AutoAttend: Automated Attention Representation Search

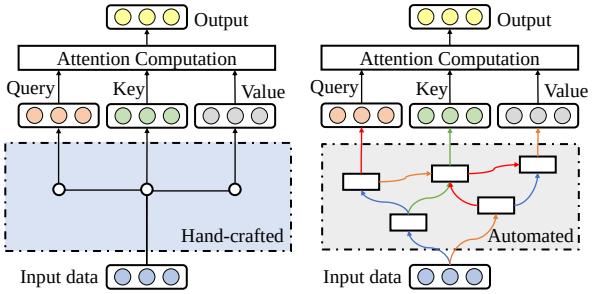
Chaoyu Guan [1] **Xin Wang** [1] **Wenwu Zhu** [1]

Abstract

Self-attention mechanisms have been widely adopted in many machine learning areas, including Natural Language Processing (NLP) and Graph Representation Learning (GRL), etc. However, existing works heavily rely on hand-crafted design to obtain customized attention mechanisms. In this paper, we automate *Key*, *Query* and *Value* representation design, which is one of the most important steps to obtain effective self-attentions. We propose an automated self-attention representation model, **AutoAttend**, which can automatically search powerful attention representations for downstream tasks leveraging Neural Architecture Search (NAS). In particular, we design a tailored search space for attention representation automation, which is flexible to produce effective attention representation designs. Based on the design prior obtained from attention representations in previous works, we further regularize our search space to reduce the space complexity without the loss of expressivity. Moreover, we propose a novel context-aware parameter sharing mechanism considering special characteristics of each sub-architecture to provide more accurate architecture estimations when conducting parameter sharing in our tailored search space. Experiments show the superiority of our proposed AutoAttend model over previous state-of-the-arts on eight text classification tasks in NLP and four node classification tasks in GRL.

1. Introduction

Self-attention mechanisms have become more and more popular in the design of Deep Neural Networks (DNNs) to achieve good performance. With the ability to help deep models selectively focus on useful information, self-attention has been proved powerful and widely used in all kinds of research directions, including Natural Language Processing (NLP) (Mittal et al., 2020; Zhou et al., 2020), Graph Representation Learning (GRL) (Sankar et al., 2020; Zhang et al., 2020), Computer Vision (CV) (Li et al., 2020; Sun et al., 2020), etc.



(a) hand-crafted attention representation (b) automated attention representation

Figure 1. An illustration of attention representation design in self-attention. (a) Traditional hand-crafted attention representation, where the computation paths are manually designed to derive *Key*, *Query*, and *Value*. (b) Our automated attention representation, where the computation paths from the input to *Key*, *Query* and *Value* are automatically searched leveraging NAS.

Typical self-attention can be regarded as extracting useful information in *Value* according to *Key* and *Query*, where *Key*, *Query*, and *Value* are different representations of the input data. Proper self-attention representations (*Key*, *Query*, and *Value*) can greatly boost the model performance (Daniluk et al., 2017; Niu et al., 2019; Dai et al., 2019). However, as shown in Figure 1(a), all previous works rely solely on manual design to obtain self-attention representations, where various functional components such as CNN, RNN, GAT, and GCN are manually stacked or combined to derive *Key*, *Query*, and *Value* from the input data (Lin et al., 2017; Vaswani et al., 2017; Velickovic et al., 2018). These hand-crafted representation designs cost huge trail-and-error human labors to derive and are sub-optimal because of human bias, thus making them hard to fit in real-world applications.

To solve this problem, we propose to automate self-attention representation through neural architecture search (NAS) in this paper (Figure 1(b)). However, directly applying the

[1]Department of Computer Science and Technology, Tsinghua University . Correspondence to: Xin Wang <xin_wang@tsinghua.edu.cn>, Wenwu Zhu <wwzhu@tsinghua.edu.cn>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

existing NAS frameworks to automate the attention representation design has the following two challenges: (1) **How to obtain the most suitable search space?** A proper search space for attention representation should i) support jointly searching for both attention representations and other functional components to achieve global optimal results, ii) flexible enough to cover most state-of-the-art (SOTA) attention representations and iii) have low complexity to ease the search. (2) **How to consider the special characteristics of each sub-architecture in parameter sharing?** In the search space mentioned above, even the same set of parameters can have different functionalities when processing or outputting tensors with different meanings (e.g., *Key*, *Query*, *Value*, etc.). Directly applying the widely used parameter sharing (Pham et al., 2018) without considering these special characteristics may fail to provide reliable architecture evaluations.

To address these challenges, we propose an automated attention search approach, **AutoAttend**, to search for models with the best attention representations. We treat DNN as a set of connected layers, then reformulate attention representation as a source layer selection and an operation selection process to construct a flexible, expressive and unified search space tailored for self-attention. By utilizing the design prior from previous hand-crafted attentions, we further regularize the search space in order to reduce the space complexity without the loss of expressivity, where n is the number of layers. We employ one-shot formulation (Bender et al., 2018; Guo et al., 2020) to search for the best architectures in the proposed search space, and develop a context-aware parameter sharing mechanism to offer reliable architecture evaluations. Such a mechanism can take characteristics of each sub-architecture into account by sharing parameters only when they have the same contexts. Particularly, we define *context* to be the layer functionalities that one operation connects with. We conduct extensive experiments over several NLP and GRL tasks, which have been known to benefit a lot from hand-crafted attention mechanisms (Vaswani et al., 2017; Velickovic et al., 2018; Devlin et al., 2019b). Specifically, we focus on text classification tasks in NLP and transductive and inductive node classification tasks in GRL. Experimental results show that our proposed AutoAttend model outperforms or is on par with the previous SOTA models under the same experimental settings. Ablation studies on our proposed attention layer and context-aware parameter sharing mechanism also demonstrate their effectiveness and necessity.

In summary, we make the following contributions:

- We propose an automated attention representation model, **AutoAttend**, to search for the best self-attention representation design, to the best of our knowledge, for the first time[1].
- We propose a tailored search space that supports jointly searching for attention representations as well as other functional components to achieve global optimal results with a low space complexity.
- We propose a context-aware parameter sharing mechanism capable of providing reliable architecture evaluations for parameter sharing in our tailored search space, by taking special characteristics of each architecture into consideration.
- Extensive experiments demonstrate the advantages of our AutoAttend approach against state-of-the-art NAS approaches over eight text classification tasks in NLP and four node classification tasks on GRL.

We organize our paper as follows. We first review the related work in Section 2. Then, we give the definition of attention mechanism and neural architecture search, and formulate the attention representation search problem definition in Section 3. The detailed AutoAttend framework, including attention representation search space and algorithm design, is given in Section 4. We present extensive comparisons with previous state-of-the-art hand-crafted and searched architectures, and ablate the AutoAttend framework in Section 5, and discuss the conclusion and future work in Section 6.

2. Related Work

In this section, we review the related works on attention representation design and neural architecture search.

2.1. Representation Design for Self-Attention

There are two main components in self-attention: attention representation and attention computation. While the latter component receives heavier research interests recently (Shaw et al., 2018; Dai et al., 2019; Kitaev et al., 2020), the former is neglected in recent researches, which is as important as the latter and is the main focus of this paper.

The concept of attention is firstly proposed in neural machine translation (Bahdanau et al., 2015) to align target and source sentences when translating. Then, it is adopted and widely applied as self-attention to model the intra-modal relation of data (Lin et al., 2017; Vaswani et al., 2017; Dai et al., 2019). Lin et al. (2017) propose to leverage a shared LSTM layer to derive the attention representations. Daniluk et al. (2017) further separate the *Key* and *Value* layer to solve the optimization problems, which is then adopted by Mino et al. (2017) for machine translation. Vaswani et al. (2017) abandon the traditional recurrent structure and use only the intra-layer self-attention as building blocks, which is then widely used as a simple and portable attention repre-

[1]Code will be published at https://github.com/THUMNLab/AutoAttend

sentation design style in many areas (Velickovic et al., 2018; Devlin et al., 2019a; Brown et al., 2020; Carion et al., 2020). Niu et al. (2019) propose to use self-attention to fuse local and global information for better sentence representation, where hidden states from local encoder are used as *Query* to attend on *Key*, *Value* from global encoder. Ma et al. (2019) further propose to deeply fuse local and global information by attending in both directions, where global hidden states can also act as *Query* to attend on *Key* and *Value* from local encoder.

Unlike previous methods that depend on human expert knowledge, we propose the first automatic framework to design self-attention representation leveraging NAS.

2.2. Neural Architecture Search

Neural architecture search (NAS) aims at searching for the best architectures for given tasks and becomes more and more popular in recent years (Zoph & Le, 2017; Pham et al., 2018; Liu et al., 2019b; Mei et al., 2020). Various search algorithms like Reinforcement Learning (Zoph & Le, 2017; Pham et al., 2018), Evolutionary Algorithm (Zoph et al., 2018; So et al., 2019; Guo et al., 2020), Gradient-based (Liu et al., 2019b; Mei et al., 2020) and Bayesian Optimization (Shi et al., 2020; Ru et al., 2020) are developed to tackle the NAS problem. Together with various kinds of space designs to search for a wide range of application domains like CV (Liu et al., 2019a; Mei et al., 2020), NLP (So et al., 2019; Wang et al., 2020b), GRL (Gao et al., 2020; Zhou et al., 2019), etc. In this paper, we aim to automate the self-attention representation using NAS.

Recently, some works (So et al., 2019; Gao et al., 2020; Zhou et al., 2019; Wang et al., 2020b; Yu et al., 2020) also utilize self-attention in their space design to search for Transformer or GNN like architectures, which is merely an application of current hand-crafted intra-layer self-attention and do not consider the attention representation design when searching. There are also some works (Wang et al., 2020a; Ma et al., 2020) focusing on how to compute attention given *Key*, *Query*, and *Value*, which can be seen as automating attention computation in Figure 1, thus are orthogonal to our work.

The search efficiency of NAS also receives huge boosts by the introduction of supernet and parameter-sharing (Pham et al., 2018; Liu et al., 2019b; Guo et al., 2020). These kinds of techniques speed up the training process by sharing parameters at the same places to avoid training each sub-architecture from scratch. However, parameter-sharing tends to eliminate the difference of operations (Chu et al., 2019) thus hard to model the special characteristics of architectures in our attention representation search space. In this paper, we propose context-aware parameter sharing to share parameters only when their contexts are the same.

3. Problem Formulation and Preliminary

3.1. Self-Attention Formulation

As shown in Figure 1, given input data $\mathbf{S}$ with n elements, the self-attention mechanism can be formulated as follows:

$$\mathbf{Q}, \mathbf{K}, \mathbf{V} = NN_Q(\mathbf{S}), NN_K(\mathbf{S}), NN_V(\mathbf{S}), \quad (1)$$

$$\mathbf{Attn_{Q\to K}} = \mathrm{Sim}(\mathbf{Q}, \mathbf{K}),$$
$$\mathbf{Out} = \mathbf{Attn_{Q\to K}}\mathbf{V}, \quad (2)$$

where NN_Q, NN_K, and NN_V are sub neural networks that derive the representation of *Query*, *Key* and *Value* $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbf{R}^{n\times d}$ from $\mathbf{S}$. The normalized attention score $\mathbf{Attn_{Q\to K}} \in \mathbf{R}^{n\times n}$ is calculated by some similarity metric defined by Sim. Every column in $\mathbf{Attn_{Q\to K}}$ stands for the normalized similarity between one query vector $\mathbf{q}_i \triangleq \mathbf{Q}[i] \in \mathbf{R}^d$ and the whole $\mathbf{K}$. The output of attention $\mathbf{Out}$ is derived by matrix production of $\mathbf{Attn_{Q\to K}}$ and $\mathbf{V}$.

This paper aims to automate the attention representation in Equation 1 (i.e., NN_Q, NN_K, NN_V), together with other functional components of networks to reach the global optimum.

3.2. NAS Preliminary

NAS aims at searching for the best architecture to model the downstream tasks. It can be formulated as a bi-level optimization problem (Liu et al., 2019b):

$$\begin{aligned} a^* &= \mathrm{argmin}_{a\in A}\, L_{val}(a, \mathbf{w}^*(a)), \\ \text{s.t. } \mathbf{w}^*(a) &= \mathrm{argmin}_{\mathbf{w}\in\mathbf{W}(a)}\, L_{train}(a, \mathbf{w}), \end{aligned} \quad (3)$$

where A stands for the search space, $\mathbf{W}(a)$ stands for the parameter space given one fixed architecture a, $\mathbf{w}^*(a)$ is the best parameters given architecture a, and a^* is the best architectures for the task and is the output of NAS.

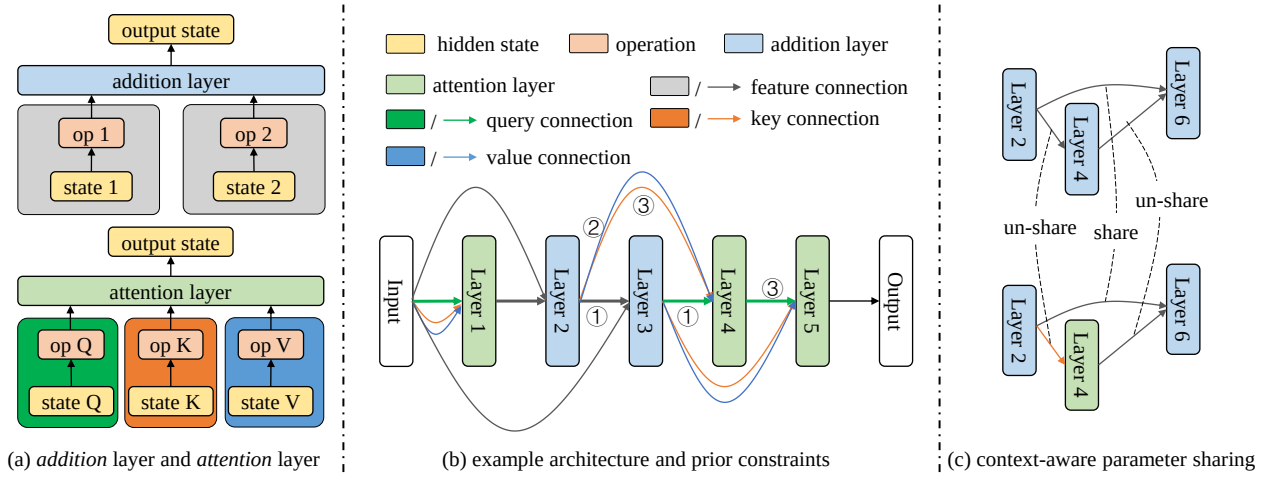
There are two key components in the NAS framework: search space A that defines the scope of possible architectures, and search algorithm that solves Equation 3.

4. Automated Attention Representation Search

In this section, we will explain in detail our AutoAttend framework that automates self-attention representation. The tailored search space is introduced in Section 4.1. The search algorithm is detailed in Section 4.2.

4.1. Attention Representation Search Space

To derive proper attention representation, one must search for attention representation and other components of DNNs altogether to reach the global optimum. Thus our search space contains both the original search space of NAS and



(a) *addition* layer and *attention* layer (b) example architecture and prior constraints (c) context-aware parameter sharing

Figure 2. **AutoAttend** framework key components. (a) Two kinds of layers in our search space. The baseline *addition* layer simply adds two connections, while the proposed *attention* layer perform attention aggregation, which is the key to automate the attention representation search. (b) An example of architecture in our search space with three constraints. The bold arrow stands for the skeleton connection in Constraint 1. ①, ② and ③ stands for Constraint 1, 2, 3 respectively. (c) Context-aware parameter sharing. Only the layer 2,4, and 6 and part of connections in two different architectures are shown. Only the connections at the same place with the same operation choice and the same context can share their weights (the connection from layer 2 to layer 6 in this example).

attention representation. We describe the original NAS search space as a baseline in Section 4.1.1, then formulate the attention layer in Section 4.1.2. Finally, prior constraints are introduced in Section 4.1.3 to reduce the complexity and redundancy without the loss of expressivity.

4.1.1. BASELINE

We first describe the baseline search space without automated attention representation. We focus on macro search space design for its ability to capture global information flow in architectures and fitness for NLP and GRL (Wang et al., 2020b; Gao et al., 2020; Zhou et al., 2019).

As shown in Figure 2(b). A modern deep neural model can be described as a set of **layers** with optional **connections** between any two layers. Here connection represents the information flow. One connection stands for a unary transform operation that can be chosen from a predefined *primitive operation pool*, which takes the feature map of the source layer as input and output another feature map for the use of the target layer. A layer, acting as an information aggregator, simply *adds* all the received outputs from the connections pointing to it, as shown in the upper of Figure 2(a). Following previous works (Zoph et al., 2018; Pham et al., 2018; Liu et al., 2019b), we bound the number of connections one layer can receive to two.

4.1.2. ATTENTION LAYER

Under the layer and connection view of DNN stated in the last section, attention can be regarded as interacting features among the same or different layers. Thus we reformulate Equation 1 as source layer and operation selection processes:

$$\mathbf{Q}, \mathbf{K}, \mathbf{V} = O_Q(\mathbf{S}_Q), O_K(\mathbf{S}_K), O_V(\mathbf{S}_V), \tag{4}$$

where $\mathbf{S}_Q$, $\mathbf{S}_K$ and $\mathbf{S}_V$ stand for the chosen source layers of $\mathbf{K}, \mathbf{Q}, \mathbf{V}$. O_Q, O_K and O_V are the selected unary operations from the *primitive operation pool*. Therefore, we automate attention representation design by introducing a new type of aggregation layer called the *attention* layer.

The *attention* layer receives three input connections. The input layers of these connections correspond to the source layer $\mathbf{S}_Q, \mathbf{S}_K, \mathbf{S}_V$, the operation choices of connections are O_Q, O_K, O_V and the outputs are the $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$. Then, the *attention* layer performs attention computation following Equation 2.

Therefore, building architecture can be described as a bunch of choices. For each layer, we need to choose the layer type and determine the source layers and operations of its connections. Given layer number n and *primitive operation pool* size b, the total number of architectures included in the defined search space is $\prod_{k=1}^{n}(k^2b^2 + k^3b^3) \in O(n!^4 b^{3n})$.

4.1.3. CONSTRAINTS

The search space defined above is difficult for searching and contains redundant, meaningless, or isomorphic architectures. We propose three main constraints to lower the complexity without hurting the expressivity. As shown in Figure 2(b).

Constraint 1 We first constrain the architecture to a chain

by forcing each layer to have at least one connection to its last layer (① in Figure 2(b). The bold connection is forced to exist). We call these kinds of connections **skeletons** because they are the main information flow of the chain networks. For the *addition* layer, we simply bind the first input connection to the last layer without the loss of generality. For the *attention* layer, we bind the *Query* connection to the last layer since *Query* in attention acts as references to reorganize information in *Key* and *Value*, which is also the common design in hand-crafted attentions (Vaswani et al., 2017; Yu et al., 2020). Despite aggressive, Constraint 1 still allows expressing most SOTA attention models. Besides, this chain-like macro search space is also widely used in previous NAS works (Guo et al., 2020; Fu et al., 2020).

Constraint 2 The second constraint binds the source layer of *Key* and *Value* connections for the *attention* layer to be the same (② in Figure 2(b). The source layer of blue and yellow connections are forced to be the same), i.e., $\mathbf{S}_K = \mathbf{S}_V$. This is because *Key* and *Value* always act as memory in previous works and should have similar semantic meanings (Niu et al., 2019; Ma et al., 2019).

Constraint 3 We further constrain the choice of operations for skeleton operations to be non-zero since a zero skeleton tends to make all the following layers meaningless. Similarly, the connections to an *attention* layer are also constraint to be non-zero to avoid meaningless attention (③ in Figure 2(b), bold connections and blue, yellow connections should be non-zero operations).

With the three constraints above, the number of searchable architectures is reduced to $\prod_{k=1}^{n}(b(b-1)k + (b-1)^3 k) \in O(n!^2 b^{3n})$, which relieves the search difficulty while still maintains high expressivity to generate powerful SOTA models.

4.2. Search Algorithm

4.2.1. One-shot Formulation

Directly solving the bi-level optimization problem in Equation 3 is highly resource exhausted because an architecture needs fully trained from scratch to solve the inner optimization problem. To make the search procedure more efficient, we separate the bi-level optimization problem into two independent optimization problems following previous works (Bender et al., 2018; Guo et al., 2020):

$$a^* = \text{argmin}_{a \in A} L_{val}(a, \mathbf{w}^*), \quad (5)$$

$$\text{s.t. } \mathbf{w}^* = \text{argmin}_{\mathbf{w} \in \mathbf{W}} \mathbb{E}_{a \sim \Gamma(A)} L_{train}(a, \mathbf{w}), \quad (6)$$

where $\Gamma(A)$ is a prior architecture distribution of $a \in A$, $\mathbf{w}$ is the weight of the supernet that contains all the architectures in the search space A. Therefore, a becomes a sub-architecture of the supernet. All architectures share the same set of parameters with the supernet, which greatly reduces the time cost since optimizing the supernet is quite faster than optimizing all the architectures from scratch.

4.2.2. Context-aware Parameter Sharing

However, directly sharing the parameters of the same operation at the same place like previous works fails to model the special characteristics of architectures, which is especially important in our case.

In our search space, the parameters of connections are highly correlated with its *contexts*: the layer choices that these connections are connected with. The meaning and way of optimization for parameters of connections that connect to the *addition* layer and the *attention* layer are quite different. Thus, we propose context-aware parameter sharing to also take the contexts of parameters as special characteristics of sub-architecture into consideration.

The key idea is that only the connections within the same contexts can share their parameters. In our search space, for one connection in the supernet, there are 4 kinds of source-target layer pairs: `add - add`, `add - attn`, `attn - add`, `attn - attn`. In addition, if the target layer is `attn`, there are three kinds of connections to distinguish (for *Query*, *Key*, and *Value*, respectively). Therefore, there are 8 kinds of contexts in total. For each connection in supernet, we assign each of its context an independent parameter to optimize them separately considering their specialties. Although this will increase the total parameters of the supernet to 8x of its origin, only a single sub-architecture is sampled and optimized for every single optimization step, whose number of parameters remains the same as origin.

For the optimization of supernet parameters in Equation 6. We use Mont-Carlo to estimate the expectation and use Gradient Descent to find the optimal solution.

4.2.3. Architecture Search

After obtaining the parameters of the supernet, we then use them as an evaluator to quickly evaluate given architectures. Following Guo et al. (2020), we adopt evolutionary search to solve the architecture optimization in Equation 5.

5. Experiments

In this section, we conduct extensive experiments and ablation studies on natural language tasks and graph tasks to demonstrate the effectiveness of the proposed attention representation search and context-aware parameter sharing.

The tasks and datasets used in this paper are introduced in Section 5.1. The implementation details are described in Section 5.2. Experimental results and analysis are shown in Section 5.3. The ablation studies on attention layer and context definition are presented in Section 5.4.

Table 1. Detailed information of natural language processing datasets used in this paper.

DATASET	#CLASS	#TRAIN	#VALID	#TEST
SST	5	8,544	1,101	2,210
SST-B	2	6,920	872	1,821
AG	4	120,000	-	7,600
DBP	14	560,000	-	70,000
YELP-B	2	560,000	-	38,000
YELP	5	650,000	-	50,000
YAHOO	10	1,400,000	-	60,000
AMZ-B	2	3,600,000	-	400,000

Table 2. Detailed information of graph representation learning datasets used in this paper.

DATASET	#CLASS	#FEATURE	#NODE	#EDGE
Transductive				
CORA	7	1,433	2,708	5,429
CITESEER	6	3,703	3,327	4,732
PUBMED	3	500	19,717	44,338
Inductive				
PPI	121	50	56,944	818,716

Table 3. Detailed primitive operations used in our experiments. For a fair comparison, we borrow the primitive operations from (Wang et al., 2020b) and (Gao et al., 2020). Common operation means these operations appear in the search space of both domains. For graph representation learning, we only show the definitions of correlation coefficient calculations.

OPERATION	DETAILED EXPLANATION
COMMON	
ZERO	Lambda x:0
NATURAL LANGUAGE PROCESSING	
IDENTITY	Lambda x:x
CONV 1	1D convolution with kernel size 1
CONV 3	1D convolution with kernel size 3
MAX POOL 3	1D max pooling with kernel size 3
GRU	Gated Recurrent Unit (Cho et al., 2014)
GRAPH REPRESENTATION LEARNING (CiteSeer and PubMed. Correlation coefficients only)	
CONST	$e_{uv}^{\text{con}} = 1$
GCN	$e_{uv}^{\text{gcn}} = 1/\sqrt{d_u d_v}$
GAT	$e_{uv}^{\text{gat}} = \text{leaky\_relu}(\mathbf{W}_l \mathbf{h}_u + \mathbf{W}_r \mathbf{h}_v)$
SYM-GAT	$e_{uv}^{\text{sym}} = e_{uv}^{\text{gat}} + e_{vu}^{\text{gat}}$
COS	$e_{uv}^{\text{cos}} = < \mathbf{W}_l \mathbf{h}_u, \mathbf{W}_r \mathbf{h}_v >$
LINEAR	$e_{uv}^{\text{lin}} = \tanh(\text{sum}(\mathbf{W}_l \mathbf{h}_u))$
GENE-LINEAR	$e_{uv}^{\text{gen}} = \mathbf{W}_a \tanh(\text{sum}(\mathbf{W}_l \mathbf{h}_u + \mathbf{W}_r \mathbf{h}_v))$

5.1. Tasks and Datasets

5.1.1. Natural Language Processing

One of the most popular application domains of self-attention is NLP. We test our AutoAttend framework on various text classification tasks, including sentiment analysis, document classification, question answering, etc.

We search for the best sentence encoder on the SST dataset and transfer it to other tasks following the settings of (Wang et al., 2020b). The detailed information of datasets we use is shown in Table 1.

More information of datasets and baseline models we compare is described in Appendix A.1.

5.1.2. Graph Representation Learning

Graph representation learning also receives huge improvement through the introduction of self-attention. We test our AutoAttend framework under two learning settings: transductive setting and inductive setting. The detailed information of datasets we use is shown in Table 2. More information about datasets and baselines we compare is described in Appendix A.2.

5.2. Implementation Details

5.2.1. Primitive Operation Pool

In this section, we give the detailed implementation of the *primitive operation pool* mentioned in Section 4.1.1. We follow (Wang et al., 2020b) to construct pools for NLP and (Gao et al., 2020) for GRL. We refer to Table 3 and Appendix B for the detailed description of all the operations leveraged. Note that for performance reasons (Gao et al., 2020), we use multi-head message passing models for dataset CiteSeer and PubMed, and use simplified stand-alone graph convolution operations for dataset Cora and PPI.

For dataset CiteSeer and PubMed, all the operations have the following multi-head message passing form:

$$\mathbf{h}_v^{out} = \sigma(\text{Merge}_{i=1}^{h_m}(\sum_{u \in N(v)} e_{uv,i} \mathbf{h}_u)), \tag{7}$$

where $\mathbf{h}_u$ stands for the input hidden state of node u. $\mathbf{h}_v^{out}$ stands for the output hidden state of node v, $N(v)$ is the neighborhood node set of node v. $e_{uv,i}$ is the correlation coefficients of the i^{th} head displaced in Table 3. h_m is the

head number and is set to 4 in all of our experiments. Merge defines how to combine representations of h_m heads, which is Sum for the last layer and Concat otherwise. σ is the activation function, which is set to $\tanh$ in our experiments.

For dataset Cora and PPI, we use the following stand-alone operations as the primitive operation pool: zero, identity, GCNConv (Kipf & Welling, 2017), SageConv (Hamilton et al., 2017), GATConv (Velickovic et al., 2018) with head number in $\{1, 2, 4, 8, 16\}$, Linear, ARMAConv (Bianchi et al., 2019), ChebConv (Defferrard et al., 2016), SGConv (Wu et al., 2019).

5.2.2. Self-Attention Implementation

For the calculation of self-attention in Equation 2, we use the simple yet powerful multi-head scaled dot-product attention proposed by Vaswani et al. (2017) for all the experiments. The head number h is a hyper-parameter and is set to 8 and 4 for NLP and GRL. To be specific, we use the following form of attention calculation:

$$\begin{aligned}
&\mathbf{T}_1, ..., \mathbf{T}_h = \mathrm{Row_Split}(\mathbf{T}), \text{for } \mathbf{T} = \mathbf{Q}, \mathbf{K}, \mathbf{V}, \\
&\mathbf{O}_i = \mathrm{Softmax}(\frac{\mathbf{Q}_i \mathbf{K}_i^T}{\sqrt{d/h}})\mathbf{V}_i, \text{for } i = 0, ..., h, \\
&\mathbf{Out} = \mathrm{Row_Cat}(\mathbf{O}_1, ..., \mathbf{O}_h),
\end{aligned} \tag{8}$$

where $\mathrm{Row\_Split}()$ and $\mathrm{Row\_Cat}()$ stand for splitting and concatenating given tensor(s) over row (the last) dimension. Note that in GRL, we only calculate attention between connected nodes by masking the energy term $\mathbf{Q}_i\mathbf{K}_i^T/\sqrt{d/h}$ according to the adjacency matrix of graphs and setting the unmasked value to $-\inf$, so that the corresponding attention score between nodes with no edge becomes 0 after $\mathrm{Softmax}$. This can be seen as a variant of GAT (Velickovic et al., 2018).

5.2.3. Training Details

For searching in NLP, we set the layer number to 24 to stay consistent with previous works. The word embeddings are initialized from pretrained GloVe (Pennington et al., 2014) and are fine-tuned during training. When searching, we use hidden size 64, batch size 128, learning rate 0.005 with Adam (Kingma & Ba, 2015), dropout 0.1, and max input sentence length 64.

After deriving the optimized supernet, we use evolution algorithm to retrieve the top 10 architectures according to their performances derived from supernet. We select the architecture with the highest validation score as the final output architecture. We then retrain the architecture with learning rate 0.0005 to report the final performance. Following (Wang et al., 2020b), we transfer the searched architecture on other text classification datasets. For a fair comparison, we basically follow (Wang et al., 2020b) to determine the re-train hyper-parameters for each dataset. The detailed re-train hyper-parameters are given in Appendix C.

For GRL, we set the layer number to 2 and 3 for transductive and inductive tasks. For transductive tasks, the dimension h is set to 64 for Cora and 256 for CiteSeer and PubMed, with learning rate 0.005, dropout 0.6, and weight decay 0.0005. For inductive PPI dataset, we set dimension h to 1024, learning rate to 0.005, and remove the dropout and weight decay following previous state-of-the-arts. Similar to NLP, we adopt evolution algorithm to find the best architectures. The only difference is that we fine-tune the best architectures from the supernet instead of training from scratch when searched from shared parameters, since we find that the performance of supernet is already competitive with previous state-of-the-arts.

For evolution settings for both NLP and GRL, we first randomly sample 500 architectures from space prior $\Gamma(A)$. Then, the top 100 architectures are selected and mutated by slightly changing the architectures through

- choosing a different operation
- choosing a different source layer
- choosing a different layer type

The newly mutated 100 architectures are added to the population. We perform the mutation 5 times in sequential in our experiment settings. Therefore, totally 1000 architectures are evaluated.

5.3. Experiment Results

5.3.1. Natural Language Processing

Table 4 shows the searched and transferred results of ours compared to previous SOTA hand-craft and searched word-level models without extra knowledges except for GloVe on NLP datasets. We report the average accuracy over 5 independent runs.

We observe that our searched architecture surpasses previous NAS results on the target SST dataset by a large margin (1.20% increase on accuracy), which demonstrate the expressivity and powerfulness of our AutoAttend framework for finding the most suitable self-attention architectures on the target dataset.

When transferred to other text classification datasets, the searched architecture also surpasses the previous best NAS results and further decreases the gap between searched and hand-crafted SOTA models, which shows that even searched in smaller datasets like the SST, the searched attention representation together with the whole sentence encoder still has the ability to generalize to similar tasks.

We also notice that the improvements on transferred datasets are not so strong (less than 1.00%) compared to the improve-

Table 4. **AutoAttend** text classification accuracy [%]. The best value of each section is shown in **bold**. The scores are averaged over 5 independent runs and accurate to 2 decimal places to compare with previous works.

MODEL	Search	Transfer						
	SST	SST-B	AG	DBP	YELP-B	YELP	YAHOO	AMZB
GUMBEL-LSTM (CHOI ET AL., 2018)	**53.70**	90.70	-	-	-	-	-	-
CAS-LSTM (CHOI ET AL., 2019)	53.60	**91.30**	-	-	-	-	-	-
DNC+CUW (LE ET AL., 2019)	-	-	93.90	99.00	96.40	65.60	74.30	-
DAGRN (LIU ET AL., 2020)	-	-	**94.93**	**99.16**	**97.34**	**70.14**	-	-
DRNN (WANG, 2018)	-	-	92.90	98.90	96.30	66.40	74.30	95.60
GELE (NIU ET AL., 2019)	-	-	93.20	99.00	96.70	67.00	**75.00**	**96.00**
24-LAYER TRANSFORMER	49.37	86.66	92.17	98.77	94.07	61.22	72.67	95.59
ENAS (PHAM ET AL., 2018)	51.55	88.90	92.39	99.01	96.07	64.60	73.16	95.80
DARTS (LIU ET AL., 2019B)	51.65	87.12	92.24	98.90	95.84	65.12	73.12	95.48
SMASH (BROCK ET AL., 2018)	46.65	85.94	90.88	98.86	95.62	65.26	73.63	95.58
ONE-SHOT (BENDER ET AL., 2018)	50.37	87.08	92.06	98.89	95.78	64.78	73.20	95.20
RANDOM (LI & TALWALKAR, 2019)	49.20	87.15	92.54	98.98	96.00	65.23	72.47	94.87
TEXTNAS (WANG ET AL., 2020B)	52.51	90.33	93.14	99.01	96.41	66.56	73.97	95.94
OURS	**53.71**	**90.50**	**93.53**	**99.08**	**96.62**	**66.82**	**74.48**	**96.04**

Table 5. **AutoAttend** graph representation learning results [%]. We report accuracy for Cora, CiteSeer, and PubMed, and F1 score for PPI. The best values of each section are shown in **bold**. The scores are averaged over 100 independent runs and accurate to 1 decimal place to compare with previous works. † means rerun by us without the leak of the test dataset.

MODEL	Transductive			Inductive
	CORA	CITESEER	PUBMED	PPI
GCN	81.5	70.3	79.5	97.7
GAT	83.1	**72.5**	79.0	97.5
ARMA	**83.4**	**72.5**	78.9	**98.5**
APPNP	83.3	71.8	**80.2**	97.8
GRAPHNAS†	80.4	73.0	80.0	98.5
AGNN	83.6	**73.8**	79.7	99.2
OURS-PS	**83.9**	72.7	79.6	98.9
OURS	**83.9**	73.0	**80.6**	**99.3**

ment on SST, meaning that a highly flexible and customized search space (like ours) may result in finding "overfit" architectures on the searched dataset. Such kinds of search spaces can find architectures with superior performances on the searched dataset, but may not be suitable for finding general architectures across different datasets or tasks when only focusing on searching for one certain dataset.

5.3.2. GRAPH REPRESENTATION LEARNING

We further test our frameworks on GRL tasks. Table 5 shows the comparison of results between AutoAttend and previous state-of-the-art hand-crafted and NAS algorithms. We report accuracy score on Cora, CiteSeer, and PubMed dataset, and F1 score on PPI dataset following previous works. All the scores are averaged over 100 independent runs.

Similar to the findings on NLP, the searched architectures (w/ or w/o parameter sharing) outperform or are on par with previous SOTA results, which demonstrates that the automated attention representation is also effective for data in the graph structure.

Table 6. Ablation study on attention layer. Results are the accuracy [%] on the validation dataset of the best models searched in baseline search space (w/o attention layer) and full search space (w/ attention layer).

SPACE	SST	CORA	CITESEER	PUBMED
BASELINE	81.15	81.80	72.18	**81.04**
FULL	**81.68**	**82.96**	**72.90**	**81.04**

5.4. Ablation Study

In this section, we aim to verify the effectiveness of the proposed attention layer in Section 4.1.2 and context-aware parameter sharing in Section 4.2.2. All the results of ablation studies are reported on the **validation** datasets of SST in NLP and Cora, CiteSeer, and PubMed in GRL.

5.4.1. ATTENTION LAYER

To test whether the attention layer and attention representation search are necessary, we carry out experiments that only search in the baseline search space described in Section 4.1.1. To be fair, we add the intra-layer self-attention operation to the *primitive operation pool* as in (Wang et al.,

Table 7. Ablation study on different definitions of contexts. Results are the average accuracy [%] on validation datasets of 100 architectures randomly sampled from space prior $\Gamma(A)$ with parameters from trained supernet.

CONTEXT	SST	CORA	CITESEER	PUBMED
NC	68.68	77.09	63.68	72.72
SC	68.96	77.81	63.62	73.31
TC	69.38	78.50	64.22	77.54
FC	**69.40**	**78.61**	**64.23**	**77.72**

2020b) so that the search space contains hand-crafted self-attention representation designs. The comparison results are shown in Table 6.

We observe a clear performance drop on all validation datasets in both domains except PubMed, which shows that attention representation search is necessary to derive powerful architectures on modeling relation in data in most cases.

For PubMed, we find that the optimal solutions w/ or w/o attention layer point to the same architecture w/o attention layer. The reason may be that the feature of PubMed is not so informative (only 500) compared to large data volumes, which is hard to formulate meaningful representations through a complicated attention layer.

5.4.2. CONTEXT-AWARE PARAMETER SHARING

To verify the effectiveness of context-aware parameter sharing, we test several context variants and compare their performances. Namely, we test four context variants:

- **(NC) No Context** This is the common parameter sharing method used in the previous NAS. The connections at the same place will share their parameters regardless of the layer functionalities connected with them.
- **(SC) Source Context** This kind of context only share parameters within the same kind of source layer.
- **(TC) Target Context** Similar to source context, but it only considers target layer functionalities.
- **(FC) Full Context** This is the context described in Section 4.2.2, which considers functionalities of both source and target layers.

We train the supernet under the same experiment settings using the contexts defined above. Then, we randomly sample 100 architectures from architecture prior $\Gamma(A)$ and report the mean validation score of the optimized parameters. The results are shown in Table 7.

We observe that on all datasets, **FC** and **TC** gives similar validation scores, both gain large comparatively improvements over other two kinds of contexts. **TC** is slightly worse than **FC**. **SC** is slightly better than **NC**. This may be because in our search space, the connections are more related to functionalities of the target layer since it determines the meaning of the chosen connections. The **SC** is a bit better compared to **NC** because it considers the special characteristics of different source layer functionalities when optimizing parameters.

Discussion on context In fact, the context of connections can be generalized beyond the connected layer functionalities. One can further consider other connections connected to the source/target layer of one connection as its context, which is similar to the second rank of neighbors considering the whole architecture as a computation graph. When we consider all ranks of neighbors for one connection, the context becomes the whole architecture, and there will be no shared parameters, which is identical to the architecture search methods without parameter sharing (Zoph & Le, 2017; So et al., 2019).

Therefore, context-aware parameter sharing can be regarded as a trade-off between efficiency and effectiveness, as the goal of the parameter sharing is to reduce the time of training architectures from scratch and is a biased approximation for optimizing parameters of given architectures (Pham et al., 2018).

6. Conclusion and Future Work

In this paper, we propose **AutoAttend** to automate the self-attention representation leveraging NAS. We propose the *attention* layer and define a unified, expressive search space to jointly search for both attention representations and other functional components. Context-aware parameter sharing is proposed to consider the special characteristics of each sub-architecture when training the supernet.

Future work includes automating more complicated attention representations for CV, encoder-decoder or multi-modal models, and automating the attention representation and calculation at the same time. How to search for generalized attention representations mentioned in Section 5.3.1 is also an interesting future work.

Acknowledgments

The authors thank anonymous reviewers and meta-reviewers for their detailed and nice comments and suggestions. The authors also thank Wenpeng Zhang (Tsinghua University) for the constructive discussions on the search space and algorithm design. This research is supported by the National Key Research and Development Program of China (No.2020AAA0106300, No.2020AAA0107800, No.2018AAA0102000) and National Natural Science Foundation of China No.62050110.

References

Bahdanau, D., Cho, K., and Bengio, Y. Neural machine translation by jointly learning to align and translate. In *ICLR*, 2015.

Bender, G., Kindermans, P., Zoph, B., Vasudevan, V., and Le, Q. V. Understanding and simplifying one-shot architecture search. In *ICML*, volume 80 of *Proceedings of Machine Learning Research*, pp. 549–558, 2018.

Bianchi, F. M., Grattarola, D., Livi, L., and Alippi, C. Graph neural networks with convolutional ARMA filters. *CoRR*, abs/1901.01343, 2019.

Brock, A., Lim, T., Ritchie, J. M., and Weston, N. SMASH: one-shot model architecture search through hypernetworks. In *ICLR*, 2018.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners. In *NeurIPS*, 2020.

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., and Zagoruyko, S. End-to-end object detection with transformers. In *ECCV*, volume 12346, pp. 213–229, 2020.

Cho, K., van Merrienboer, B., Bahdanau, D., and Bengio, Y. On the properties of neural machine translation: Encoder-decoder approaches. In *SSST@EMNLP*, pp. 103–111, 2014.

Choi, J., Yoo, K. M., and Lee, S. Learning to compose task-specific tree structures. In *AAAI*, pp. 5094–5101, 2018.

Choi, J., Kim, T., and Lee, S. Cell-aware stacked lstms for modeling sentences. In *ACML*, volume 101 of *Proceedings of Machine Learning Research*, pp. 1172–1187, 2019.

Chu, X., Zhang, B., Xu, R., and Li, J. Fairnas: Rethinking evaluation fairness of weight sharing neural architecture search. *CoRR*, abs/1907.01845, 2019.

Dai, Z., Yang, Z., Yang, Y., Carbonell, J. G., Le, Q. V., and Salakhutdinov, R. Transformer-xl: Attentive language models beyond a fixed-length context. In *ACL*, pp. 2978–2988, 2019.

Daniluk, M., Rocktäschel, T., Welbl, J., and Riedel, S. Frustratingly short attention spans in neural language modeling. In *ICLR*, 2017.

Defferrard, M., Bresson, X., and Vandergheynst, P. Convolutional neural networks on graphs with fast localized spectral filtering. In *NeurIPS*, pp. 3837–3845, 2016.

Devlin, J., Chang, M., Lee, K., and Toutanova, K. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL*, pp. 4171–4186, 2019a.

Devlin, J., Chang, M., Lee, K., and Toutanova, K. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL*, pp. 4171–4186. Association for Computational Linguistics, 2019b.

Fu, Y., Chen, W., Wang, H., Li, H., Lin, Y., and Wang, Z. Autogan-distiller: Searching to compress generative adversarial networks. In *ICML*, volume 119 of *Proceedings of Machine Learning Research*, pp. 3292–3303, 2020.

Gao, Y., Yang, H., Zhang, P., Zhou, C., and Hu, Y. Graph neural architecture search. In *IJCAI*, pp. 1403–1409, 2020.

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 *ECCV*, Lecture Notes in Computer Science, pp. 544–560, 2020.

Hamilton, W. L., Ying, Z., and Leskovec, J. Inductive representation learning on large graphs. In *NeurIPS*, pp. 1024–1034, 2017.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *ICLR*, 2015.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *ICLR*. OpenReview.net, 2017.

Kitaev, N., Kaiser, L., and Levskaya, A. Reformer: The efficient transformer. In *ICLR*, 2020.

Le, H., Tran, T., and Venkatesh, S. Learning to remember more with less memorization. In *ICLR*, 2019.

Li, L. and Talwalkar, A. Random search and reproducibility for neural architecture search. In *UAI*, pp. 367–377, 2019.

Li, Z., Tran, Q., Mai, L., Lin, Z., and Yuille, A. L. Context-aware group captioning via self-attention and contrastive features. In *CVPR*, pp. 3437–3447, 2020.

Lin, Z., Feng, M., dos Santos, C. N., Yu, M., Xiang, B., Zhou, B., and Bengio, Y. A structured self-attentive sentence embedding. In *ICLR*, 2017.

Liu, C., Chen, L., Schroff, F., Adam, H., Hua, W., Yuille, A. L., and Li, F. Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In *CVPR*, pp. 82–92. Computer Vision Foundation / IEEE, 2019a.

Liu, H., Simonyan, K., and Yang, Y. DARTS: differentiable architecture search. In *ICLR*. OpenReview.net, 2019b.

Liu, Y., Meng, F., Chen, Y., Xu, J., and Zhou, J. Depth-adaptive graph recurrent network for text classification. *CoRR*, abs/2003.00166, 2020.

Ma, B., Zhang, J., Xia, Y., and Tao, D. Auto learning attention. In *NeurIPS*, 2020.

Ma, Q., Yu, L., Tian, S., Chen, E., and Ng, W. W. Y. Global-local mutual attention model for text classification. *IEEE ACM Trans. Audio Speech Lang. Process.*, 27(12):2127–2139, 2019.

Mei, J., Li, Y., Lian, X., Jin, X., Yang, L., Yuille, A. L., and Yang, J. Atomnas: Fine-grained end-to-end neural architecture search. In *ICLR*, 2020.

Mino, H., Utiyama, M., Sumita, E., and Tokunaga, T. Key-value attention mechanism for neural machine translation. In *ICNLP*, pp. 290–295, 2017.

Mittal, S., Lamb, A., Goyal, A., Voleti, V., Shanahan, M., Lajoie, G., Mozer, M., and Bengio, Y. Learning to combine top-down and bottom-up signals in recurrent neural networks with attention over modules. In *ICML*, volume 119 of *Proceedings of Machine Learning Research*, pp. 6972–6986, 2020.

Niu, G., Xu, H., He, B., Xiao, X., Wu, H., and Gao, S. Enhancing local feature extraction with global representation for neural text classification. In *EMNLP*, pp. 496–506, 2019.

Pennington, J., Socher, R., and Manning, C. D. Glove: Global vectors for word representation. In *EMNLP*, pp. 1532–1543, 2014.

Pham, H., Guan, M. Y., Zoph, B., Le, Q. V., and Dean, J. Efficient neural architecture search via parameter sharing. In *ICML*, volume 80 of *Proceedings of Machine Learning Research*, pp. 4092–4101, 2018.

Ru, R., Esperança, P. M., and Carlucci, F. M. Neural architecture generator optimization. In *NeurIPS*, 2020.

Sankar, A., Wu, Y., Gou, L., Zhang, W., and Yang, H. Dysat: Deep neural representation learning on dynamic graphs via self-attention networks. In *WSDM*, pp. 519–527, 2020.

Shaw, P., Uszkoreit, J., and Vaswani, A. Self-attention with relative position representations. In *NAACL*, pp. 464–468, 2018.

Shi, H., Pi, R., Xu, H., Li, Z., Kwok, J. T., and Zhang, T. Bridging the gap between sample-based and one-shot neural architecture search with BONAS. In *NeurIPS*, 2020.

So, D. R., Le, Q. V., and Liang, C. The evolved transformer. In *ICML*, volume 97 of *Proceedings of Machine Learning Research*, pp. 5877–5886. PMLR, 2019.

Sun, Y., Wang, Y., Liu, Z., Siegel, J. E., and Sarma, S. E. Pointgrow: Autoregressively learned point cloud generation with self-attention. In *WACV*, pp. 61–70, 2020.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. In *NeurIPS*, pp. 5998–6008, 2017.

Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph attention networks. In *ICLR*. OpenReview.net, 2018.

Wang, B. Disconnected recurrent neural networks for text categorization. In *ACL*, pp. 2311–2320, 2018.

Wang, X., Xiong, X., Neumann, M., Piergiovanni, A. J., Ryoo, M. S., Angelova, A., Kitani, K. M., and Hua, W. Attentionnas: Spatiotemporal attention cell search for video classification. In *ECCV*, volume 12353 of *Lecture Notes in Computer Science*, pp. 449–465, 2020a.

Wang, Y., Yang, Y., Chen, Y., Bai, J., Zhang, C., Su, G., Kou, X., Tong, Y., Yang, M., and Zhou, L. Textnas: A neural architecture search space tailored for text representation. In *AAAI*, pp. 9242–9249, 2020b.

Wu, F., Jr., A. H. S., Zhang, T., Fifty, C., Yu, T., and Weinberger, K. Q. Simplifying graph convolutional networks. In *ICML*, volume 97 of *Proceedings of Machine Learning Research*, pp. 6861–6871. PMLR, 2019.

Yu, Z., Cui, Y., Yu, J., Wang, M., Tao, D., and Tian, Q. Deep multimodal neural architecture search. In *MM*, pp. 3743–3752, 2020.

Zhang, R., Zou, Y., and Ma, J. Hyper-sagnn: a self-attention based graph neural network for hypergraphs. In *ICLR*, 2020.

Zhou, K., Song, Q., Huang, X., and Hu, X. Auto-gnn: Neural architecture search of graph neural networks. *CoRR*, abs/1909.03184, 2019.

Zhou, X., Pappas, N., and Smith, N. A. Multilevel text alignment with cross-document attention. In *EMNLP*, pp. 5012–5025, 2020.

Zoph, B. and Le, Q. V. Neural architecture search with reinforcement learning. In *ICLR*. OpenReview.net, 2017.

Zoph, B., Vasudevan, V., Shlens, J., and Le, Q. V. Learning transferable architectures for scalable image recognition. In *CVPR*, pp. 8697–8710, 2018.