Analyzing and Mitigating Interference in Neural Architecture Search

Jin Xu [1] **Xu Tan** [2] **Kaitao Song** [2] **Renqian Luo** [1] **Yichong Leng** [2 3] **Tao Qin** [2] **Tie-Yan Liu** [2] **Jian Li** [1]

Abstract

Weight sharing is a popular approach to reduce the cost of neural architecture search (NAS) by reusing the weights of shared operators from previously trained child models. However, the rank correlation between the estimated accuracy and ground truth accuracy of those child models is low due to the interference among different child models caused by weight sharing. In this paper, we investigate the interference issue by sampling different child models and calculating the gradient similarity of shared operators, and observe: 1) the interference on a shared operator between two child models is positively correlated with the number of different operators; 2) the interference is smaller when the inputs and outputs of the shared operator are more similar. Inspired by these two observations, we propose two approaches to mitigate the interference: 1) MAGIC-T: rather than randomly sampling child models for optimization, we propose a gradual modification scheme by modifying one operator between adjacent optimization steps to minimize the interference on the shared operators; 2) MAGIC-A: forcing the inputs and outputs of the operator across all child models to be similar to reduce the interference. Experiments on a BERT search space verify that mitigating interference via each of our proposed methods improves the rank correlation of super-net and combining both methods can achieve better results. Our discovered architecture outperforms $\text{RoBERTa}_{\text{base}}$ by 1.1 and 0.6 points and $\text{ELECTRA}_{\text{base}}$ by 1.6 and 1.1 points on the dev and test set of GLUE benchmark. Extensive results on the BERT compression, reading comprehension and ImageNet task demonstrate the effectiveness and generality of our proposed methods.

[1]Institute for Interdisciplinary Information Sciences (IIIS), Tsinghua University [2]Microsoft Research Asia [3]University of Science and Technology of China. Correspondence to: Xu Tan <xuta@microsoft.com>, Jian Li <lijian83@mail.tsinghua.edu.cn>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

1. Introduction

Neural Architecture Search (NAS) (Zoph & Le, 2017; Zoph et al., 2018; Liu et al., 2018) aims to automatize the process of discovering high-performance architectures. Conventional NAS (Zoph & Le, 2017; Zoph et al., 2018; Real et al., 2017) obtains the accuracy of an architecture by training it from scratch, which requires huge computational resources (e.g., hundreds of GPU days). To speed up the training process, an important technique, called weight sharing, is proposed by Pham et al. (2018). Briefly speaking, weight sharing is to maintain a single copy of weights on a super-net that contains all the architectures in the search space. Rather than stand-alone training from scratch for each architecture, the weight sharing method samples an architecture at each training step as a child model or a sub-graph of the super-net, and continues the training on the weights of shared operators from previously trained architectures in the super-net. Due to low computational cost and competitive performance, weight sharing has drawn wide attention (Pham et al., 2018; Bender et al., 2018; Liu et al., 2018; Xie et al., 2019; Cai et al., 2018; 2019; Li & Talwalkar, 2020; Guo et al., 2020; Peng et al., 2020; Chu et al., 2021; Ning et al., 2021).

Although a number of NAS methods based on weight sharing have been proposed and achieved impressive results, many works (Ning et al., 2021; Guo et al., 2020; Zela et al., 2019; Yu et al., 2019; Zhang et al., 2020a;b) find that the rank correlation between the estimated accuracy and ground truth accuracy of child models is low due to the interference among different child models on shared weights. The shared operators receive different gradient directions from child models with different architecture topologies for optimization even with the same batch of training data, which seriously affects the rank of the child models since interference can cause insufficient training and inaccurate performance evaluation. This phenomenon is especially severe for the complex search space that contains different types of candidate operators (e.g., convolutions and multi-head attention) and numerous child models.

While previous research works have noticed the interference issue and empirically found that the interference is correlated with factors such as the size of a search space (Shu et al., 2019; Zhang et al., 2020a), little has been discussed about the causes of the interference and how to mitigate it.

Through our quantitative analyses, we observe the interference on a shared operator is positively correlated with the number of different operators that exist between two child models. The main reason for the interference is that the topology of the architecture varies randomly along with the training process. Then the operator shared by different architecture topologies may receive activations from different inputs during the forward process and receive gradients from different outputs during the backward process. In this way, the shared operator is optimized towards different directions by different child models, which causes interference. Thus, we force the inputs and outputs of the operator to be similar to the average inputs and outputs of all the child models, and find that the interference can be reduced.

Inspired by the above observations, we propose two methods for MitigAtinG InterferenCe (MAGIC) from different perspectives: 1) reduce the changes of the architecture topologies in the adjacent sampling steps (MAGIC-T) and 2) align the inputs and outputs of the operator shared by different child models (MAGIC-A). As for MAGIC-T, we first analyze the commonly-used random single path one-shot algorithms (Guo et al., 2020) and find that the number of different operators between adjacent sampling steps is positively correlated with the number of layers of the super-net and number of candidate operators, which leads to serious interference in a large search space. To minimize the interference, we gradually change the topological environment of the shared operators by sampling a child model that differs from the child model sampled at the previous step with only one operator at each training step. As for MAGIC-A, we select the model with the best validation accuracy among the child models as an anchor model to align the inputs and outputs of all child models together respectively. The anchor model can be replaced when the performance of any child model outperforms it. Finally, MAGIC-T and MAGIC-A can be combined to further reduce the interference.

To verify the effectiveness of our methods, we adopt a challenging hybrid super-net including three types of operators: multi-head attention, feed-forward network and convolution, and conduct experiments on the large scale BERT pre-training task. The experiments verify that MAGIC-T and MAGIC-A can mitigate the interference and improve rank correlation individually. Combining them can achieve better performance. Extensive experiments on 10 natural language processing datasets of BERT pre-training, and ImagetNet classification tasks on MobileNet-V2 search space (Sandler et al., 2018) show the effectiveness and generality of our proposed methods.

The contributions of this paper are summarized as follows:

- We conduct thorough analyses on the interference issue in NAS, and find interference between two models is caused by diverse gradient directions, and is positively correlated with differences of their architecture topologies as well as their inputs and outputs of shared operators. To the best of our knowledge, we are the first to quantitatively analyze the interference with thorough empirical results.
- Inspired by our analyses, we propose two methods, MAGIC-T and MAGIC-A, to mitigate interference by reducing topological changes during the sampling process and aligning the inputs and outputs of the shared operators among different child models.
- Experiments on the BERT search space demonstrate that our methods can significantly improve the rank correlation of the super-net by mitigating interference. Our discovered architecture outperforms RoBERTa$_{\text{base}}$ by 1.1 and 0.6 points, and ELECTRA$_{\text{base}}$ by 1.6 and 1.1 points on the dev and test of GLUE tasks respectively. Extensive experiments on the BERT compression task, reading comprehension and ImageNet classification tasks verify the generality of our proposed methods.

2. Related Work

Neural architecture search. Recently, NAS methods (Pham et al., 2018; Xu et al., 2019; Ning et al., 2021) mainly adopt weight sharing to re-use the weights of previously trained architectures within a super-net (a.k.a one-shot NAS methods) to speed up the training process. Many methods leverage differentiable NAS (Liu et al., 2018; Dong & Yang, 2019; Wang et al., 2021b) by relaxing the discrete architectures into continuous space, and jointly learn the super-net weights and architectural weights. A number of other approaches utilize sampled single path one-shot NAS approaches (Bender et al., 2018; Chu et al., 2021; Li & Talwalkar, 2020; Guo et al., 2020; Cai et al., 2019). They usually adopt a chain-styled super-net, train it by randomly sampling and optimizing a single path (child model), and then use the trained super-net as a performance estimator to evaluate the accuracy of child models to search for good architectures using progressively shrinking (Xu et al., 2021; Hu et al., 2020) or evolution algorithms (Guo et al., 2020; Yu et al., 2020). They enjoy the flexibility of decoupling the training and searching stages and can support searching multiple efficient networks under different constraints (Cai et al., 2019; Xu et al., 2021). However, in the sampling process, different optimization directions, caused by training different child models, interfere with each other, which causes insufficient training and inaccurate performance evaluation (Zhang et al., 2020a). Our work focuses on the interference issue of chain-styled search space in sampled single path one-shot NAS approaches.

Weight sharing and interference. Multiple works have analyzed the effects of weight sharing. Shu et al. (2019) calculate gradient variance by adding randomly sampled

Gaussian noise on the weights of super-net, and find the optimization process is noisier and less efficient in complex search space such as stacking more cells. Pourchot et al. (2020) observe that, for weight sharing algorithms, architectures with a residual connection or a 3×3 convolution on the first node are preferred. Laube & Zell (2021) explore various training settings, including regularization, learning rate schedule and gradient clipping, and study their effects on the rank correlation of sampled single path one-shot NAS approaches (Guo et al., 2020). These methods study how different training configurations and search spaces affect the results of weight sharing algorithms. Different from these works, we try to understand and mitigate the interference on the shared weights and improve the rank ability of super-net.

Previous works have noticed the interference issue (Bender et al., 2018; Guo et al., 2020; Laube & Zell, 2021; Xie et al., 2020). Among them, Zhang et al. (2020a) conduct experiments to study the interference of weight sharing using a search space of 64 architectures. By randomly sampling architectures per step for training and plotting the accuracy of previous sampled architectures, they find that current updating with the sampled child model is detrimental to other models and thus causes high variance of the rank. In contrast, our quantitative experiment analyses reveal that the essence of the interference is different gradient directions on shared weights caused by different topologies of child models. Several works alleviate the interference with search space pruning. Zhang et al. (2020a) propose to divide the search space according to similarities of child models, such as sorting the models lexicographically and evenly slice these models into several groups, then train the super-net on individual groups of models. Moreover, to obtain more accurate performance, they directly shrink the size of search space to one child model and fine-tune it individually. However, for a large search space, identifying similar architectures is difficult and fine-tuning each child model could incur significant computational overhead. Other methods (Zhang et al., 2020b; Hu et al., 2020; Xu et al., 2021) progressively shrink the search space by removing the unpromising architectures and operators in the training process to reduce interference. Ning et al. (2021) propose several practical approaches such as operation pruning, progressively search space pruning and removing affine operations in batch normalization to reduce sharing extent. Different from these methods, inspired by our analyses, our work mitigates interference by modifying the sampling procedure and aligning inputs and outputs of shared operators. Thus, progressively shrink (pruning) methods are complementary to our methods. Chu et al. (2021) also modify the sampling process by accumulating the updates over k samples, chosen such that each of the k operators of the super-net appears exactly once in the super-net to ensure fairness of operators updating, which increase the training costs by over $k\times$ and cannot avoid the interference between different training steps. Our work aims at quantitatively analyzing the interference and proposing effective methods to mitigate it.

3. Analyzing Interference

In single path one-shot NAS approaches (Bender et al., 2018; Li & Talwalkar, 2020; Guo et al., 2020; Cai et al., 2019), the operator shared by different child models receives different gradients that result in different optimization directions, which cause the interference among different child models. Such interference causes insufficient training and inaccurate performance evaluation, which affects the rank of child models. If we can understand what factors may influence the differences of gradients from different child models, we can gain insights for designing better solutions to mitigate the interference. Thus, in this section, we conduct a series of experiments to analyze interference.

3.1. Analysis Setup

The gradients are influenced by various factors including input data, weights of the super-net and selected child model. To study the gradient changes solely caused by different child models, we first train a super-net using single path one-shot NAS algorithm (Guo et al., 2020). Then, we freeze the weights of super-net and only study the gradients caused by different child models under the same batch of training data. The super-net is organized in chain-style with N layers. Each layer contains all candidate operators in $\mathcal{O} = \{o_1, \cdots, o_C\}$, where C is the number of predefined candidate operators. A child model is a single path from the bottom layer to the top layer. Thus, there are C^N possible child models in the super-net. To study the interference in more challenging settings, we adopt a hybrid BERT search space including several popular candidate operators - multi-head attention (MHA), feed-forward network (FFN) and convolution (CONV), and conduct experiments on the BERT pre-training task (Devlin et al., 2019). Specifically, to study the interference both within the same type operators and between different types of operators, we use candidate operators $\mathcal{O}$ ={MHA6, MHA8, FFN, FFN', CONV3, CONV5}, where MHA6 is MHA with 6 heads, CONV3 is convolution with kernel size 3 and FFN' is FFN with a slightly larger inner hidden size. To exclude the influence of parameter size of different operators, we further adjust the inner hidden size of operators to ensure that they have similar parameter sizes (see Appendix A for more details). We train an $N = 12$ layer super-net using a batch of 1024 sentences on 32 NVIDIA P40 GPUs until 62,500 steps. Then we freeze the super-net, and study the gradients of different child models by feeding the same batch of data (a large batch of 2048 sentences).

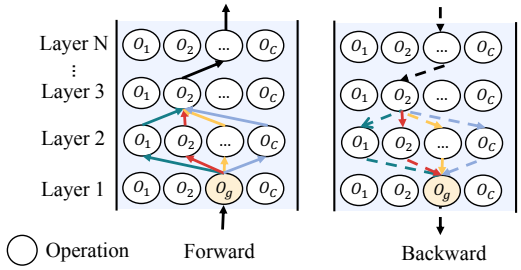


Figure 1: The illustration of the forward and backward process regarding the operator o_g in layer 1 that is shared by child models that differ in layer two.

3.2. Results and Analyses

We first focus on a simple case: child models that only differ in one operator. As shown in Figure 1, we choose C child models and they only differ in layer 2. Then we compare their gradients on o_g at the first layer[1]. To measure the gradients similarity on o_g between different child models, we flatten the gradients and calculate their cosine similarity[2]. A lower cosine similarity indicates larger difference in gradients and thus more interference on the shared operators. As shown in Figure 2 (a), we can find 1) although different child models differ by only one operator, the gradient cosine similarity is still not high, indicating that different child models can actually lead to very different gradient directions on their shared operators; 2) the same type of operators have less gradient interference (larger cosine similarity) than different types of operators, demonstrating that the interference in hybrid search space tends to be more serious than those with the same type operators.

For the hybrid search space, although all child models are trained under the same task, there is no guarantee that the operator should learn the same transformation on different child models. In other words, the inputs and outputs of the operator differ when it locates on different child models. It motivates us to explicitly force the inputs and outputs of the operators to be similar to reduce such interference. To this end, we re-train a super-net. At each training step, we first randomly choose C child models and calculate their average inputs and outputs in each layer. Then we randomly sample a child model for training and add an extra alignment training objective by calculating MSE loss between its inputs and outputs and the average ones in a layer-by-layer manner (see detailed formulation in Appendix A). Finally, we freeze the super-net and re-calculate the gradient cosine similarity as aforementioned. As shown in Figure 2 (b), we can observe that *by aligning the inputs and outputs of the shared operators to be similar to the average inputs and outputs, the gradient interference can be reduced.*

We further extend the analyses to a more general case: child models that differ in m operators. Specifically, we choose C child models that differ from the second layer to the $(m+1)$-th layers and obtain the cosine similarity matrix on o_g similar to Figure 2 (a). Then we calculate the average of the cosine similarity matrix to represent the average interference by varying m operators. The results are shown in Figure 2 (c). As we can see, *the interference on a shared operator between two child models is positively correlated with the number of different operators between them.* It demonstrates that randomly sampling child models in the training process in a complex search space could cause serious interference since their architecture topologies may differ a lot.

Although the analyses are performed under conditions of freezing the super-net and feeding the same batch of data, our findings about which factors influence the interference are general, and can help develop new methods to mitigate interference during training, as described in the next section.

4. Mitigating Interference

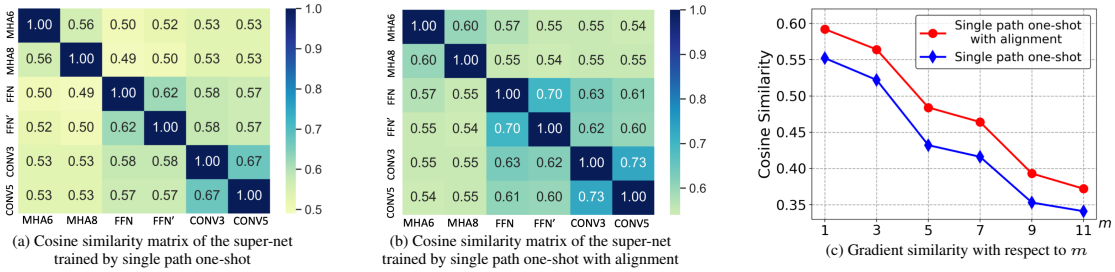
Inspired by the observations in Sec. 3, we propose two methods: MAGIC-T and MAGIC-A, to mitigate the interference.

MitigAtinG InTerferenCe from the pespective of Topological environment (MAGIC-T). According to our analyses in Sec. 3.2, more topological differences between two child models can cause more interference on the shared operators. Given a super-net with N layers and C candidate operators, previous single path NAS approaches (Bender et al., 2018; Chu et al., 2021; Li & Talwalkar, 2020; Guo et al., 2020) change $\frac{N(C-1)}{C}$ operators between α_t and α_{t-1} on average, where α_t is the sampled child model at step t, because there are N layers (operators) and the probability of each operator in α_t being different from the one in α_{t-1} is $\frac{C-1}{C}$. Modern algorithms usually employ a huge search space with dozens of layers (Pham et al., 2018; Liu et al., 2018; Cai et al., 2019; Chu et al., 2021). For such a search space, two child models sampled between adjacent sampling steps differ a lot and the gradients of shared operators interfere with each other, which is detrimental to making progress in training. To mitigate such interference, we propose MAGIC-T to gradually change the topological environment. Specifically, at each training step, MAGIC-T samples a child model α_t by randomly substituting one operator ($k = 1$) in the child model α_{t-1} sampled at the last step with another operator, and applies the forward and backward computation using α_t for weights updating.

Although MAGIC-T substitutes only one operator in one iteration, it does not only sample locally around the ini-

[1] In fact, the gradients on other shared operators are also different. We take o_g as an example for the analyses.

[2] The cosine similarity between vector $\boldsymbol{g_i}$ and $\boldsymbol{g_j}$ is calculated by $\frac{\boldsymbol{g_i} \cdot \boldsymbol{g_j}}{||\boldsymbol{g_i}|| ||\boldsymbol{g_j}||}$.



(a) Cosine similarity matrix of the super-net trained by single path one-shot

(b) Cosine similarity matrix of the super-net trained by single path one-shot with alignment

(c) Gradient similarity with respect to m

Figure 2: Gradient cosine similarity. In Figures (a) and (b), the value represents the cosine similarity between two child models. For example, MHA6-FFN-0.50 in row 1 and column 3 of the similarity matrix in Figure (a) refers to the cosine similarity of the gradients on operator o_g between two paths is 0.50 where the operator in layer 2 of one path is MHA6, and that in another path is FFN. For Figure (b), the super-net is obtained by training with an extra alignment training objective, and then is used to calculate gradient cosine similarity. In Figure (c), we choose child models that differ in m operators and calculate the average value of the cosine similarity matrix. In each experiment, we randomly choose an o_g and C paths as shown in Figure 1. The presented results are an average of 10 repeated experiments.

tial child model α_0. Indeed, as t increases, MAGIC-T can sample diverse architectures, approaching to the uniform sampling over the entire search space. This can bee shown by viewing MAGIC-T performing a random walk on the *architecture graph* $\mathcal{G}$, in which each architecture corresponds to a node and two nodes are connected by an edge if and only if they differ in only one operator. It is easy to see that $\mathcal{G}$ consists of C^N nodes (recall that N is the number of layers and C is the number of operators) and the (shortest) distance between any two nodes is at most N. Suppose the random walk $\{X_t\}_t$ starts at node $X_0 = \alpha_0$, and denote the node distribution at time t as π_t (π_t is a C^N dimensional vector and $\pi_t(v) = \Pr[X_t = v]$). Let π be the uniform distribution over the nodes of $\mathcal{G}$ (which is the stationary distribution). By standard convergence theory of random walk in Markov chain (Levin & Peres, 2017)[3] , one can show that the total variational distance between π_t and π

$$d_{TV}(\pi_t, \pi) \leq \exp(-t/N - \ln N).$$

Hence, if we want $d_{TV}(\pi_t, \pi) \leq \epsilon$, we only need $t \geq N \ln N + N \log 1/\epsilon$. Since t is sufficiently large (e.g., 250,000 steps for BERT), the variational distance between the node distribution of MAGIC-T and uniform sampling is sufficiently small. Thus, MAGIC-T can sample diverse architectures from the whole search space like uniform sampling.

MitigAtinG InTerferenCe from the perspective of inputs and outputs Alignment (MAGIC-A). In Sec. 3.2, we find that using the average inputs and outputs of shared operators for alignment can reduce the interference. However, it increases the computational cost by a factor of C, and limits the flexibility of search algorithms. Thus, we directly pick a top-performing anchor child model from the search space to align other child models. The anchor model can be replaced when the performance of another child model outperforms it. Formally, the inputs and outputs alignment loss between sampled child model $\boldsymbol{\alpha_t}$ and the anchor model $\boldsymbol{\alpha^l}$ is defined as

$$\mathcal{L}_{align}(\mathbf{H}^{(\boldsymbol{\alpha^l})}, \mathbf{H}^{(\boldsymbol{\alpha_t})}) = \sum_{n=1}^{N} \text{MSE}(\mathbf{H}_n^{(\boldsymbol{\alpha^l})}, \mathbf{H}_n^{(\boldsymbol{\alpha_t})}), \quad (1)$$

where $\mathbf{H}_n$ is the outputs of n-th layer (the inputs of $(n+1)$-th layer), and N is the number of layers. Then the training objective of a sampled child model $\boldsymbol{\alpha_t}$ is

$$\mathcal{L} = \mathcal{L}_{pred} + \lambda \mathcal{L}_{align}, \quad (2)$$

where $\mathcal{L}_{pred}$ is the supervised loss between predictions and ground truth (e.g., cross entropy loss in masked language modeling), λ is the scaling parameter that controls the weight of alignment loss. In practice, we can adopt block-wise alignment where each block contains a few layers to avoid over-regularization. The training procedure of MAGIC-A at each step is as follows:

- Obtain a batch of data and an anchor child model $\boldsymbol{\alpha}^l$, and randomly sample a child model $\boldsymbol{\alpha_t}$,
- Calculate the loss according to Eq. (5) and update the weights of $\boldsymbol{\alpha_t}$,
- Replace $\boldsymbol{\alpha}^l$ with $\boldsymbol{\alpha_t}$ if $\text{Val}(\boldsymbol{\alpha}_t) > \text{Val}(\boldsymbol{\alpha}^l)$,

where $\text{Val}(\cdot)$ is the accuracy obtained from the dev set. Maintaining a top-performing anchor model is similar to

[3]This can be proved using the standard coupling argument. See the example of random walk on the hypercube in Sec. 5.3 (Levin & Peres, 2017). Although our graph $\mathcal{G}$ is slightly more general than hypercube, the same argument applies.

prioritized paths introduced by Peng et al. (2020). However, our anchor model is used to align inputs and outputs of shared operators and reduce interference rather than distillation on the final layer to boost the training of child models.

5. Experiments and Results

5.1. Setup

Search space and super-net training. We adopt a chain-styled super-net with candidate operators $\mathcal{O}$ ={MHA12, FFN, CONV3, CONV5}. Following BERT (Devlin et al., 2019), we train the super-net and discover architectures using BookCorpus plus English Wikipedia (16GB in total). The super-net is trained with the batch size of 1024 sentences for 250,000 steps (same computational resources as $\text{BERT}_{\text{base}}$). We adopt commonly-used SPOS (Guo et al., 2020) as the baseline method, which randomly samples a child model for training at each step. For MAGIC-T, unless otherwise mentioned, it gradually changes $k = 1$ operator to reduce the interference. For MAGIC-A, we train the super-net with masked language modeling loss in the first three epochs for a warm-start and then add the alignment loss with $\lambda = 0.5$ in every four layers to avoid over-regularization. Since MAGIC-T and MAGIC-A are independent methods proposed from two different perspectives, they can also be combined (MAGIC-AT) to collaboratively mitigate the interference. See Appendix C for detailed configurations.

Architecture search and re-training. Previous single path one-shot methods usually adopt progressively shrinking (Xu et al., 2021; Hu et al., 2020) or evolution algorithms (Guo et al., 2020; Cai et al., 2019) to search effective models. Since progressively shrinking can allocate more computational resources to promising architectures and speed up the search process, we adopt it by deleting five unpromising operators (Xu et al., 2021) at the end of each epoch until only one child model is left in the search space. Note that we do not adopt progressively shrinking when analyzing the rank correlation in Sec. 5.2. To evaluate the standalone performance of child models, we re-train them from scratch for 125,000 steps with batch size 2048 on 32 NVIDIA P40 GPUs and keep other configurations the same as those of super-net. [4] We train $\text{RoBERTa}_{\text{base}}$ (Liu et al., 2019) following its original configurations while using the same computational resources as $\text{BERT}_{\text{base}}$.

Evaluation. We evaluate performance by fine-tuning pre-trained models on GLUE benchmark (Wang et al., 2019) and follow hyper-parameters of Liu et al. (2019) for fine-tuning.

[4]The total computational resources (125,000*2048 sentences of 16GB corpus) is exactly the same as $\text{BERT}_{\text{base}}$ (Devlin et al., 2019), $\text{MPNet}_{\text{base}}$ (Song et al., 2020) and $\text{ELECTRA}_{\text{base}}$ (Clark et al., 2020) for a fair comparison.

5.2. Analyses on the Rank Correlation

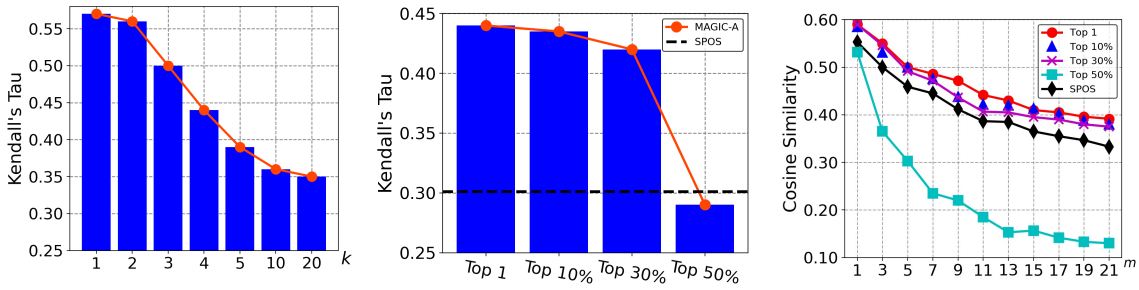
Comparison of Kendall rank correlation. We first perform correlation analysis to evaluate whether our method MAGIC can improve the rank of child models by mitigating interference. First, we uniformly sample 60 child models, train them on the pre-training tasks and obtain their ground-truth performance on the downstream MNLI task (Williams et al., 2018), which is widely used to indicate the performance of the NLP pre-training models (Liu et al., 2019; Song et al., 2020; Dong et al., 2019). Then, given a super-net, we can calculate the Kendall rank correlation coefficient (Kendall's Tau) (Kendall, 1938) between their negative weight sharing validation loss and ground-truth performance. As shown in Table 1, compared with the baseline, MAGIC-T improves the rank correlation from 0.36 to 0.49, which shows the benefits of gradual modification to mitigate the interference. Furthermore, combining MAGIC-T and MAGIC-A achieves an even higher rank correlation of 0.57. It shows that MAGIC-T and MAGIC-A, proposed from two different perspectives, can collaboratively reduce interference to further improve the rank correlation.

Table 1: Kendall's Tau on the BERT search space.

Method	SPOS (Guo et al., 2020)	MAGIC-T	MAGIC-A	MAGIC-AT
Kendall's Tau	0.36	0.49	0.44	**0.57**

Analyses of MAGIC-T. MAGIC-T gradually modifies $k = 1$ operators at each training step to reduce the interference. We further analyze MAGIC-T by varying k and measuring the rank correlation. According to analyses in Sec. 3.2, more topological changes (larger k) can lead to more serious interference on the shared operators. As shown in Figure 3 (a), it is clear to observe that larger k causes worse rank correlation, which demonstrates that the interference is detrimental to the super-net training, and our proposed MAGIC-A can efficiently mitigate the interference and improve the rank ability of the super-net.

Analyses of MAGIC-A. MAGIC-A selects a top-performing anchor model to align inputs and outputs of other child models to be similar to reduce the interference. Besides a top-performing one, we further analyze the method by choosing different anchor models. Specifically, we evaluate $10,000$ child models at the end of every epoch and select a top $p\%$ child model as the anchor model. However, different from the top-performing model, the exact top $p\%$ model varies dramatically, which results in selecting different anchor models in different epochs. Using such anchor models to align other models in each epoch can lead to unstable optimization of the super-net. For stable training, we substitute the anchor model when the previously selected anchor model in the last epoch escapes from the range of



(a) Study MAGIC-T by varying k. (b) Study of MAGIC-A. (c) Gradient cosine similarity of MAGIC-A.

Figure 3: Analyses of MAGIC-T and MAGIC-A.

Table 2: Results of pre-training models on GLUE benchmark. The test set results are obtained from the official GLUE leaderboard. All models are trained under the same computational resources as $\text{BERT}_{\text{base}}$ for a fair comparison. "E-" refers to training in the ELECTRA style. Following (Devlin et al., 2019), Spearman correlation is reported for STS-B, Matthews correlation is reported for CoLA and accuracy is reported for other tasks.

Model	Params	FLOPs	MNLI	QQP	QNLI	CoLA	SST-2	STS-B	RTE	MRPC	AVG
dev set											
$\text{BERT}_{\text{base}}$ (Devlin et al., 2019)	110M	2.9e10	84.4	89.9	88.4	54.3	92.7	88.9	71.1	86.7	82.1
$\text{RoBERTa}_{\text{base}}$ (Liu et al., 2019)	125M	3.3e10	85.3	91.1	91.1	61.0	92.7	90.0	77.5	87.9	84.6
$\text{ELECTRA}_{\text{base}}$ (Clark et al., 2020)	110M	2.9e10	-	-	-	-	-	-	-	-	85.1
$\text{MPNet}_{\text{base}}$ (Song et al., 2020)	110M	2.9e10	85.2	-	-	-	93.4	-	-	-	-
SPOS (Guo et al., 2020)	114M	3.3e10	84.7	91.4	91.4	59.6	92.1	89.7	80.9	86.3	84.4
MAGIC-AT	113M	3.3e10	85.6	91.3	91.8	61.1	**93.5**	90.3	80.9	**90.9**	85.7
E-MAGIC-AT	110M	2.9e10	**86.3**	**91.7**	**92.5**	**65.8**	92.5	**91.0**	**84.0**	89.7	**86.7**
test set											
$\text{BERT}_{\text{base}}$ (Devlin et al., 2019)	110M	2.9e10	84.6	89.2	90.5	52.1	93.5	85.8	66.4	84.8	80.9
$\text{RoBERTa}_{\text{base}}$ (Liu et al., 2019)	125M	3.3e10	84.8	89.0	91.7	57.1	93.3	88.0	74.1	84.1	82.8
$\text{ELECTRA}_{\text{base}}$ (Clark et al., 2020)	110M	2.9e10	85.8	89.1	**92.7**	59.7	93.4	87.7	73.1	86.7	83.5
SPOS (Guo et al., 2020)	114M	3.3e10	84.3	88.6	91.0	56.1	92.8	88.1	74.9	83.4	82.4
MAGIC-AT	113M	3.3e10	84.9	89.1	92.0	57.0	**94.1**	**87.8**	77.4	85.2	83.4
E-MAGIC-AT	110M	2.9e10	**85.9**	**89.6**	92.4	**60.3**	93.4	87.3	**80.4**	**87.4**	**84.6**

top $p \pm r\%$ models ($r = 10$) in the current epoch. We train different super-nets with different p and present the ranking correlation in Figure 3 (b). The cosine similarity results, as introduced in Sec. 3.2, are presented in Figure 3 (c). We can observe that a proper anchor model that ranks within the top 30% is enough to align the inputs and outputs and mitigate interference, and a better anchor model can obtain slightly better results. However, choosing a model that performs poorly as the anchor model has a negative effect on super-net training. Since the optimization goal is $\mathcal{L}_{pred} + \lambda \mathcal{L}_{align}$, alignment regularization guided by a bad child model may deviate with the training target $\mathcal{L}_{pred}$.

In the following sections, we demonstrate the effectiveness and generality of MAGIC-AT on the BERT pre-training, compression task and image classification tasks.

5.3. Searching Effective BERT Models

Results on GLUE benchmark. To show the generality of the architecture discovered by our MAGIC-AT, we train it with two different settings: 1) masked language modeling (MLM) (Devlin et al., 2019) proposed in BERT (Devlin et al., 2019) and 2) replace token detection (RTE) proposed in ELECTRA (Clark et al., 2020). The discovered architecture is shown in Figure 6 and the experimental results are shown in Table 2. Using the MLM object, our model consistently outperforms other models by a large margin on both the dev and the test set. Compared with RoBERTa, our model achieves 1.1 and 0.6 higher points on the dev and the test set respectively. Using replace token detection objective following ELECTRA, our model is superior to ELECTRA by 1.6 and 1.1 points on the dev and the test set respectively.

Results on SQuAD datasets. We further evaluate the generalizability of our searched architecture by fine-tuning it to reading comprehension tasks SQuAD v1.1 (Rajpurkar et al., 2016) and SQuAD v2.0 (Rajpurkar et al., 2018). We adopt standard evaluation metrics of Exact-Match (EM) and F1 scores following (Devlin et al., 2019; Liu et al., 2019; Clark et al., 2020). The results are shown in Table 4.

Table 3: Comparison of compression methods in the commonly-used 60M model size. "*" means using data augmentation.

Model	Params	MNLI	QQP	QNLI	CoLA	SST-2	STS-B	RTE	MRPC	AVG
dev set										
DistilBERT (Sanh et al., 2019)	66M	82.2	88.5	89.2	51.3	91.3	86.9	59.9	87.5	79.6
MiniLM (Wang et al., 2020)	66M	84.0	91.0	91.0	49.2	92.0	-	71.5	88.4	-
BERT-of-Theseus (Xu et al., 2020)	66M	82.3	89.6	89.5	51.1	91.5	88.7	68.2	-	-
PD-BERT (Turc et al., 2019)	66M	82.5	90.7	89.4	-	91.1	-	66.7	84.9	-
DynaBERT* (Hou et al., 2020)	60M	84.2	91.2	91.5	56.8	92.7	89.2	72.2	84.1	82.7
NAS-BERT (Xu et al., 2021)	60M	84.1	91.0	91.3	58.1	92.1	89.4	79.2	88.5	84.2
SPOS (Guo et al., 2020)	60M	84.0	90.7	**91.1**	57.1	91.6	88.2	75.9	86.5	83.1
MAGIC-AT	60M	84.5	90.9	91.1	61.8	92.8	89.0	78.9	89.2	**84.8**
test set										
BERT-of-Theseus (Xu et al., 2020)	66M	82.4	89.3	89.6	47.8	92.2	84.1	66.2	83.2	79.4
PD-BERT (Turc et al., 2019)	66M	82.8	88.5	88.9	-	91.8	-	65.3	81.7	-
BERT-PKD (Sun et al., 2019)	66M	81.5	88.9	89.0	-	92.0	-	65.5	79.9	-
TinyBERT* (Jiao et al., 2020)	66M	84.6	89.1	90.4	51.1	93.1	83.7	70.0	82.6	80.6
NAS-BERT (Xu et al., 2021)	60M	83.5	88.9	90.9	48.4	92.9	86.1	73.7	84.5	81.1
SPOS (Guo et al., 2020)	60M	83.5	88.5	**90.6**	52.4	91.7	86.5	74.2	83.6	81.4
MAGIC-AT	60M	84.2	88.8	90.6	53.6	92.1	86.8	75.6	84.3	**82.0**

Table 4: Results on the dev set of SQuAD datasets.

Model	SQuAD v1.1 EM	SQuAD v1.1 F1	SQuAD v2.0 EM	SQuAD v2.0 F1
$\text{BERT}_{\text{base}}$ (Devlin et al., 2019)	80.5	88.5	-	-
$\text{RoBERTa}_{\text{base}}$ (Liu et al., 2019)	82.1	89.3	74.9	78.2
$\text{ELECTRA}_{\text{base}}$ (Clark et al., 2020)	84.5	90.8	80.5	83.3
MAGIC-AT	82.7	90.0	76.6	80.4
E-MAGIC-AT	**84.8**	**91.4**	**80.8**	**83.9**

Compared with RoBERTa, our model achieves 0.6 and 0.7 higher points on EM and F1 for SQuAD v1.1, 1.7 and 2.2 higher points on EM and F1 for SQuAD v2.0 while using fewer parameters. Furthermore, our model also surpasses the ELECTRA using replace token detection objective.

5.4. Searching Compressed BERT Models

We further validate the generalizability of our algorithms for BERT model compression. Previous works on BERT compression aim to design novel architectures to explore the potential of different architectures (Xu et al., 2021) or advanced distillation methods (Sanh et al., 2019; Wang et al., 2020; Turc et al., 2019; Hou et al., 2020) to efficiently learn knowledge from the teacher model. We build a super-net as described in Sec. 5.1 but reduce the hidden size of candidate operators from 768 to 512. We re-run MAGIC-AT on this search space and evaluate the architecture found by our algorithm for model compression. Following NAS-BERT (Xu et al., 2021), we apply knowledge distillation on two stages to train the discovered model (i.e., pre-training and fine-tuning) and do not add extra techniques such as attention matrix distillation (Jiao et al., 2020; Hou et al., 2020). We adopt the teacher model used in NAS-BERT (Xu et al., 2021) to ensure that the improvement is caused by a superior architecture. Other training hyper-parameters are the same as those in Sec. 5.3. The discovered architecture and detailed configurations are presented in Appendix C. As shown in Table 3, our model achieves better performance compared to all previous approaches. This again shows the general effectiveness of our algorithms.

Table 5: Comparison of models on ImageNet.

Model	Top1/Top5 Err.	Params	FLOPS
MobileNetV2 (Sandler et al., 2018)	25.3/-	6.9M	585M
ShuffleNetV2 (Zhang et al., 2018b)	25.1/-	∼5M	591M
DARTS (Liu et al., 2018)	26.9/9.0	4.9M	595M
PC-DARTS (Xu et al., 2019)	24.2/7.3	5.3M	597M
CARS (Yang et al., 2020)	24.8/7.5	5.1M	591M
PC-NAS (Li et al., 2020)	23.9/-	5.1M	-
EnTranNAS-DST (Yang et al., 2021)	23.8/7.0	5.2M	594M
Models searched on the MobileNetV2 search space			
NAO (Luo et al., 2018)	24.5/7.8	6.5M	590M
LaNAS (Wang et al., 2021a)	25.0/7.7	5.1M	570M
BN-NAS (Chen et al., 2021)	24.3/-	4.4M	470M
ProxelessNAS (Cai et al., 2018)	24.0/7.1	5.8M	595M
RLNAS (Zhang et al., 2021)	24.4/7.4	5.3M	473M
SemiNAS (Luo et al., 2020)	23.5/6.8	6.3M	599M
MAGIC-AT	**23.2/6.7**	6.0M	598M

5.5. Searching on ImageNet

We use a MobileNet-v2 (Sandler et al., 2018) based search search space following ProxylessNAS (Cai et al., 2018), which excludes squeeze-and-excitation block (Hu et al., 2018). Candidate operations include inverted bottleneck convolution (Sandler et al., 2018) with various kernel sizes {3, 5, 7}, expansion ratios {3, 6} and zero-out layer. For super-net training, we use the SGD optimizer with an initial learning rate of 0.4 and a cosine learning rate, and train the super-net on 8 V100 GPUs for 150 epochs with a batch size

of 512. For stand-alone model training, to be consistent with the previous works, we follow the same strategy as ProxylessNAS[5] and do not employ tricks like cutout (Devries & Taylor, 2017) or mixup (Zhang et al., 2018a). We mainly compare MAGIC-AT to works with the same search space and present the results in Table 5. MAGIC-AT achieves 23.2% top-1 test error rate on ImageNet under the 600M FLOPS constraint, which outperforms baseline NAS works. The discovered architecture is depicted in Figure 11.

6. Conclusion

In this paper, we quantitatively analyze the causes of interference on shared weights in neural architecture search and develop two approaches: MAGIC-T and MAGIC-A, to mitigate it. The proposed methods can improve the rank correlation of the super-net and can search efficient architectures. Experiments on various natural language tasks and ImageNet task demonstrate the effectiveness of our methods. We hope our quantitative analyses and approaches to mitigate interference can help the NAS community to better understand and resolve the interference issue.

7. Acknowledgements

Jin Xu and Jian Li are supported in part by the National Natural Science Foundation of China Grant 62161146004, Turing AI Institute of Nanjing and Xi'an Institute for Interdisciplinary Information Core Technology.

References

Bender, G., Kindermans, P.-J., Zoph, B., Vasudevan, V., and Le, Q. Understanding and simplifying one-shot architecture search. In *International Conference on Machine Learning*, pp. 550–559. PMLR, 2018.

Cai, H., Zhu, L., and Han, S. Proxylessnas: Direct neural architecture search on target task and hardware. In *International Conference on Learning Representations*, 2018.

Cai, H., Gan, C., Wang, T., Zhang, Z., and Han, S. Once-for-all: Train one network and specialize it for efficient deployment. In *International Conference on Learning Representations*, 2019.

Cer, D., Diab, M., Agirre, E., Lopez-Gazpio, I., and Specia, L. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pp. 1–14, August 2017.

[5] https://github.com/mit-han-lab/proxylessnas

Chen, B., Li, P., Li, B., Lin, C., Li, C., Sun, M., Yan, J., and Ouyang, W. Bn-nas: Neural architecture search with batch normalization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 307–316, October 2021.

Chen, Z., Zhang, H., Zhang, X., and Zhao, L. Quora question pairs, 2018.

Chollet, F. Xception: Deep learning with depthwise separable convolutions. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pp. 1800–1807. IEEE Computer Society, 2017. doi: 10.1109/CVPR.2017.195. URL https://doi.org/10.1109/CVPR.2017.195.

Chu, X., Zhang, B., and Xu, R. Fairnas: Rethinking evaluation fairness of weight sharing neural architecture search. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 12239–12248, October 2021.

Clark, K., Luong, M., Le, Q. V., and Manning, C. D. ELECTRA: pre-training text encoders as discriminators rather than generators. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=r1xMH1BtvB.

Dagan, I., Glickman, O., and Magnini, B. The pascal recognising textual entailment challenge. In *Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Tectual Entailment*, pp. 177–190, 2006.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, 2019.

Devries, T. and Taylor, G. W. Improved regularization of convolutional neural networks with cutout. *CoRR*, abs/1708.04552, 2017. URL http://arxiv.org/abs/1708.04552.

Dolan, W. B. and Brockett, C. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*, 2005.

Dong, L., Yang, N., Wang, W., Wei, F., Liu, X., Wang, Y., Gao, J., Zhou, M., and Hon, H. Unified language model pre-training for natural language understanding and generation. In Wallach, H. M., Larochelle, H., Beygelzimer,

A., d'Alché-Buc, F., Fox, E. B., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 13042–13054, 2019.

Dong, X. and Yang, Y. Searching for a robust neural architecture in four gpu hours. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1761–1770, 2019.

Guo, Z., Zhang, X., Mu, H., Heng, W., Liu, Z., Wei, Y., and Sun, J. Single path one-shot neural architecture search with uniform sampling. In *European Conference on Computer Vision*, pp. 544–560. Springer, 2020.

Hou, L., Huang, Z., Shang, L., Jiang, X., Chen, X., and Liu, Q. Dynabert: Dynamic bert with adaptive width and depth. *Advances in Neural Information Processing Systems*, 33, 2020.

Hu, J., Shen, L., and Sun, G. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7132–7141, 2018.

Hu, Y., Liang, Y., Guo, Z., Wan, R., Zhang, X., Wei, Y., Gu, Q., and Sun, J. Angle-based search space shrinking for neural architecture search. In Vedaldi, A., Bischof, H., Brox, T., and Frahm, J. (eds.), *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XIX*, volume 12364 of *Lecture Notes in Computer Science*, pp. 119–134. Springer, 2020. doi: 10.1007/978-3-030-58529-7_8. URL https://doi.org/10.1007/978-3-030-58529-7_8.

Jiao, X., Yin, Y., Shang, L., Jiang, X., Chen, X., Li, L., Wang, F., and Liu, Q. Tinybert: Distilling bert for natural language understanding. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pp. 4163–4174, 2020.

Kaiser, L., Gomez, A. N., and Chollet, F. Depthwise separable convolutions for neural machine translation. In *International Conference on Learning Representations*, 2018.

Karatzoglou, A., Schnell, N., and Beigl, M. Applying depthwise separable and multi-channel convolutional neural networks of varied kernel size on semantic trajectories. *Neural Computing and Applications*, 32(11):6685–6698, 2020.

Kendall, M. G. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93, 1938.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In Bengio, Y. and LeCun, Y. (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980.

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., and Soricut, R. ALBERT: A lite BERT for self-supervised learning of language representations. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=H1eA7AEtvS.

Laube, K. A. and Zell, A. Exploring single-path architecture search ranking correlations, 2021. URL https://openreview.net/forum?id=J40FkbdldTX.

Levin, D. A. and Peres, Y. *Markov chains and mixing times*, volume 107. American Mathematical Soc., 2017.

Li, L. and Talwalkar, A. Random search and reproducibility for neural architecture search. In *Uncertainty in Artificial Intelligence*, pp. 367–377. PMLR, 2020.

Li, X., Lin, C., Li, C., Sun, M., Wu, W., Yan, J., and Ouyang, W. Improving one-shot nas by suppressing the posterior fading. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13836–13845, 2020.

Liu, H., Simonyan, K., and Yang, Y. Darts: Differentiable architecture search. In *International Conference on Learning Representations*, 2018.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

Luo, R., Tian, F., Qin, T., Chen, E., and Liu, T.-Y. Neural architecture optimization. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pp. 7827–7838, 2018.

Luo, R., Tan, X., Wang, R., Qin, T., Chen, E., and Liu, T.-Y. Semi-supervised neural architecture search. *Advances in Neural Information Processing Systems*, 33, 2020.

Ning, X., Tang, C., Li, W., Zhou, Z., Liang, S., Yang, H., and Wang, Y. Evaluating efficient performance estimators of neural architectures. *Advances in Neural Information Processing Systems*, 34, 2021.

Ott, M., Edunov, S., Baevski, A., Fan, A., Gross, S., Ng, N., Grangier, D., and Auli, M. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*, 2019.

Peng, H., Du, H., Yu, H., LI, Q., Liao, J., and Fu, J. Cream of the crop: Distilling prioritized paths for one-shot neural architecture search. *Advances in Neural Information Processing Systems*, 33, 2020.

Pham, H., Guan, M., Zoph, B., Le, Q., and Dean, J. Efficient neural architecture search via parameters sharing. In *International Conference on Machine Learning*, pp. 4095–4104. PMLR, 2018.

Pourchot, A., Ducarouge, A., and Sigaud, O. To share or not to share: A comprehensive appraisal of weight-sharing. *arXiv preprint arXiv:2002.04289*, 2020.

Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. SQuAD: 100,000+ questions for machine comprehension of text. In *EMNLP*, pp. 2383–2392, 2016.

Rajpurkar, P., Jia, R., and Liang, P. Know what you don't know: Unanswerable questions for squad. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 784–789, 2018.

Real, E., Moore, S., Selle, A., Saxena, S., Suematsu, Y. L., Tan, J., Le, Q. V., and Kurakin, A. Large-scale evolution of image classifiers. In *International Conference on Machine Learning*, pp. 2902–2911. PMLR, 2017.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4510–4520, 2018.

Sanh, V., Debut, L., Chaumond, J., and Wolf, T. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

Shu, Y., Wang, W., and Cai, S. Understanding architectures learnt by cell-based neural architecture search. In *International Conference on Learning Representations*, 2019.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A., and Potts, C. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*, pp. 1631–1642, October 2013.

Song, K., Tan, X., Qin, T., Lu, J., and Liu, T. Mpnet: Masked and permuted pre-training for language understanding. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Sun, S., Cheng, Y., Gan, Z., and Liu, J. Patient knowledge distillation for bert model compression. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 4314–4323, 2019.

Turc, I., Chang, M.-W., Lee, K., and Toutanova, K. Well-read students learn better: On the importance of pre-training compact models. *arXiv preprint arXiv:1908.08962*, 2019.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. R. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=rJ4km2R5t7.

Wang, L., Xie, S., Li, T., Fonseca, R., and Tian, Y. Sample-efficient neural architecture search by learning actions for monte carlo tree search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021a.

Wang, R., Cheng, M., Chen, X., Tang, X., and Hsieh, C.-J. Rethinking architecture selection in differ-entiable nas. In *International Conference on Learning Representations*, 2021b.

Wang, W., Wei, F., Dong, L., Bao, H., Yang, N., and Zhou, M. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Warstadt, A., Singh, A., and Bowman, S. R. Neural network acceptability judgments. *Trans. Assoc. Comput. Linguistics*, 7:625–641, 2019. URL https://transacl.org/ojs/index.php/tacl/article/view/1710.

Williams, A., Nangia, N., and Bowman, S. A broad-coverage challenge corpus for sentence understanding through inference. In *NAACL*, pp. 1112–1122, June 2018.

Xie, L., Chen, X., Bi, K., Wei, L., Xu, Y., Chen, Z., Wang, L., Xiao, A., Chang, J., Zhang, X., et al. Weight-sharing neural architecture search:a battle to shrink the optimization gap. *arXiv preprint arXiv:2008.01475*, 2020.

Xie, S., Zheng, H., Liu, C., and Lin, L. SNAS: stochastic neural architecture search. In *7th International Conference on Learning Representations, ICLR 2019, New*

Orleans, LA, USA, May 6-9, 2019. OpenReview.net, 2019. URL https://openreview.net/forum?id=rylqooRqK7.

Xu, C., Zhou, W., Ge, T., Wei, F., and Zhou, M. Bert-of-theseus: Compressing bert by progressive module replacing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 7859–7869, 2020.

Xu, J., Tan, X., Luo, R., Song, K., Li, J., Qin, T., and Liu, T. NAS-BERT: task-agnostic and adaptive-size BERT compression with neural architecture search. In Zhu, F., Ooi, B. C., and Miao, C. (eds.), *KDD '21: The 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, Singapore, August 14-18, 2021*, pp. 1933–1943. ACM, 2021. doi: 10.1145/3447548.3467262. URL https://doi.org/10.1145/3447548.3467262.

Xu, Y., Xie, L., Zhang, X., Chen, X., Qi, G.-J., Tian, Q., and Xiong, H. Pc-darts: Partial channel connections for memory-efficient architecture search. In *International Conference on Learning Representations*, 2019.

Yang, Y., You, S., Li, H., Wang, F., Qian, C., and Lin, Z. Towards improving the consistency, efficiency, and flexibility of differentiable neural architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6667–6676, June 2021.

Yang, Z., Wang, Y., Chen, X., Shi, B., Xu, C., Xu, C., Tian, Q., and Xu, C. Cars: Continuous evolution for efficient neural architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1829–1838, 2020.

Yu, J., Jin, P., Liu, H., Bender, G., Kindermans, P.-J., Tan, M., Huang, T., Song, X., Pang, R., and Le, Q. Bignas: Scaling up neural architecture search with big single-stage models. In *European Conference on Computer Vision*, pp. 702–717. Springer, 2020.

Yu, K., Sciuto, C., Jaggi, M., Musat, C., and Salzmann, M. Evaluating the search phase of neural architecture search. In *International Conference on Learning Representations*, 2019.

Zela, A., Siems, J., and Hutter, F. Nas-bench-1shot1: Benchmarking and dissecting one-shot neural architecture search. In *International Conference on Learning Representations*, 2019.

Zhang, H., Cisse, M., Dauphin, Y. N., and Lopez-Paz, D. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018a.

Zhang, X., Zhou, X., Lin, M., and Sun, J. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 6848–6856, 2018b.

Zhang, X., Hou, P., Zhang, X., and Sun, J. Neural architecture search with random labels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10907–10916, June 2021.

Zhang, Y., Lin, Z., Jiang, J., Zhang, Q., Wang, Y., Xue, H., Zhang, C., and Yang, Y. Deeper insights into weight sharing in neural architecture search. *arXiv preprint arXiv:2001.01431*, 2020a.

Zhang, Y., Zhang, Q., and Yang, Y. How does supernet help in neural architecture search? *arXiv preprint arXiv:2010.08219*, 2020b.

Zoph, B. and Le, Q. V. Neural architecture search with reinforcement learning. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=r1Ue8Hcxg.

Zoph, B., Vasudevan, V., Shlens, J., and Le, Q. V. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8697–8710, 2018.

A. Configurations and More Analyses of Interference

In this section, we present detailed configurations of results presented in Sec. 3 and more analyses of interference. We first describe the structure of operators in Sec. A.1. Then we present how to align the inputs and outputs of shared operators with the average inputs and outputs in Sec. A.2. Next, to give readers a better understanding, we depict how we extend the analyses to a more general case: child models that differ in m operators in Sect. A.3. Finally, we discuss more results of interference when o_g locates in different layers in Sec. A.4.

A.1. Candidate Operators

We adopt a hybrid BERT search space including popular and different types of candidate operators: multi-head attention (MHA), feed-forward network (FFN) and convolution (CONV). The structures of FFN and MHA follow those in BERT (Devlin et al., 2019) as shown in Figure 4 (b) and (c). We adopt separable convolution (Chollet, 2017), as shown in Figure 4 (a), since 1) its effectiveness in natural language processing tasks have been demonstrated by previous work (Kaiser et al., 2018; Karatzoglou et al., 2020; Xu et al., 2021) and 2) its number of parameters does not change much by varying the kernel size, which allows us to conveniently obtain different CONV operators with the similar size of parameters. To exclude the influence of parameter size of different type operators, we further adjust the inner hidden size of operators to ensure that they have a similar number of parameters as shown in Table 6.

Table 6: Configurations of different operators used for analyzing interference.

	Candidate Operators		
O	{MHA6, MHA8, FFN, FFN', CONV3, CONV5}		
Operator	Hidden Size	Parameters	Configurations
MHA 6	768	1.18M	Heads 6, QKV hidden 384
MHA 8	768	1.18M	Heads 8, QKV hidden 384
FFN	768	1.18M	Inner hidden 768
FFN'	768	1.28M	Inner hidden 832
CONV3	768	1.18M	Kernel 3
CONV5	768	1.19M	Kernel 5

A.2. Aligning the Inputs and Outputs with the Average Inputs and Outputs

To explicitly align the inputs and outputs of the shared operators to be similar to the average inputs and outputs, we should first enumerate all possible child models in the search space and obtains their inputs and outputs in every layer. However, it can incur significant computational overhead. Thus, we only randomly choose C child models and obtain their average inputs and outputs as follows:

$$\mathbf{H}_n^{\text{avg}} = \frac{1}{C}\sum_{i=1}^{C}\mathbf{H}_n^{(\boldsymbol{\alpha^i})}, n = 1, 2, \cdots, N \tag{3}$$

where $\mathbf{H}_n^{(\boldsymbol{\alpha_i})}$ is the outputs of n-th layer (the inputs of $(n+1)$-th layer) of child model $\boldsymbol{\alpha^i}$, $\mathbf{H}_n^{\text{avg}}$ is the average outputs of C child models in n-th layer and N is the number of layers. Then, the alignments loss is defined as

$$\mathcal{L}_{align}(\mathbf{H}_n^{\text{avg}}, \mathbf{H}^{(\boldsymbol{\alpha})}) = \sum_{n=1}^{N}\text{MSE}(\mathbf{H}_n^{\text{avg}}, \mathbf{H}_n^{(\boldsymbol{\alpha})}), \tag{4}$$

where α is a sampled child model for optimization and MSE is mean square error loss. The training objective of a child model α is

$$\mathcal{L} = \mathcal{L}_{pred} + \lambda\mathcal{L}_{align}, \tag{5}$$

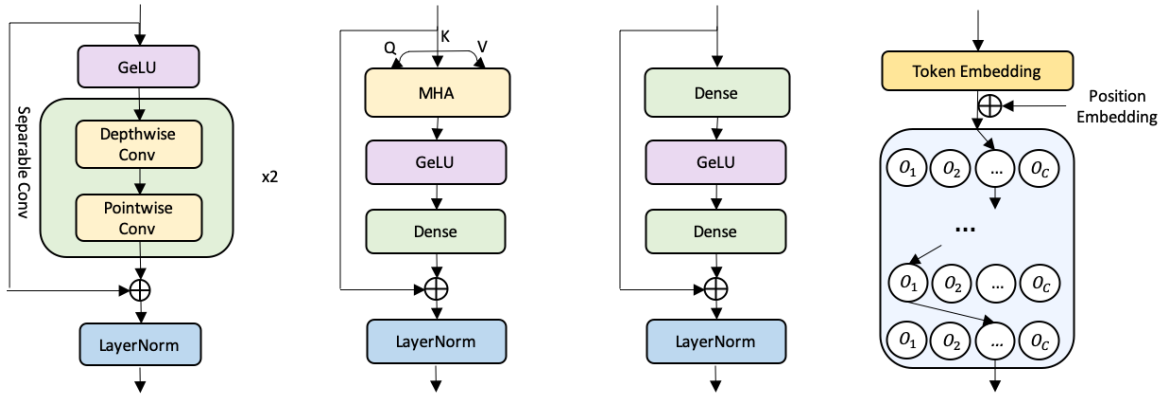
where $\mathcal{L}_{pred}$ is the supervised loss between the predictions and the ground truth (e.g., cross entropy loss in masked language modeling), λ is the scaling parameter that controls the weight of the alignment loss and we adopt $\lambda = 0.5$ in our experiments.

A.3. The Setting about Interference of the Operator Shared by Child Models that Differ in m Operators

To extend the analyses to a more general case, we study the child models that differ in m operators in Sec. 3.2. In this section, we depict how child models differ in m operators to give readers a better understanding. Figure 5 (c) shows the case that we randomly choose C child models that differ from the second layer to the fourth layer ($m = 2$ different operators). Then the C child models can apply the forward and backward computation individually and obtain their gradients on o_g. In this way, we can study the gradient interference on o_g when child models differ in m operators.

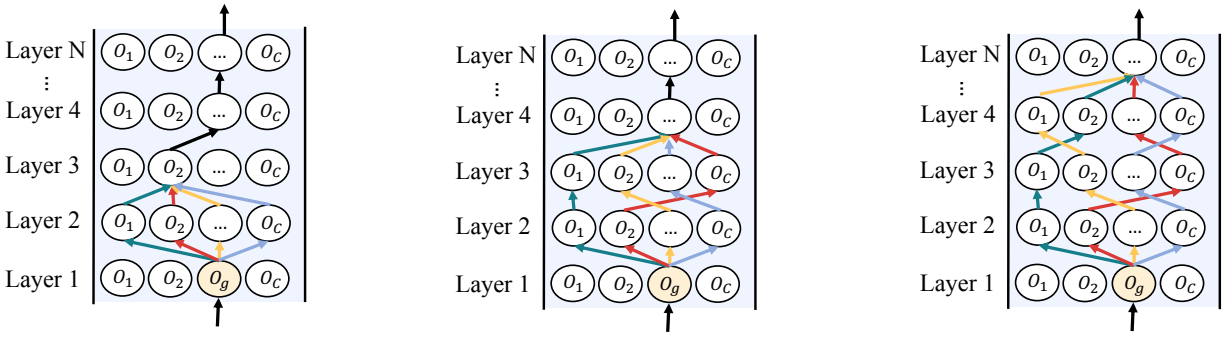
A.4. Study of Interference when o_g Locates in Different Layers

In the above analyses, we select o_g in the first layer as an example to study interference. In this section, we further investigate interference when o_g locates in different layers. As shown in Figure 6, we choose C child models which differ in layer $j+1$. Then we compare their gradients on o_g at the j-th layer. The gradient cosine similarities with different j are presented in Figure 8. We can find that our previous observations still hold no matter which layer o_g locates in. Specifically, 1) by aligning the inputs and outputs of the shared operators to be similar to the average inputs and outputs, the cosine similarity is improved and the gradient interference can be reduced; 2) the interference on a shared operator between two child models is positively correlated with the number of different operators between them (the larger m, the lower cosine similarity).



(a) Conv (b) MHA (c) FFN (d) Super-net

Figure 4: The architecture of operators and the super-net for analyzing interference.



(a) $m = 1$ (b) $m = 2$ (c) $m = 3$

Figure 5: The illustration of the forward process regarding the operator o_g in layer 1 that is shared by child models that differ from the second layer to the $m + 1$ layers.

B. MAGIC-AT

It is straightforward to combine MAGIC-A and MAGIC-T to collaboratively mitigate interference since these two independent methods proposed from two different perspectives are independent. Here we present the complete procedure for MAGIC-AT as follows:

- Obtain a batch of data, an anchor child model $\boldsymbol{\alpha}^l$ and a sampled child model $\boldsymbol{\alpha_{t-1}}$ at step $t-1$.
- Sample a child model α_t by randomly substituting one operator ($k = 1$) in $\boldsymbol{\alpha_{t-1}}$ with another operator,
- Calculate the loss according to Eq. 2 and update the weights of $\boldsymbol{\alpha_t}$,
- Replace $\boldsymbol{\alpha}^l$ with $\boldsymbol{\alpha_t}$ if $\text{Val}(\boldsymbol{\alpha}_t) > \text{Val}(\boldsymbol{\alpha}^l)$,

where $\text{Val}(\cdot)$ is the accuracy obtained from the validation set.

C. Experiment Configurations

Candidate operators for BERT tasks. To search an effective architecture for practical usage, we adopt candidate operators $\mathcal{O}$ ={MHA12, FFN, CONV3, CONV5}, where FFN and MHA12 (head dimension 64) follow the designs in $\text{BERT}_{\text{base}}$ (Devlin et al., 2019), CONV with kernel size 3 and 5 are used since their effectiveness and efficiency in natural language tasks have been demonstrated by previous work (Xu et al., 2021). We adopt the architecture of CONV as shown in Figure 7 since we observe its superior performance and efficiency than that of the separable convolution. The hidden size of operators is 768 following $\text{BERT}_{\text{base}}$ (Devlin et al., 2019). $\text{BERT}_{\text{base}}$ has 24 sublayers (each Transformer layer has a MHA12 and FFN). Since CONV operators have fewer parameters than FFN, the searched architectures usually have fewer parameters than $\text{BERT}_{\text{base}}$ with the same layers in the super-net. We adopt a chain-styled super-net with two more layers to enable that searched architecture can have similar parameters and FLOPs as $\text{BERT}_{\text{base}}$. The space contains about 4.5×10^{15}

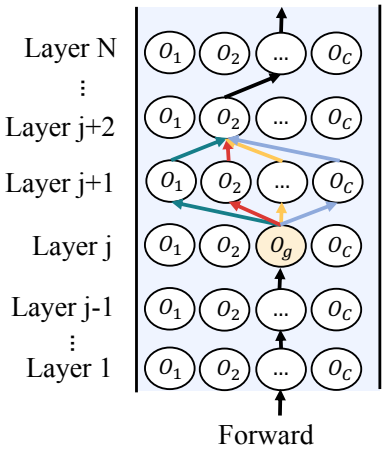


Figure 6: The illustration of the forward and backward process regarding the operator o_g in layer j that is shared by child models that differ in layer $j+1$.

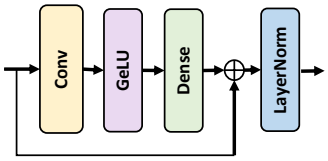


Figure 7: Architecture of convolution operator.

child models in total.

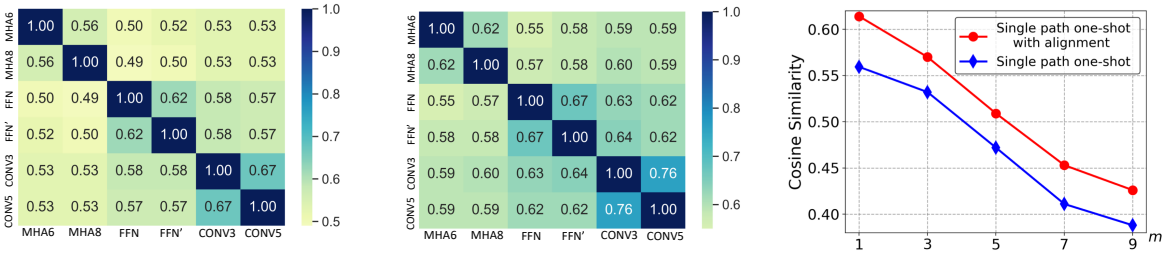
Super-net Training We use Adam (Kingma & Ba, 2015) with a learning rate of 1e-4, $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The peak learning rate is 5e-4 with a warmup step of 10,000 followed by linear annealing. The dropout rate is 0.1 and the weight decay is 0.01. We set the max length of sentences as 128 tokens. The super-net is trained with the batch size of 1024 sentences for 250,000 steps (same computational resources as $\text{BERT}_{\text{base}}$). We adopt commonly-used SPOS (Guo et al., 2020) as the baseline method, which randomly samples a child model for training at each step. For our proposed method MAGIC-T, unless otherwise mentioned, it gradually changes $k = 1$ operator to reduce the interference. For MAGIC-A, we train the super-net with masked language modeling loss in the first three epochs for a warm-start and then add the alignment loss with $\lambda = 0.5$ in every four layers to avoid over-regularization. Our experiments are implemented with fairseq codebase (Ott et al., 2019).

GLUE Benchmark We evaluate performance by fine-tuning the pre-trained model on GLUE benchmark (Wang et al., 2019), which includes three inference tasks (MNLI (Williams et al., 2018), QNLI (Rajpurkar et al., 2016) and RTE (Dagan et al., 2006)), three similarity and paraphrase tasks (MRPC (Dolan & Brockett, 2005) STS-B (Cer et al., 2017), QQP (Chen et al., 2018)) and two single-sentence tasks (CoLA (Warstadt et al., 2019), and SST-2 (Socher et al., 2013)). We follow hyper-parameters of RoBERTa for fine-tuning, where STS-B, MRPC and RTE are started from the model fine-tuned on MNLI (Liu et al., 2019; Clark et al., 2020; Song et al., 2020).

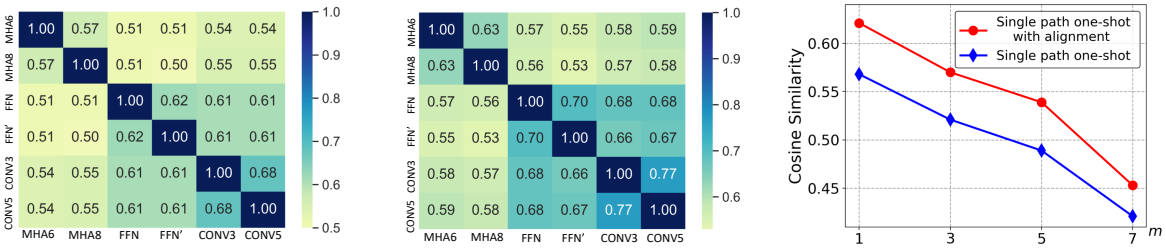
Configurations of the BERT pre-training task. To show the generality of the architecture found by our proposed MAGIC, we train the architecture with two different training objectives: 1) masked language modeling (MLM) (Devlin et al., 2019) proposed in BERT (Devlin et al., 2019) and 2) replace token detection (RTE) proposed in ELECTRA (Clark et al., 2020). For MLM training objective, RoBERTa (Liu et al., 2019) is a competitive baseline. For a fair comparison with it, we train our pre-training model with a large byte-level BPE vocabulary containing 50K subword units following RoBERTa (Liu et al., 2019). However, a large vocabulary increases the size of parameters in the embedding layer. Thus, we factorize the embedding matrix into a multiplication of a small embedding matrix with hidden size 512 and another 512×768 transformation matrix following (Lan et al., 2020). For RTE training objective, we used the character-level BPE vocabulary of size 30K following ELECTRA (Clark et al., 2020). The searched architecture is shown in Figure 9.

Configurations of the BERT compression task. Previous works usually compress BERT into a model size of 66M or 60M by taking advantage of novel architectures (Xu et al., 2021) or sophisticated distillation techniques (Sanh et al., 2019; Wang et al., 2020; Turc et al., 2019; Hou et al., 2020). Following NAS-BERT (Xu et al., 2021), we search a novel architecture of about 60M for comparison. To this end, we reduce the hidden size of candidate operators from 768 to 512 and re-run MAGIC-AT to search for architectures. The searched architecture is shown in Figure 10. We adopt the large byte-level BPE vocabulary containing 50K subword units (Liu et al., 2019) and re-train the searched architecture exactly following NAS-BERT for a fair comparison.

Configurations of the ImageNet task. We adopt candidate operators $\mathcal{O}$ ={MB3 3×3, MB3 6×6, MB5 3×3, MB5 6×6, MB7 3×3, MB7 6×6, Zero-out} following Proxyless-NAS (Cai et al., 2018), where MB7 6×6 refers to mobile inverted bottleneck convolution with kernel size 6 and expansion rate 7. We search the operation of each individual layer via MAGIC-AT. The discovered architecture by MAGIC-AT is shown in Figure 11.

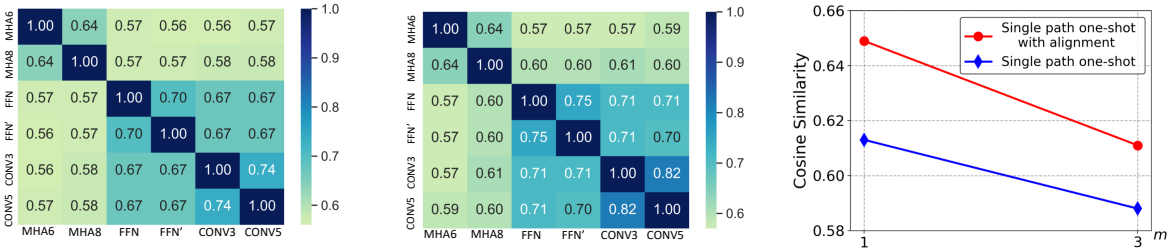


(a) Gradient cosine similarity comparison when o_g locates in layer $j = 3$

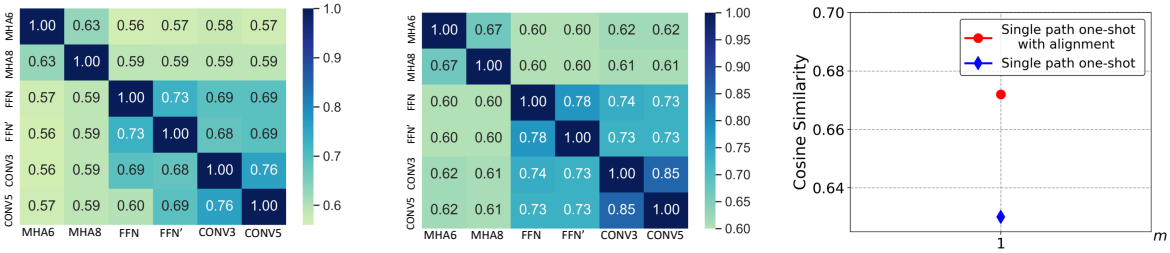


(b) Gradient cosine similarity comparison when o_g locates in layer $j = 5$

(c) Gradient cosine similarity comparison when o_g locates in layer $j = 7$



(d) Gradient cosine similarity comparison when o_g locates in layer $j = 9$



(e) Gradient cosine similarity comparison when o_g locates in layer $j = 11$

Figure 8: Gradient cosine similarity. By varying j, we repeat the experiments and present results as described in Sec. 3.2.

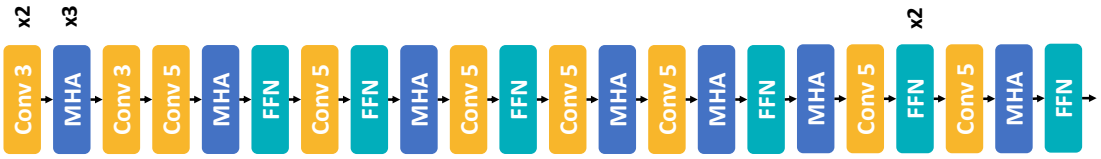


Figure 9: Architecture found by MAGIC-AT for the BERT pre-training task.

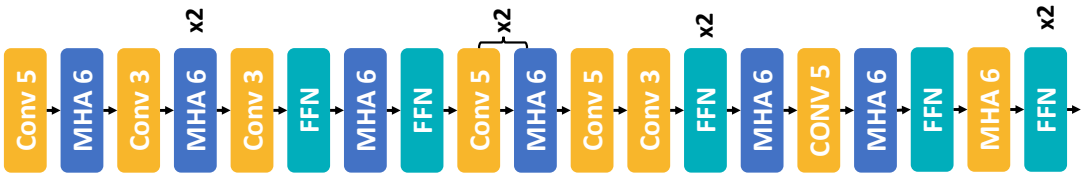


Figure 10: Architecture found by MAGIC-AT for BERT model compression.

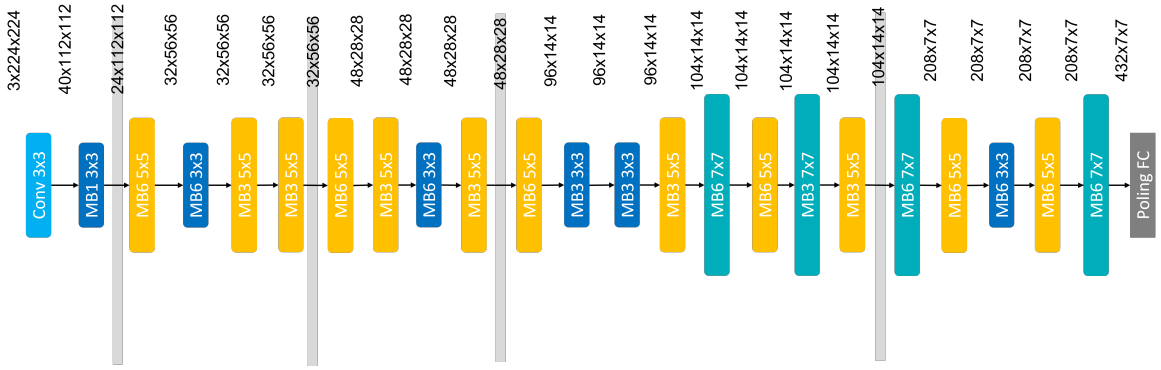


Figure 11: Architecture found by MAGIC-AT for ImageNet.