots with random subnets: $P_i = P_i \cup \text{Random\_Candidates}(P - |P_i|, \mathcal{N}, C)$
end for
$\mathcal{N}^* = \text{Select\_Topk}(P_T, \mathcal{L}_{\text{val}}, 1)$
Return: Optimal subnet $\mathcal{N}^*$

References

[1] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 1997. 1

[2] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *NeurIPSW*, 2014. 1

[3] Minghao Chen, Houwen Peng, Jianlong Fu, and Haibin Ling. AutoFormer: Searching transformers for visual recognition. In *ICCV*, pages 12270–12280, 2021. 2, 3

[4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *CVPR*, pages 248–255, 2009. 2

[5] Qingru Zhang, Minshuo Chen, Alexander Bukharin, Nikos Karampatziakis, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adalora: Adaptive budget allocation for parameter-efficient fine-tuning. *ICLR*, 2023. 3

[6] Chongyang Gao, Kezhen Chen, Jinmeng Rao, Baochen Sun, Ruibo Liu, Daiyi Peng, Yawen Zhang, Xiaoyuan Guo, Jie Yang, and VS Subrahmanian. Higher layers need more lora experts. *arXiv preprint arXiv:2402.08562*, 2024. 3