AutoBERT-Zero: Evolving BERT Backbone from Scratch

Jiahui Gao[1], Hang Xu[2*], Han Shi[3], Xiaozhe Ren[2],
Philip L.H. Yu[4], Xiaodan Liang[5], Xin Jiang[2], Zhenguo Li[2]

[1] The University of Hong Kong, Hong Kong
[2] Huawei Noah's Ark Lab
[3] Hong Kong University of Science and Technology, Hong Kong
[4] The Education University of Hong Kong, Hong Kong
[5] Sun Yat-sen University, China
sumiler@hku.hk, {xu.hang, renxiaozhe, Jiang.Xin, li.zhenguo}@huawei.com
hshiac@cse.ust.hk, plhyu@eduhk.hk, xdliang328@gmail.com

Abstract

Transformer-based pre-trained language models like BERT and its variants have recently achieved promising performance in various natural language processing (NLP) tasks. However, the conventional paradigm constructs the backbone by purely stacking the manually designed global self-attention layers, introducing inductive bias and thus leads to sub-optimal. In this work, we make the first attempt to automatically discover novel pre-trained language model (PLM) backbone on a flexible search space containing the most fundamental operations from scratch. Specifically, we propose a well-designed search space which (*i*) contains primitive math operations in the intra-layer level to explore novel attention structures, and (*ii*) leverages convolution blocks to be the supplementary for attentions in the inter-layer level to better learn local dependency. To enhance the efficiency for finding promising architectures, we propose an Operation-Priority Neural Architecture Search (OP-NAS) algorithm, which optimizes both the search algorithm and evaluation of candidate models. Specifically, we propose Operation-Priority (OP) evolution strategy to facilitate model search via balancing exploration and exploitation. Furthermore, we design a Bi-branch Weight-Sharing (BIWS) training strategy for fast model evaluation. Extensive experiments show that the searched architecture (named **AutoBERT-Zero**) significantly outperforms BERT and its variants of different model capacities in various downstream tasks, proving the architecture's transfer and scaling abilities. Remarkably, AutoBERT-Zero-base outperforms RoBERTa-base (using much more data) and BERT-large (with much larger model size) by 2.4 and 1.4 higher score on GLUE test set.

1 Introduction

Benefiting from the powerful capacity of self-attention structures in transformers (Vaswani et al. 2017), the pre-trained language models (PLM) (*e.g*. BERT (Devlin et al. 2019), RoBERTa (Liu et al. 2019b), ALBERT (Lan et al. 2020), GPT3 (Brown et al. 2020)) have achieved satisfying performance across various NLP tasks (Wang et al. 2018; Rajpurkar et al. 2016; Rajpurkar, Jia, and Liang 2018; Zellers et al. 2018). All these models are based on the fixed
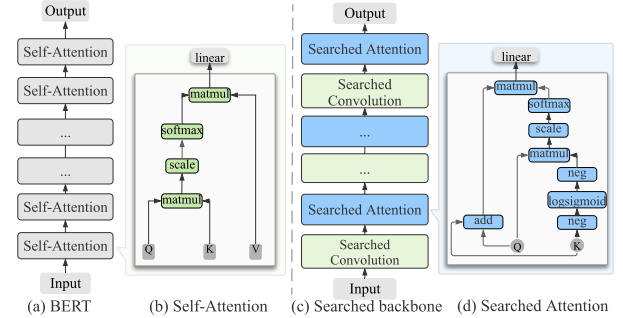
---

*Corresponding author.

Figure 1: Comparison between BERT and our searched model. Our searched AutoBERT-Zero is a hybrid structure with convolution layers and the novel searched attention layers, whose kernel sizes and attention structures are various across different layers.

hand-crafted self-attention structure by varying training resources, parameter numbers, layer numbers and inputs.

The conventional paradigm constructs the backbone by stacking the manually-designed global self-attention layers. However, many recent works have pointed out that the design of self-attention structures is not optimal (Kovaleva et al. 2019; Michel et al. 2019; Dong et al. 2021), whose inductive bias limits its performance as well as efficiency. In particular, (Dong et al. 2021) find that repeatedly stacking self-attention results to "token-uniformity" problem, meaning that different tokens are mapped to similar latent representations. Even though they claim that skip connection and multi-layer perceptions mitigate this problem, we still observe it on the BERT output (see Figure 5). Another work Reformer (Kitaev et al. 2020) discovered that sharing the weights for query and key does not impact the model's performance, indicating that redundant parameters exist in self-attention structure. In addition, ConvBERT (Jiang et al. 2020) shows that local operations such as convolution helps better learn the inherent local dependencies in natural languages. Here, we raise the following questions: Does there exist more powerful and efficient attention beyond the pure query-key-value self-attention for PLM? Can we boost the model performance and efficiency by flexibly combining global attention with local operations?

To address the above fundamental challenges in the NLP field, we resort to Neural Architecture Search (NAS), which

has emerged as a powerful technique to automatically discover promising models without excessive human intervention and tedious tuning. NAS is empowered by a search algorithm and a well-designed search space. The effectiveness of NAS is validated on many computer vision tasks (e.g., image classification (Zoph and Le 2016; Shi et al. 2020), object detection (Xu et al. 2019; Yao et al. 2021) . Nevertheless, few works leverage NAS to design backbone structure for PLM. The only related works, AdaBERT (Chen et al. 2020) and DynaBERT (Hou et al. 2020) use NAS to compress the full-sized BERT into small models, while Evolved Transformer (So, Le, and Liang 2019) searches architecture on specific downstream tasks. Besides, as architectures in AdaBERT and Evolved Transformer are task-specific, those models are not applicable for general NLP tasks. Meanwhile, the searched models in DynaBERT and Evolved Transformer are still transformer-based, which does not explore more powerful attention structure.

To the best of our knowledge, using NAS to discover a novel general PLM backbone from scratch has not been investigated. In this work, we aim to explore powerful PLM backbone by discovering novel attention structures as well as whole backbone architecture from a flexible search space. Specifically, we design both intra-layer and inter-layer search spaces that provide a wide variety of candidate architectures to prevent the inductive bias in conventional transformer. The intra-layer search space with few constraints enables finding novel self-attention mechanism, which contains various primitive mathematical operations to construct computation graph with variable path length and flexible input nodes. The inter-layer search space contains global (self-attention) and local operations (convolution) on the backbone level, which provides flexibility in learning global and local dependencies at different layers.

Since pretraining a PLM is quite time consuming, the computational burden of NAS for PLM is much more overwhelming than utilizing NAS for CV tasks, especially given that our search space is extremely huge. Thus, it is crucial to make the NAS algorithm more efficient in terms of both speed and memory. To this end, we propose a novel Operation-Priority Neural Architecture Search (OP-NAS) algorithm. During search phase, we promote Operation-Priority (OP) evolution strategy. This strategy leverages prior information of operations at each position in the computation path to flexibly balance exploration and exploitation when mutating new architectures, which escapes local optimal and speeds up the search. To facilitate model evaluation, we design Bi-branch Weight-Sharing (BIWS) training strategy, which introduces a super-net to keep track of the trained weights for both the attention structures and convolution blocks on each layer. The candidates are initialized with the weights extracted from the super-net during evaluation to prevent repeated pretraining.

Extensive experiments are conducted on the widely used Natural Language Understanding(NLU) and Question Answering(QA) benchmarks. The best searched architecture(named **AutoBERT-Zero**) is shown on Figure 1(c), which stacks novel searched attention structures and convolutions. Our AutoBERT-Zero achieves 87.7 GLUE score

when trained on the commonly used vallina pre-train tasks, consistently outperforming current state-of-the-art (SOTA) methods by a large margin (4.1 higher than T5), while requiring fewer parameters (52.7% fewer parameters than T5). More remarkably, our AutoBERT-Zero-base surpasses RoBERTa-base (using much more data) and BERT-large (with much larger model size) by 2.4 and 1.4 higher score on GLUE test set.

Our main contributions are summarized as follows: (*i*) This is the first work conducting NAS to automatically discover new self-attention structures and better backbones for PLM. (*ii*) The well-designed search space allows flexible variations in self-attention structures/input nodes/combinations of local and global operations, which enables deriving powerful architectures. (*iii*) The proposed OP evolution algorithm and BIWS training significantly accelerate the model search and evaluation. (*iv*) Extensive downstream evaluations demonstrate the effectiveness and scaling ability of the searched model AutoBERT-Zero.

2   Related Works

Pre-trained Language Model (PLM). Recently, the transformer-like paradigm (Vaswani et al. 2017; Radford et al. 2018) has dominated the research on pre-trained language models. BERT (Devlin et al. 2019) achieves SOTA performance in various NLU tasks by stacking the encoder of the transformer. Later, diverse BERT variants appear. For example, UniLM (Dong et al. 2019), XLNet (Yang et al. 2019), ELECTRA (Clark et al. 2019) introduce new pretraining objectives; Synthesizer (Tay et al. 2021) considers using random matrices to replace the dot-product self-attention mechanism; ConvBERT (Jiang et al. 2020) replaces part of attention heads with span-based convolution. However, to the best of our knowledge, apart from ConvBERT and Synthesizer, no other work challenges the transformer-based backbone that purely uses the dot-product self-attention module. In this work, we delve into a more general formulation of attention expression by the combination of primitive math operations.

Neural Architecture Search (NAS). Early NAS methods search SOTA architectures based on reinforcement learning (Zoph and Le 2016), which is computationally expensive. Subsequently, AmoebaNet (Real et al. 2019) applies the evolution algorithm for NAS. More EA-based methods were further proposed, which exploit the evaluated candidates by modifying how the population list is maintained (Zhu et al. 2019; Liu et al. 2019a). Gradient-based methods such as DARTS (Liu, Simonyan, and Yang 2018) were designed to speed up the model search at the expense of higher memory consumption. More recently, AutoML-Zero (Real et al. 2020) proves that using the basic mathematical operators can successfully develop a machine learning algorithm.

NAS for Pre-trained LM. Despite the satisfying performance in CV fields, for pre-trained language model, NAS methods are only adopted to BERT compression. AdaBERT (Chen et al. 2020) first introduces NAS to compress BERT into small models using traditional convolution operations. However, the searched architectures are task-specific rather than general pre-trained language models. DynaBERT (Hou
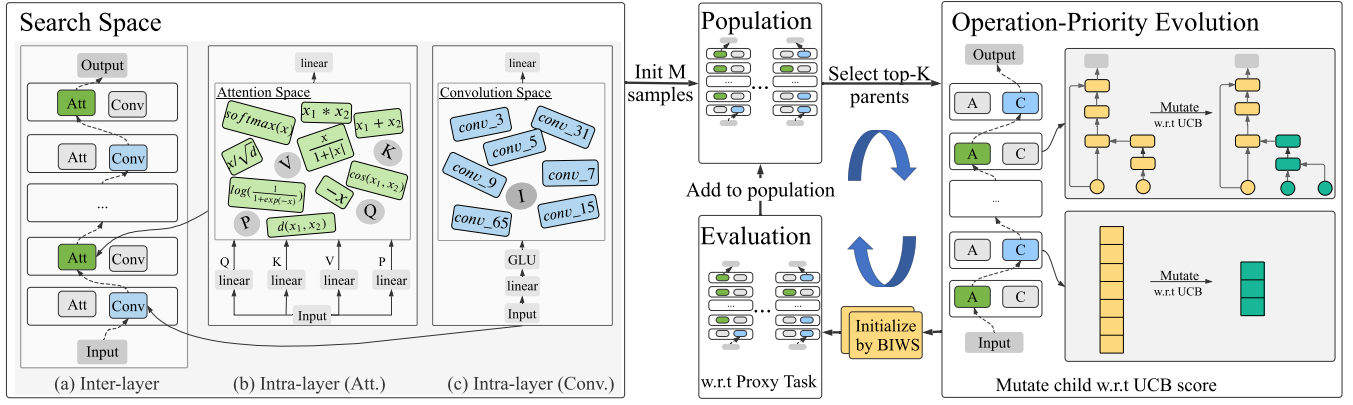
Figure 2: An overview of our OP-NAS framework for pre-trained language models. Our method directly searches better backbone architectures from scratch (using primitive operations). We propose a hierarchical search space for exploring new self-attention structures and an efficient combination of local and global dependencies. By introducing operation-priority(OP) evolution algorithm with BIWS strategy, our method efficiently searches over a wide range of the possible arichitecures.

et al. 2020) proposes a training method allowing compression in both width and depth directions w.r.t the full-sized teacher BERT model, whose searched models are still transformer backbones. Orthogonal to the above methods, inspired by the view of AutoML-Zero, we design a search space containing primitive operators and propose a novel NAS method to develop novel attention structure and backbone for general PLM from scratch.

3 Methods

In this section, we present an efficient PLM architecture searching pipeline that evolves the backbone from scratch, as shown in Figure 2. We first introduce our hierarchical coarse-to-fine search space, then elaborate on our operation-priority Neural Architecture Search (OP-NAS) algorithm.

3.1 Search Space Design

We design a two-level search space for discovering novel self-attention structures as well as an overall efficient PLM backbone: (*i*) **intra-layer level search space** enables exploring new self-attention structures from primitive operation level; (*ii*) **inter-layer level search space** leverages global attention layers and local convolution towards an efficient combination of local and global dependencies.

Intra-layer Search Space As shown in Figure 1(b), the original self-attention head can be expressed as follows:

$$Attn(X) = \sigma(XW_Q(XW_K)^\top/\sqrt{d_h})XW_V W_O^\top \quad (1)$$
$$= \sigma(QK^\top/\sqrt{d_h})VW_O^\top, \quad (2)$$

where $X \in \mathbb{R}^{n \times d}$ is the input, σ is softmax function and self-attention layer is parameered by $W_Q^k, W_K^k, W_V^k, W_O^k \in \mathbb{R}^{d \times d_h}(d_h = d/H)$. The input nodes for a typical self-attention layer are calculated by three fully connected layers from the inputs, called *query* ($Q = XW_Q$), *key* ($K = XW_K$) and *value* ($V = XW_V$). We raise two questions: (a) Can we use fewer inputs (e.g., two inputs) to make the transformer more efficient? (b) Can we build a more powerful self-attention architecture by incorporating various mathematical operations?

Type	Operation	Expression
unary	neg	$-x$
	transpose	$x^\top$
	scale	$x/\sqrt{d_x}$
	softmax	$softmax(x)$
	logsigmoid	$log(1/(1+exp(-x)))$
	softsign	$x/(1+|x|)$
binary	add	$x_1 + x_2$
	matmul	$x_1 \cdot x_2$
	cosine similarity	$cos(x_1, x_2)$
	euclidean distance	$d(x_1, x_2)$

Table 1: Mathematical primitive operations in our Intra-layer Search Space. We try to find a better self-attention structure by construct those operations in a DAG computation graph.

(1) Flexible Input Nodes. For question (a), we allow flexible number of input nodes for our self-attention architecture. More specifically, we add another input node P to construct a search space with four input nodes, where P is mapped through another linear transformation matrix from the original input ($P = XW_P$). Different from the original transformers with fixed three input nodes, our intra-layer search space allows a range of $2 \sim 4$ input nodes.

(2) Primitive Operations. The key component of transformer architecture is the self-attention layer, which first generates an attention matrix, then use it to calculate the weighted sum of *values*. The attention matrix measures the similarity between the *queries* and *keys*. For question (b), we enable finding a better structure of self-attention by designing a more flexible primitive operation search space. Rather than only using ⟨*matmul*⟩ and ⟨*softmax*⟩ as in the original transformer, our primitive operation search space includes various kinds of unary element-wise functions and binary aggregation functions as shown in Table 1. The operations such as *neg, add* and *multiplication* can be performed on both scalar and matrix inputs.

(3) Computation Graph with Variable Path Length. As Figure 2 illustrates, we represent the new attention structure as a directed acyclic graph (DAG), which transforms input

nodes into the tensor output (i.e., the output of self-attention layers) with multiple primitive operators in the intermediate graph. To better promote exploration of novel attention structures, we do not fix the path length of attention computation graphs. Note that it is possible that the dimension of the input features in the computation graph are not matched during the calculation. We examine whether every operation is legit and early reject those illegal computation graphs. We also verify that the input and output dimensions of searched attention architectures are matched to ensure layers can be stacked correctly.

Inter-layer Search Space For the design of the whole backbone, we 1) incorporate local dependency via lightweight convolution and 2) adopt a macro search space to promote the flexibility of design.

(1) Incorporating Local Dependencies. As pointed out by (Jiang et al. 2020; Wu et al. 2018), some of the attention heads can be replaced by local operations to better learn local dependencies as well as reduce model complexity. Thus, to enable a powerful and efficient language model, we consider searching a hybrid backbone to replace the attention-only architecture by adding local operations into our inter-layer search space. Specifically, we incorporate the lightweight convolution as our candidate operation, since its effectiveness has been proven in NLP tasks such as machine translation (Wu et al. 2018). To explore whether different reception fields are preferred for different layers, we further allow different kernel sizes ($3 \times 1, 5 \times 1, 7 \times 1, 9 \times 1, 15 \times 1, 31 \times 1, 65 \times 1$) across layers. For each convolution layer, the projected input is followed by a Gated Linear Unit (GLU) (Dauphin et al. 2017), as shown in Figure 2.

(2) Macro Search Space. We adopt macro search space for the backbone architecture. Specifically, we allow each layer to have different searched self-attention structure and convolution block. Comparing with the micro (cell-based) search space adopted in previous works (Liu, Simonyan, and Yang 2018; Shi et al. 2020), from which a cell structure is searched and the backbone is constructed by repeatedly stacking the cell, our search space is much more flexible, which has more than 10^{20} possible combinations. As a result, the searched backbone architecture is more efficient and can effectively capture both global and local contexts.

3.2 Operation-Priority Neural Architecture Search Algorithm (OP-NAS)

Since we search for new architectures from scratch in an extremely large macro search space, which involves both intra-layer and inter-layer level, our NAS algorithm must be efficient, scalable, and computationally feasible. Though gradient-based search algorithms such as DARTS are attractive due to their search speed, they do not fit our demand for exploring novel attention mechanism with more flexibility. The supernet in gradient-based algorithms needs to store all the intermediate variables for gradient updates, which requires huge memory cost. This drawback hinders their application on our search space, since we do not restrict the length of attention path and allow a large number of possible operation combinations.

Evolution algorithms (EA) (Real et al. 2019) poses less constraints over the search space as per our request. However, traditional EA suffers from the risk of being trapped by local optimal in a huge search space. To this end, we propose an operation-priority (OP) acquisition method to improve the efficiency of model search by balancing exploration and exploitation. Furthermore, we propose Bi-branch Weight-Sharing (BIWS) training strategy to boost model evaluation by preventing repeated pretraining. The details are described in Algorithm 1.

Algorithm 1: OP-NAS Algorithm.

1: Initialize population $\mathcal{M}$ from search space $\mathcal{A}$;
2: Model evaluation in $\mathcal{M}$;
3: **repeat**
4:     $\mathcal{P} \leftarrow$ Top-K ($\mathcal{M}$);
5:     **for** each parent p in $\mathcal{P}$ **do**
6:         $p' \leftarrow Mutation_{InterLayer}(p)$;
7:         $c \leftarrow Mutation_{IntraLayer}(p',\text{UCB})$;
8:         Initialize c with BIWS strategy ;
9:         Evaluate c on the proxy task;
10:     **end for**
11:     Update $\mathcal{M}$ with the newly evaluated children.
12:     Update UCB scores by Equation (3);
13: **until** convergence

Operation-priority Evolution Strategy Our OP-NAS is an evolution-based search algorithm. Specifically, it begins by randomly sampling candidates and evaluating them to initialize the population $\mathcal{M}$. In every iteration, the top-K individuals in $\mathcal{M}$ are treated as the $parents$ to generate the $children$ via mutation. In inter-layer level, the parent follows the vanilla EA (Goldberg and Deb 1991) to perform random mutation. In intra-layer level, however, random mutation leads to severe inefficiency when searching for attention structures, as there are many possible operation combinations and the length of attention path is unconstrained.

To address the aforementioned issue, we leverage the prior information of each operation when performing intra-layer mutation. The greedy assumption is that if a model performs well, then the operations in its architecture (path) are promising, which should have a higher chance to be sampled. However, the algorithm should also encourage the less frequently sampled operations to prevent getting trapped in local optimal. Thus, we adopt the upper confidence bound (UCB) (Auer, Cesa-Bianchi, and Fischer 2002) acquisition function, which balances exploitation and exploration to enhance the search efficiency and reduce the number of candidates that need to be evaluated.

In contrast to previous methods which utilize acquisition functions to measure the potential of whole architectures (Li et al. 2017; Shi et al. 2020), while the mutation is still performed randomly, our method uses the UCB acquisition function as a metric to guide the operation selection on each position during mutation. Our method is therefore more efficient and flexible, as the prior knowledge of each operation can be harnessed to generate promising children. For operation i, the UCB score u_i is calculated as:

$$u_i = \mu_i + \alpha\sqrt{2\log N/N_i} \qquad (3)$$

where μ_i is the average proxy task score of the enumerated paths where operation i is included, α is the hyperparameter controlling the level of exploration, N_i is the number of times that operation i has been sampled and N is the total number of operations sampled in history. When the operation is infrequently sampled, the right part dominates the score function.

As opposed to other NAS methods such as DARTS (Liu, Simonyan, and Yang 2018) and ENAS (Pham et al. 2018), whose architecture path lengths are fixed, the length of our attention path is flexible and is allowed to change during the search. Thus, assigning independent probability distributions for operations at each position is not feasible, as the position may shift due to the change of path length. To tackle this problem, we model n probability distributions, where n is the length of the longest path sampled during the search. For parent path of length k, the child path is always mutated based on the first k distributions. For convolution layers, the empirical probability distribution for different kernel sizes can be directly calculated for each layer. The probabilities for operations (or kernel sizes) are calculated as: $p_1, \ldots, p_n = \text{softmax}(u_1, \ldots, u_n)$, where u_i represents the UCB score for operation i.

Bi-branch Weight-Sharing (BIWS) Training Strategy
To avoid the repeated pretraining of candidate models, we design BIWS training strategy to speed up the model evaluation. Note that even using a very reduced training scheme, evaluating one architecture by training from scratch requires 200 GPU hours. With our BIWS, the evaluation cost is greatly reduced by 80%. The main idea of our strategy is to reuse the trained model parameters in the previous round of searching. To achieve this, we first introduce a bi-branch super-net which contains the largest set of the possible candidate models: one branch contains max attention structure (4 input nodes), and the other branch contains the largest convolution structure (kernel size = 65×1). Each candidate model is initialized by the parameters fetched from the corresponding layers and positions of the super-net. In this way, we can obtain evaluation results with high fidelity after only a few epochs of fine-tuning. To enable a reusable super-net, we design the following strategies:
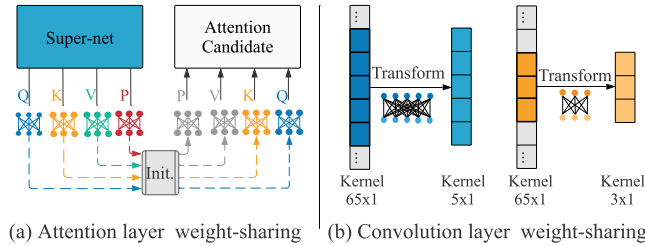

(a) Attention layer weight-sharing (b) Convolution layer weight-sharing

Figure 3: Illustration of BIWS strategy. For attention layers, transformation matrices of K,Q are initialized from corresponding positions of the largest 4-node attention. For convolution layers, small kernels are initialized by the center of the largest kernel.

(1) Convolution layer weight-sharing. Inspired by (Cai et al. 2019), we maintain the weights for the largest convolution layer (kernel size = 65×1) throughout searching,
then the weights at the center position are shared to initialize the small kernels for the candidate models (as shown in Figure 3). Since the shared weights play multiple roles when they are applied to sub-kernels of various sizes, the weights in those sub-kernels should have different properties of distributions and magnitudes. To this end, we introduce the kernel transformation matrices to adapt the shared weights for sub-kernels of different sizes. Specifically, different kernel transformation matrices are learnt during training for different layers, while being shared across all the channels within each layer. The weights of the sub-kernels are updated to the largest kernel in the super-net after training the candidate models in each round.

(2) Attention layer weight-sharing. The parameters in self-attention structure lie in the linear transformation matrices for *key, query, value* and P. Since we only mutate parts of the computation graph in each round of searching, we can directly initialize these fully-connected layers in the child individuals using the weights extracted from the corresponding layers of the super-net.

4 Experiments
4.1 Dataset and Setting
Datasets and metrics. We first pre-train the backbone architectures using a large corpus of text data and then fine-tune the model for each specific downstream task. For pre-training, we use the BooksCorpus (Zhu et al. 2015) and English Wikipedia (Devlin et al. 2019). For finetuning and evaluation, we use the General Language Understanding Evaluation (GLUE) (Wang et al. 2018) and the Stanford Question Answering Dataset (SQuAD) (Rajpurkar et al. 2016). Unless stated otherwise, downstream tasks are reported using the same metrics in BERT (Devlin et al. 2019). For other settings, we follow the settings of BERT paper.

Implementation Details. We use Masked Language Model (MLM) and Next Sentence Prediction (NSP) as pre-training tasks. The whole process can be divided into two phases, namely the NAS phase and the fully-train phase. For NAS phase, we train the base model, whose configuration is the same as BERT-base ($L = 12, H = 768, A = 12$). Initial $\mathcal{M}$ is set as 100, and K is set as 5. Each parent will mutate 5 child architectures. In the NAS phase, we train each candidate architecture for 40,000 steps, which is then evaluated on the proxy task (GLUE). The searching phase costs around 24K GPU hours (760+ evaluated candidates) on Nvidia V100. If we only use EA without BIWS strategy, the computation cost is estimated to be about 182K GPU hours. In fully-train phase, we first pre-train the searched base-size model. To further verify the model's scaling ability, we also fully-train the model on small model ($L = 12, H = 256, A = 4$) and large model ($L = 24, H = 1024, A = 16$). Specifically, we treat each two continuous layers as a block and expand the base model to large model by inserting the same block after the original one. More details are attached to Appendix.

4.2 Results and Analysis
Structure Analysis of AutoBERT-Zero. We name the best searched architecture of OP-NAS **AutoBERT-Zero**. As

	#Params	Infer FLOPs	CoLA	MRPC	MNLI-(m/mm)	STS-B	RTE	QQP	QNLI	SST-2	AVG
Development Set											
BERT-base(ours)	110M	2.9e10	58.1	89.7	84.8/85.2	88.8	69.0	88.2	91.5	92.9	83.1
AutoBERT-att	104M	2.3e10	65.4	92.2	84.6/85.0	90.4	81.6	88.5	91.8	93.8	85.9
AutoBERT-conv	104M	2.2e10	63.8	92.6	84.4/84.6	90.1	80.5	88.3	91.7	93.5	85.5
AutoBERT-w/o-desc	104M	2.3e10	65.1	92.8	84.5/85.0	90.5	78.7	88.2	91.6	93.7	85.6
AutoBERT-Zero	104M	2.3e10	64.5	93.3	85.5/85.3	90.8	81.9	88.9	92.0	94.2	86.3
AutoBERT-Zero*	104M	2.3e10	**67.3**	**93.8**	**86.4/86.3**	**90.8**	**85.2**	**91.7**	**92.5**	**95.2**	**87.7**
Test Set											
GPT(Radford et al. 2018)	117M	3.0e10	45.4	82.3	82.1/81.4	82.0	56.0	70.3	88.1	91.3	75.4
BERT-base(Devlin et al. 2019)	110M	2.9e10	52.1	88.9	84.6/83.4	85.8	66.4	71.2	90.5	93.5	79.6
DynaBERT-base(Hou et al. 2020)	110M	2.9e10	54.9	87.9	84.5/84.1	84.4	69.9	72.1	91.3	93.0	80.2
ConvBERT-base (Jiang et al. 2020)	106M	2.7e10	53.7	89.3	84.6/83.6	86.1	72.1	71.3	90.1	93.5	80.5
Roberta-base (Liu et al. 2019b)	110M	2.9e10	50.5	90.0	86.0/85.4	88.1	73.0	70.9	92.5	94.6	81.1
BERT-Large(Devlin et al. 2019)	340M	8.7e10	60.5	89.3	86.7/89.5	86.5	70.1	72.1	92.7	94.9	82.1
AutoBERT-Zero	104M	2.3e10	55.9	89.5	85.4/84.9	88.3	77.8	71.8	91.2	94.6	82.2
AutoBERT-Zero*	104M	2.3e10	**59.5**	**90.5**	**86.1/86.0**	**88.9**	**80.2**	**72.8**	**92.1**	**95.1**	**83.5**
AutoBERT-Zero-Large	318M	6.8e10	**63.8**	**90.7**	**87.7/87.1**	**90.1**	**80.4**	**72.1**	**93.6**	**95.4**	**84.5**

Table 2: Performance comparison on the test set of GLUE. Our 12-layer base model AutoBERT-Zero significantly surpasses RoBERTa-Base and BERT-large (24 layers). Note that Roberta (Liu et al. 2019b) runs on 160G corpus, whereas our model runs on 16G corpus. Infer FLOPs assumes single inputs with length 128. AutoBERT-Zero* is initialized from the surpernet.
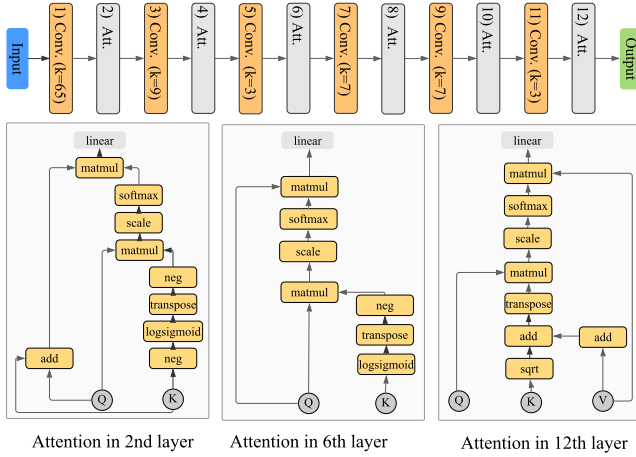

Figure 4: The detailed architecture of AutoBERT-Zero. We only show the 2nd, 6th and 12th discovered attention structures due to limited space. Att. and Conv. represents the searched attention layer and convolution layer respectively. The whole backbone is attached to the Appendix.

shown in Figure 4, the hybrid backbone of AutoBERT-Zero is constructed with stacked conv-att blocks (searched convolution followed by searched attention layer), which effectively integrates the local and global dependencies of natural language. For the searched attentions, V is shared with Q/K in shallow layers, but non-shared in the deeper layers. This is reasonable since the shallow layer only process the low-level features, whereas the deep layers need more parameters to capture the complex semantic features. For example, $\hat{Attn}(X)_{L_2}$ introduces K-V and Q-V sharing mechanisms, while $\hat{Attn}(X)_{L_{12}}$ adopts separate weights for K, Q and V:

$$\hat{Attn}(X)_{L_2} = \sigma(Q \log(1 + \exp(K^\top))/\sqrt{d_h})(K + Q)W_O^\top.$$
$$\hat{Attn}(X)_{L_{12}} = \sigma(Q(K/\sqrt{d_h} + V)^\top/\sqrt{d_h})VW_O^\top.$$

Besides, the kernel sizes of convolution layers roughly follow a descending order (changing from 65 to 3), which indicates the convolution layers learn local information from wide to narrow. This is justifiable as the a larger receptive field captures more information, which helps emphasize

	#Params of Att	SQuAD v1.1		SQuAD v2.0	
		EM	F1	EM	F1
BERT-base(ours)	21.3M	78.9	86.7	70.2	72.8
AutoBERT-att	15.9M	79.7	87.5	**72.9**	**75.7**
AutoBERT-conv	15.4M	79.1	86.5	71.9	74.6
AutoBERT-w/o-desc	15.4M	79.5	87.0	71.5	73.9
AutoBERT-Zero	15.4M	**79.9**	**87.6**	72.5	75.0

Table 3: Results on SQuAD(dev). "#Params of Att" counts parameters in attention structures.

	#Params	FLOPs	Pre-train Task	GLUE
ELMO (Peters et al. 2018)	96M	2.6e10	LM	71.2
GPT(Radford et al. 2018)	117M	3.0e10	LM	78.8
BERT-small (Jiang et al. 2020)	14M	3.7e9	MLM	75.1
ELECTRA-small (Clark et al. 2019)	14M	3.7e9	RTD	79.9
ConvBERT-small (Jiang et al. 2020)	14M	4.1e9	MLM	75.9
AutoBERT-Zero-small	13M	2.9e9	MLM	**80.5**
BERT-large(Devlin et al. 2019)	340M	8.7e10	MLM	84.4
AutoBERT-Zero-large	318M	6.8e10	MLM	**87.9**

Table 4: Scaling ability of the searched model. Results are reported on GLUE dev set.[2]

on the informative features while suppress the unimportant ones. After the shallower layers effectively reduce the information redundancy, the deeper layers can focus on the important semantic features.

Results on GLUE & SQuAD. After the NAS phase, the searched models are fully-trained and evaluated on downstream tasks. Our AutoBERT-Zero consistently outperforms other baselines by a large margin. To demonstrate the superiority of AutoBERT-Zero's structure, we fully-train several other searched backbones for comparison: (*i*) **AutoBERT-w/o-desc.** A backbone without descending kernel sizes for convolution layers. (*ii*) **AutoBERT-att.** A backbone containing three continuous attention layers. (*iii*) **AutoBERT-conv.** A backbone containing three continuous convolution layers. The details of architectures can be found in Appendix. As shown in Table 2, AutoBERT-Zero achieves the highest GLUE score, with a significant performance gain over BERT-base while having less parameters and FLOPs. Specifically, AutoBERT-Zero performs much better than AutoBERT-att and AutoBERT-conv, demonstrating

---

[2]Following ConvBERT, we count accuracy for MRPC and QQP for small model. Small model results are median results of 3 runs.

	#Params of Att	CoLA	MRPC	MNLI-(m/mm)	STS-B	RTE	QQP	QNLI	SST-2	AVG
BERT-base	21.3M	58.1	89.7	84.8/85.2	88.8	69.0	88.2	91.5	92.9	83.1
Att-only	16.5M	60.0	92.1	84.9 /84.1	90.6	79.4	88.3	91.5	92.5	84.8
Conv-only	15.4M	53.7	82.9	69.0/66.1	81.0	64.2	82.0	75.7	86.7	73.3
AutoBERT-Zero	15.4M	**64.5**	**93.3**	**85.5/85.3**	**90.8**	**81.9**	**88.9**	**92.0**	**94.2**	**86.3**

Table 5: Model comparison among AutoBERT-Zero and its Attention-only, Conv-only variants. Models are fully-trained and evaluated on GLUE dev set.
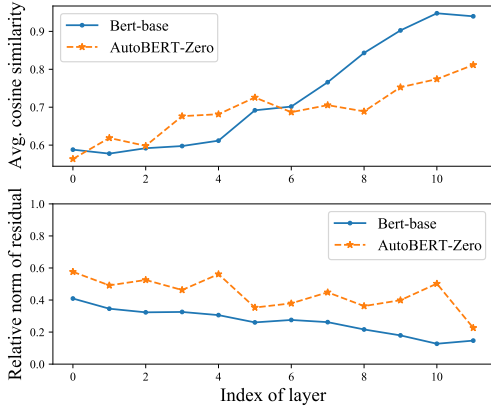

Figure 5: Oversmooth exists in purely-stacked transformer structure, whereas the hybrid structure helps to tackle this problem.

Figure 6: Searching performance comparison among Random Search (RS), RS with weight sharing, EA with weight sharing and OP-NAS.

that the conv-att block can better integrate the local and global dependencies. Besides, AutoBERT-Zero's advantage over AutoBERT-w/o-desc indicates that the kernel size pattern from wide to narrow in convolution layers benefits the performance. As shown in Table 3, AutoBERT-Zero consistently surpasses BERT-base on both SQuAD v1.1 and v2.0, demonstrating the generalizibility of our searched model.

Representation ability of AutoBERT-Zero. "Token-uniformity" damages model's representation ability. To measure the degree of "token-uniformity", following (Dong, Cordonnier, and Loukas 2021; Gong et al. 2021), we use *relative norm of residual* to measure the rank of output (the rank is equal to 1 when *residual* is equal to 0), and measure the average *pairwise cosine-similarity* between the representations of different tokens on 1,280 samples of STS-B. As shown in Figure 5, latent representations from purely-stacked BERT-base have high similarity, and the rank of output is closer to 1 (*relative norm of residual* is closer to 0), showing no significant difference between the tokens. On the other hand, the output of AutoBERT-Zero has relatively larger *residual* and lower token similarity, showing that the hybrid backbone helps mitigate this problem.

Scaling ability of AutoBERT-Zero. We further extend AutoBERT-Zero structure to different capacities, showing strength in both large and small models. Specifically, Table 4 shows that our large model surpasses BERT-large by 3.5 in GLUE. Remarkably, our small model significantly surpasses the SOTA ConvBERT-small (4.6 higher) and BERT-small (5.4 higher) under the vanilla MLM task. Besides, our small model considerably outperforms the large GPT in terms of both performance and complexity: 1.7 higher GLUE, 88% less parameters, and 90% less FLOPs. Despite the advantage of strong pre-train task (RTD), ELECTRA-small is still outperformed by our model.
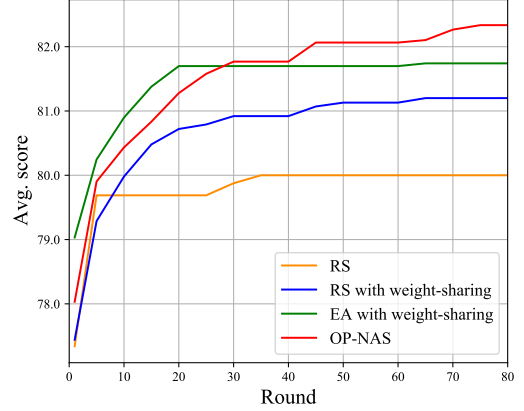
The Efficiency of OP-NAS. During the search, we observe that by adopting the proposed operation-priority strategy, the exploration ability of the EA is highly improved, which prevents getting trapped in local optimal (see Figure 6). The results shows that searched model using OP-NAS outperforms other NAS algorithms by a large margin. As the quality of model evaluation during NAS phase greatly impacts the algorithm's effectiveness, we further examine the fidelity of the evaluation results. Kendall (Kendall 1938) correlation analysis is performed to evaluate the correlation between model performances in the NAS phase and fully-train phase. As shown in Appendix B, high correlations are captured in most of the downstream tasks, which is owing to the effectiveness of our BIWS strategy.

Ablation study. To investigate the superiority of searched hybrid architecture, we evaluate performance of attention-only and convolution-only variants, which are constructed by stacking either the searched attention or the convolution layers of AutoBERT-Zero. For example, for the attention-only variant, each convolution block is replaced with the attention layer directly behind it. From Table 5, we find that the hybrid backbone architecture outperforms both attention-only and convolution-only variants. Besides, the attention-only variant surpasses BERT-base by a large margin, showing effectiveness of searched attention structures.

5 Conclusion

In this work, we propose a novel hierarchical search space and an efficient NAS framework to automatically find promising PLM backbones from scratch, which prevents the tedious manual tuning. The searched self-attention structure and backbone architecture can inspire new insights for model design in the NLP community.

Acknowledgments

References

Auer, P.; Cesa-Bianchi, N.; and Fischer, P. 2002. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2): 235–256.

Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Cai, H.; Gan, C.; Wang, T.; Zhang, Z.; and Han, S. 2019. Once-for-All: Train One Network and Specialize it for Efficient Deployment. In *ICLR*.

Chen, D.; Li, Y.; Qiu, M.; Wang, Z.; Li, B.; Ding, B.; Deng, H.; Huang, J.; Lin, W.; and Zhou, J. 2020. AdaBERT: Task-Adaptive BERT Compression with Differentiable Neural Architecture Search. In *IJCAI*.

Clark, K.; Luong, M.; Le, Q.; and Manning, C. 2019. ELEC-TRA: Pre-training Text Encoders as Discriminators Rather Than Generators. In *ICLR*.

Dauphin, Y.; Fan, A.; Auli, M.; and Grangier, D. 2017. Language modeling with gated convolutional networks. In *ICML*.

Devlin, J.; Chang, M.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL*.

Dong, L.; Yang, N.; Wang, W.; Wei, F.; Liu, X.; Wang, Y.; Gao, J.; Zhou, M.; and Hon, H. 2019. Unified Language Model Pre-training for Natural Language Understanding and Generation. In *NeurIPS*.

Dong, Y.; Cordonnier, J.; and Loukas, A. 2021. Attention is not all you need: Pure attention loses rank doubly exponentially with depth. *arXiv preprint arXiv:2103.03404*.

Goldberg, D.; and Deb, K. 1991. A comparative analysis of selection schemes used in genetic algorithms. In *Foundations of genetic algorithms*.

Gong, C.; Wang, D.; Li, M.; Chandra, V.; and Liu, Q. 2021. Improve Vision Transformers Training by Suppressing Over-smoothing. *arXiv preprint arXiv:2104.12753*.

Hou, L.; Huang, Z.; Shang, L.; Jiang, X.; Chen, X.; and Liu, Q. 2020. DynaBERT: Dynamic BERT with Adaptive Width and Depth. In *NeurIPS*.

Jiang, Z.; Yu, W.; Zhou, D.; Chen, Y.; Feng, J.; and Yan, S. 2020. ConvBERT: Improving BERT with Span-based Dynamic Convolution. *NeurIPS*.

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

Kitaev, N.; Kaiser, L.; and Levskaya, A. 2020. Reformer: The Efficient Transformer. In *ICLR*.

Kovaleva, O.; Romanov, A.; Rogers, A.; and Rumshisky, A. 2019. Revealing the Dark Secrets of BERT. In *EMNLP*.

Lan, Z.; Chen, M.; Goodman, S.; Gimpel, K.; Sharma, P.; and Soricut, R. 2020. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. In *ICLR*.

Li, L.; Jamieson, K.; DeSalvo, G.; Rostamizadeh, A.; and Talwalkar, A. 2017. Hyperband: A novel bandit-based approach to hyperparameter optimization. *The JMLR*, 18(1): 6765–6816.

Liu, H.; Simonyan, K.; and Yang, Y. 2018. DARTS: Differentiable Architecture Search. In *ICLR*.

Liu, P.; El Basha, M. D.; Li, Y.; Xiao, Y.; Sanelli, P. C.; and Fang, R. 2019a. Deep evolutionary networks with expedited genetic algorithms for medical image denoising. *Medical image analysis*, 54: 306–315.

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

Michel, P.; Levy, O.; and Neubig, G. 2019. Are Sixteen Heads Really Better than One? In *NeurIPS*.

Peters, M. E.; Neumann, M.; Iyyer, M.; Gardner, M.; Clark, C.; Lee, K.; and Zettlemoyer, L. 2018. Deep contextualized word representations. In *NAACL*.

Pham, H.; Guan, M.; Zoph, B.; Le, Q.; and Dean, J. 2018. Efficient Neural Architecture Search via Parameter Sharing. In *ICML*.

Radford, A.; Narasimhan, K.; Salimans, T.; and Sutskever, I. 2018. Improving language understanding by generative pre-training.

Rajpurkar, P.; Jia, R.; and Liang, P. 2018. Know What You Don't Know: Unanswerable Questions for SQuAD. In *ACL*.

Rajpurkar, P.; Zhang, J.; Lopyrev, K.; and Liang, P. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *EMNLP*.

Real, E.; Aggarwal, A.; Huang, Y.; and Le, Q. 2019. Regularized evolution for image classifier architecture search. In *AAAI*.

Real, E.; Liang, C.; So, D.; and Le, Q. 2020. AutoML-zero: evolving machine learning algorithms from scratch. In *ICML*.

Shi, H.; Pi, R.; Xu, H.; Li, Z.; Kwok, J.; and Zhang, T. 2020. Bridging the Gap between Sample-based and One-shot Neural Architecture Search with BONAS. In *NeurIPS*.

So, D.; Le, Q.; and Liang, C. 2019. The Evolved Transformer. In *ICML*.

Tay, Y.; Bahri, D.; Metzler, D.; Juan, D.; Zhao, Z.; and Zheng, C. 2021. Synthesizer: Rethinking self-attention in transformer models. In *ICML*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *NeurIPS*.

Wang, A.; Singh, A.; Michael, J.; Hill, F.; Levy, O.; and Bowman, S. 2018. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In *EMNLP Workshop BlackboxNLP*.

Wu, F.; Fan, A.; Baevski, A.; Dauphin, Y.; and Auli, M. 2018. Pay Less Attention with Lightweight and Dynamic Convolutions. In *ICLR*.

Xu, H.; Yao, L.; Zhang, W.; Liang, X.; and Li, Z. 2019. Auto-fpn: Automatic network architecture adaptation for object detection beyond classification. In *ICCV*.

Yang, Z.; Dai, Z.; Yang, Y.; Carbonell, J.; Salakhutdinov, R.; and Le, Q. 2019. XLNet: Generalized Autoregressive Pretraining for Language Understanding. In *NeurIPS*.

Yao, L.; Pi, R.; Xu, H.; Zhang, W.; Li, Z.; and Zhang, T. 2021. Joint-DetNAS: Upgrade Your Detector with NAS, Pruning and Dynamic Distillation. In *CVPR*.

Zellers, R.; Bisk, Y.; Schwartz, R.; and Choi, Y. 2018. SWAG: A Large-Scale Adversarial Dataset for Grounded Commonsense Inference. In *EMNLP*.

Zhu, H.; An, Z.; Yang, C.; Xu, K.; Zhao, E.; and Xu, Y. 2019. EENA: efficient evolution of neural architecture. In *ICCV Workshops*.

Zhu, Y.; Kiros, R.; Zemel, R.; Salakhutdinov, R.; Urtasun, R.; Torralba, A.; and Fidler, S. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *ICCV*.

Zoph, B.; and Le, Q. 2016. Neural architecture search with reinforcement learning. In *ICLR*.