Neural Architecture Search for Natural Language Understanding

Anonymous Authors

Abstract

Neural architecture search (NAS) has made rapid progress in computer vision, whereby new state-of-the-art results have been achieved in a series of tasks with automatically searched neural network (NN) architectures. In contrast, NAS has not made comparable advances in natural language understanding (NLU). Corresponding to encoder-aggregator meta architecture of typical neural networks models for NLU tasks (Gong et al. 2018), we re-define the search space, by splitting it into two parts: encoder search space, and aggregator search space. Encoder search space contains basic operations such as convolutions, RNNs, multi-head attention and its sparse variants, star-transformers. Dynamic routing is included in the aggregator search space, along with max (avg) pooling and self-attention pooling. Our search algorithm is then fulfilled via DARTS, a differentiable neural architecture search framework. We progressively reduce the search space every few epochs, which further reduces the search time and resource costs. Experiments on five benchmark data-sets show that, the new neural networks we generate can achieve performances comparable to the state-of-the-art models that does not involve language model pre-training.

Introduction and Related Work

Neural architecture search (NAS) has recently attracted intensive attention. On one hand, promising methodological innovation for NAS have been developed, e.g. the seminal gradient-based NAS approach DARTS (Liu, Simonyan, and Yang 2018), followed by improvements such as SNAS (Xie et al. 2018), P-DARTS (Chen et al. 2019), PC-DARTS (Xu et al. 2019), etc. On the other hand, NAS has helped to discover better models to for a variety of vision tasks, e.g., image classification (Zoph and Le 2017; Zoph et al. 2017; Cai, Zhu, and Han 2018), semantic segmentation (Liu et al. 2019), object detection (Ghiasi, Lin, and Le 2019), super-resolution (Ahn, Kang, and Sohn 2018), etc.

For natural language processing tasks, NAS is relatively less studied. Except for the general methodology-wise innovations NASNet (Zoph and Le 2016), ENAS (Pham et al. 2018) and DARTS (Liu, Simonyan, and Yang 2018) which pay slight extra effort on searching for new RNN cells on language modeling (LM) tasks, there is little studies tailored to the NLU task. One such an example is the evolved transformer (So, Liang, and Le 2019), which uses the evolution-based NAS algorithm to search for better transformer architecture for machine translation. Although state-of-the-art performance has been achieved on 4 machine translation tasks, the computation cost is exceedingly high since they have to evaluate a large number of models.

In fact, NAS has not been fully investigated for a wide variety of fundamental natural language understanding (NLU) tasks, such as classification (e.g. or sentiment analysis), natural language inference (NLI), sequence tagging tasks such as named entity recognition (NER). Especially, there is no existing work on the effectiveness of one-shot architecture search (Bender et al. 2018) methods on NLU tasks, which could also otherwise significantly reduce the search cost as done in vision tasks.

A typical neural network architecture for NLU includes an encoder which contextualizes the embedded text inputs and extracts higher-level features, and an aggregator that aggregates the encoded inputs to a fix-length vector to make a prediction (Gong et al. 2018). In terms of encoders, many previous NAS literature restrict the search space to non-linear maps such as $tanh$ and $sigmoid$, and the objective to be the discovery of a new recurrent cell to form a new type of recurrent neural network (RNN). However, other than RNNs, there are many other available encoders, for example, convolutional networks (CNN) (Kim 2014), and attention-based model such as transformer (Vaswani et al. 2017), etc. In addition, recent works e.g. star-transformer (Guo et al. 2019) have proposed more sparse versions of transformer to reduce the computational complexity and improve the generalization when there is no pre-trained language model. In addition, as far as we know, there is no existing work on searching for an aggregator. A collection of aggregators are available (Gong et al. 2018). However, one have to choose manually in a trial-and-error fashion.

In this work, we design an encoder search space that contains a rich collection of encoders. The involved operations include: i) the zero map and identity map; ii) the two most commonly used RNNs, LSTM (Hochreiter and Schmidhuber 1997) and GRU (Cho et al. 2014); iii) highway net-

work (Srivastava, Greff, and Schmidhuber 2015); iv) a series of convolutional networks with different kernel sizes; v) multi-head attention from (Vaswani et al. 2017); vi) star-transformer (Guo et al. 2019) and its variants, which will be explained later in the next section. The combination of encoder operations is searched in a encoder search cell, which is a directed acyclic graph (DAG) of intermediate nodes collected by the encoder operations from the encoder search space.

To further reduce the human designs, we propose to search for a suitable aggregator along with the search of encoder cell via an aggregator search cell which includes max (average) pooling, self-attention pooling and dynamic routing (Gong et al. 2018). The aggregator search cell is a DAG with only one step in which the only node is connected to the inputs by a mixture of aggregators.

Our search strategy is mainly based on DARTS (Liu, Simonyan, and Yang 2018). To reduce computation cost, we employ a progressive search space reduction strategy similar to P-DARTS (Chen et al. 2019). Experiments are performed on three different kinds of NLU tasks, i.e., text classification, NLI and NER, with 5 benchmark datasets. For fair comparison, we only compare our results with former state-of-the-art (SOTA) models without large-scale LM pre-training, or any other outside resources like knowledge bases, or any human designed features. Results have shown that with the help of NAS on our search space, we achieve results that are comparable to the SOTA on these 5 tasks, indicating the effectiveness of NAS in the field of NLU research.

Our work contributes the field by the following aspects:

- We re-define the search space for neural architecture search in NLU tasks, by extending and modifying the encoder search space from the evolved transformer, and define the aggregator search space.
- To the best of our knowledge, we are the first to conduct NAS experiments on NLU tasks such as classification, NLI, NER tasks, with one-shot NAS.
- Our approach achieves the results that are comparable to the state-of-the-art models designed by human experts, on various NLU tasks (classification, NLI, NER), by using neural architecture search over the search space defined above. In addition, we demonstrate the effectiveness of one-shot architecture search for NLU tasks.
- We propose a modularized version of star-transformer and its variant, thus including a sparse version of transformer into the search space, which is also novel in the literature. The resulting advantage is that the search cost can be reduced notably and the network's generalization capability can also be improved.

Related Work Recently, a new research field named neural architecture search (NAS) has been drawing more and more attention. The goal is to find automatic mechanisms for generating new neural architectures to replace conventional handcrafted ones. Recently, it is widely applied to computer vision tasks, such as image classification (Zoph and Le 2017; Zoph et al. 2017; Cai, Zhu, and Han 2018), semantic segmentation (Liu et al. 2019), object detection (Ghiasi, Lin, and Le 2019), super-resolution (Ahn, Kang, and Sohn 2018), etc. However, NAS is less well studied in the field of natural language understanding (NLU). Recent works (Zoph and Le 2016; Pham et al. 2018; Liu, Simonyan, and Yang 2018) search new recurrent cells for the language modeling (LM) task on the PennTreebank dataset[1]. The recurrent cell discovered by (Liu, Simonyan, and Yang 2018) achieves the test perplexity of 56.1, which is competitive with the state-of-the-art model enhanced by a mixture of softmaxes (Yang et al. 2017). The evolved transformer (So, Liang, and Le 2019) applies NAS to discover better versions of the transformer architecture. Eploying an evolution-based search algorithm, and the vanilla transformer as the initial population, it generates a better transformer architecture that consistently outperform the vanilla transformer on 4 benchmark machine translation tasks. Our work contributes by going beyond the RNN structure and re-defining the search space to include a richer connection of operations.

Our work is implemented on DARTS (Liu, Simonyan, and Yang 2018) and P-DARTS (Chen et al. 2019). DARTS relaxes the search space to be continuous, so that the architecture can be optimized with respect to its validation set performance by gradient descent. Due to its simplicity, DARTS has inspired a series follow-up work to improve the search stability and efficiency. Based on DARTS, P-DARTS (Chen et al. 2019) divides the search process into multiple stages and progressively increase the network depth at the end of each stage. Our work contributes to the gradient-based NAS (and more generally, one-shot NAS) research by investigating its effectiveness in discovering new NN architectures for a series of NLU tasks.

Our search space design takes advantages of the recent advances in the NLU field. One of the most import advances in sentence encoding is the application of various self-attention mechanisms, among which the transformer (Vaswani et al. 2017) is the most prominent one, which has become ubiquitous in NLU research. Specifically, the QANet (Yu et al. 2018) modifies the transformer architecture to obtain the first place on the SQuaD leaderboard[2]. The transformer is powerful due to its multi-head self-attention mechanism, which can well capture the contextual information. However, the transformer maybe be difficult to train and generalize well on a small or medium sized data-set (Guo et al. 2019). Thus, many other self-attention operations are proposed, e.g., dynamic self-attention (Yoon, Lee, and Lee 2018) and DiSAN (Shen et al. 2018). Recently, (Guo et al. 2019) propose the star-transformer, a sparser version of the multi-head attention model, and achieves competitive results on a series of benchmark datasets like SST-1, SNLI, CoNLL2003. On the aggregation side, an important advancement is the application of capsule networks and dynamic routing policy in text classification (Zhao et al. 2018; Gong et al. 2018). Capsule networks can dynamically decide what and how much information need to be transferred from each word to the final encoding of the text sequence, thus achieving better results even with simple encoders (Gong

[1] https://catalog.ldc.upenn.edu/LDC99T42

[2] https://rajpurkar.github.io/SQuAD-explorer/

et al. 2018). Our work is built upon these work and contributes by: i) include some of the most prominent attention based encoders and aggregators into the search space, and experiment on whether NAS can generate new architectures that have competitive results; ii) we are the first to propose the aggregator search space; iii) we include a modularized version of the star-transformer and its variant into the search space, thus we are the first to combine the dense and sparse multi-head self-attention operations into the same search space.

Search Space Design

We first analyze the meta architectures for a series of NLU tasks, based on which we will define the search space

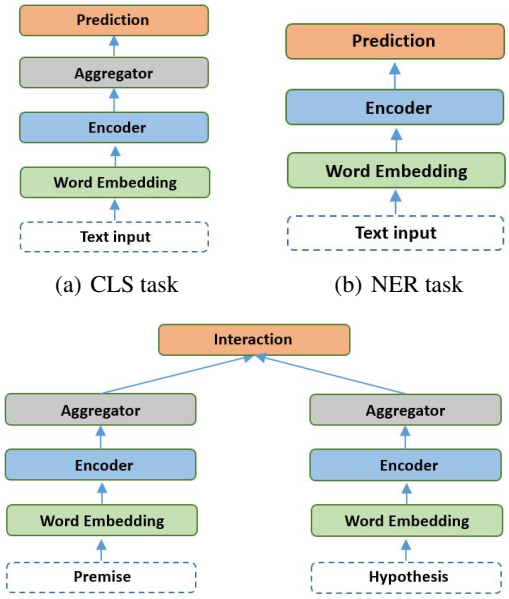
Meta architectures for NLU tasks

As pointed out by (Gong et al. 2018), an NLP model with text inputs and discrete labels (possibly a sequence of labels) can be assembled by the following components: an embedding layer, an encoding layer, an aggregation layer and a prediction layer. Embedding layers usually are static pretrained embeddings like word2vec (Mikolov et al. 2013), or contextualized embedding like ELMO (Peters et al. 2018) and BERT (Devlin et al. 2018). We mainly focus on the encoding layer and aggregation layer.

A text sequence with words $S = w_0, w_1, ..., w_{L-1}$ is mapped into a d-dimensional embedding vector space as $X = x_0, x_1, ..., x_{L-1}$. The encoding layer integrates the information inside the embedding layer and extract higher-level features. The encoded inputs are denoted as $H = h_0, h_1, ..., h_{L-1}$. When the task at hand requires predicting a single label, the final prediction layer requires a fix-length vector. Thus an aggregator is needed to aggregate the information inside sequences of various length to a single fix-length vector, namely h^*.

In this work, we investigate the neural architecture search for three different tasks: classification (CLS), natural language inference (NLI) and named entity recognition (NER). The meta architectures of neural networks are depicted in Figure 1. For classification, the encoder is followed by an aggregator whose output will be passed to the prediction layer. For the NLI task, the encoding and aggregating for the premise and the hypothesis are the same as CLS task, and the aggregated vectors, h_1^*, h_2^*, will interact via an interaction layer before being passed into the prediction layer. Following (Chen et al. 2016), we define the interaction layer as $concat[h_1^*, h_2^*, h_1^* - h_2^*, h_1^* * h_2^*]$. Note that in this work, we will not consider any sort of cross attentions between the two inputs before or after the interaction layer. In addition, due to limited resources for search, we restrict that the encoder and aggregator are shared by both inputs. For the NER task, the aggregator is not required. Note that in this work, we will not consider adding a CRF layer after the enocoder, as done by some other NER models e.g. (Lample et al. 2016). Recall our goal here is to discover and evaluate new model architectures.

Based on the above discussion, we propose to divide the search space into two subspace: encoder search space and aggregator search space.



(a) CLS task (b) NER task

(c) NLI task

Figure 1: Meta architectures for three different NLU tasks.

Encoder search space

In (Liu, Simonyan, and Yang 2018) and (Pham et al. 2018), the objective is to discover new variants of RNNs, so their search space are a collection of linear or non-linear maps, such as, $tanh$ and $sigmoid$. In this work, we will define the encoder space at a more coarse granularity, allowing us to build a richer search space. As (Liu, Simonyan, and Yang 2018), the encoder search space contains the zero map and the identity map.

We then include 1-d convolutional networks (conv1d), and two of RNNs, namely, LSTM and GRU, and the highway network. Highway network is introduced to help train deeper neural network models (see e.g., yu2018qanet). These basic models have been widely used in NLP tasks, such as classification (Kim 2014), Question-answering (Yu et al. 2018), NER (Lample et al. 2016), relation extraction (Zeng et al. 2015), to name just a few. Note that we will use the depth-wise separable convolutions (Chollet 2017) instead of the vanilla convolutions, since the former are more parameter efficient.

Recent years has witnessed the architecture of Transformers (Vaswani et al. 2017) becoming ubiquitous in research community. At its core, the multi-head self-attention mechanism has shown its superior ability to model long-distance contexts. In this work, similar to (So, Liang, and Le 2019), we include the multi-head attention layer, excluding the residual connection that is usually added in a transformer block, into the search space. The point-wise feed-forward layer contains conv1d and residual connection, so we will not include it as a basic operation.

Although transformer has shown its great expressiveness capability, it is difficult to train and easy to over-fit on a small or medium dataset (Guo et al. 2019). One reason is that the

Algorithm 1: Star-transformer module

Input: $h_0^0, h_1^0, ..., h_{L-1}^0$
Output: $h_0^1, h_1^1, ..., h_{L-1}^1$
Initialize: $s^0 \leftarrow average(h_0^0, h_1^0, ..., h_{L-1}^0)$;
for $i = 0$ *to* $L-1$ **do**
  $C_i^1 = [h_{i-1}^0, h_i^0, h_{i+1}^0, s^0]$;
  $h_i^1 = MultiAttn(h_i^0, C_i^1)$;
  $h_i^1 = LayerNorm(ReLU(h_i^1))$;

Algorithm 2: Reversed star-transformer module

Input: $h_0^0, h_1^0, ..., h_{L-1}^0$
Output: $h_0^1, h_1^1, ..., h_{L-1}^1$
Initialize: $s^0 \leftarrow average(h_0^0, h_1^0, ..., h_{L-1}^0)$;
for $i = 0$ *to* $L-1$ **do**
  $s^1 = MultiAttn(s^0, [s^0; H^0])$;
  $s^1 = LayerNorm(ReLU(s^1))$;
  $C_i^1 = [h_{i-1}^0, h_i^0, h_{i+1}^0, s^1]$;
  $h_i^1 = MultiAttn(h_i^0, C_i^1)$;
  $h_i^1 = LayerNorm(ReLU(h_i^1))$;

multi-head attention let each token pay attention to every token in the sentence, resulting in over-parametrization. Thus some more sparse variants are proposed e.g. sparse transformers (Child et al. 2019). We include the star-transformer proposed by (Guo et al. 2019).

The original design of star-transformers requires the relay node being initialized with average pooling and then updated iteratively in a few layers that follows. For better modularization, we modify the Algorithm 1 of (Guo et al. 2019) to Algorithm 3 below. Note that without iterative updating, the relay node is initialized by simple averaging, during which the information of the sentence may not be well preserved. Thus, we propose to first enrich the semantic representation of relay node via multi-head attention between the initialized relay node and the inputs, before we update the satellite nodes. This variant is called the reversed star-transformer, and the algorithm is described in Algorithm 3.

Due to limited resources, for attention based operations, we will only set the attention head number to be 2. Now we formally define the encoder search space, which consists of the following operations:

- Special zero operation, denoted as **null**;
- Skip connection, denoted as **identity**;
- Highway network, denoted as **highway**;
- 1-d separable convolutions, with kernel size k, where $k = 1, 3, 5$, denoted as **sep_conv1d_1**, **sep_conv1d_3** and **sep_conv1d_5**;
- Two RNNs, which are denoted as **lstm** and **gru**;
- Attention-based operations, with attention head number set to be 2, including: multi-head attention (**multi_head_attn_2**), star-transformer (**star_trans_2**) and reversed star-transformer (**reversed_star_trans_2**).

Aggregator search space

There are several different aggregation operations. The most common two are the max pooling and the average pooling. Self-attention technique is also used for aggregation. It assigns each word a weight to indicate the importance of a word depending on the task on hand. A few words that are crucial to the task will be emphasized while the "boring" words are ignored. We also include dynamic routing (Gong et al. 2018) into our aggregator operation space. Unless specified, we use a dynamic routing aggregator with 4 capsules and 3 iteration steps.

Now we formally define the aggregator search space, which includes the following modules:

- Max pooling, denoted as **max-pool**;
- Average pooling, denoted as **avg-pool**;
- Self-attention pooling, denoted as **self-attn-pool**;
- Dynamic routing, denoted as **dynamic-routing**.

Architecture Search

Preliminary on differentiable architecture search

In this work we use Differentiable Architecture Search (DARTS) (Liu, Simonyan, and Yang 2018) as our architecture search framework. The goal is to search for an encoder cell alongside with an aggregator cell. Employing the terminology in (Liu, Simonyan, and Yang 2018), a cell is defined as a directed acyclic graph (DAG) of N nodes, $x_0, x_1, \cdots, x_{N-1}$, where each node is a network layer, i.e., performing a specific mathematical function. We denote the search space as $\Phi_{i,j}$, in which each element represents the information flow connecting node i to node j, which consists of a set of operations weighted by the architecture parameters $\alpha^{(i,j)}$, and is thus formulated as:

$$f_{i,j}(x_i) = \sum_{\phi \in \Phi_{i,j}} \frac{\exp(\alpha_\phi^{(i,j)})}{\sum_{\phi' \in \Phi_{i,j}} \exp(\alpha_{\phi'}^{(i,j)})} \phi(x_i), \quad (1)$$

where $i < j$. An intermediate node can be represented as $f_{i,j}(x_i) = \sum_{i<j} f_{i,j}(x_i)$. Given multiplier m, a hyper-parameter determining the number of nodes whose results are included as the cell's output, the output of the cell is $x_c = concat[x_{N-m}; \cdots; x_{N-1}]$.

The NLU tasks we are dealing with are small or medium sized, so we will not consider stacking multiple encoder cells in this paper.

Progressive search space reduction

Our search space is large, thus we employ a progressive search space reduction strategy during search, which is similar to P-DARTS (Chen et al. 2019). The procedure is as follows: i) start the search with the whole operation space; ii) let k denote the epoch interval length for a reduction. after every k epochs, we drop the operation in $\Phi_{i,j}$ having the lowest score; iii) after each reduction, the search re-starts; iv) repeat step ii) and step iii) till the order of encoder search space drops to 5, and the order of the aggregator search space drops to 1; v) the search procedure continues with the remaining search space till convergence.

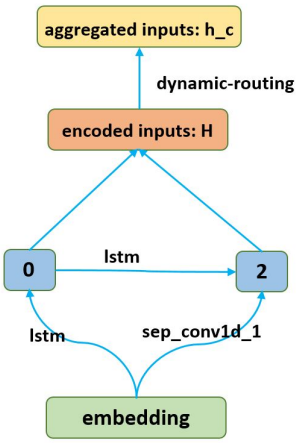


Figure 2: The neural architecture for sentiment classification learned from SST-1.

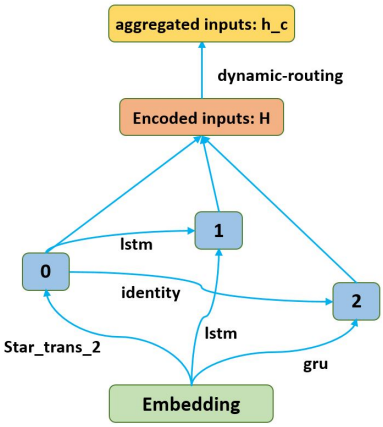


Figure 3: The neural architecture for sentiment classification learned on SST-2.

Deriving discrete architectures

After obtaining the continuous architecture encoding α, deriving the aggregator is simple, since we only need to select the most likely aggregation operation. For the encoder cell, we consider two approaches to derive the final discrete network architecture:

- Following (Liu, Simonyan, and Yang 2018), retain up to k strongest predecessors for each intermediate node, where $k = 1, 2$. Note this approach ignores the appearance of the **null** operation if it obtains the highest score.
- Directly replace every mixed operation as the most likely operation by taking the arg-max. If the best operation connecting two intermediate nodes is a **null** operation, this connection is dropped. In this way, we may find a sparser new encoder cell.

Experiments

In our experiments, for each search or evaluation, we assign 2 CPU cores, 20G memory and 1 Tesla P100 GPU card.

Table 1: An overall of data-sets.

Dataset	Task	Train #	Dev #	Test #	Label #	Metrics
SST-1	Classification	162k	1.1k	2.2k	5	Accuracy
SST-2	Classification	77k	872	1.8k	2	Accuracy
MedNLI	NLI	11.2k	1.4K	1.4k	3	Accuracy
SciTail	NLI	23.5k	1.3K	2.1k	2	Accuracy
CoNLL2003	NER	14k	3.2K	3.4k	4	F1

Datasets

We conduct experiments on three different kinds of tasks with 5 benchmark datasets, whose statistics are shown in Table 1. Specifically, SST-1 and SST-2 (Socher et al. 2013) are two text classification data-sets. Sci-tail and MedNLI are NLI datasets, and CoNLL2003 (Sang and De Meulder 2003) is a benchmark NER datasets[3].

SST-1 Stanford Sentiment Treebank is a movie review dataset which has been parsed and further splitted to train/dev/test set (Socher et al. 2013).

SST-2 This dataset is a binary-class version of SST-1, with neutral reviews removed.

SciTail This is a textual entailment dataset derived from a science question answering (SciQ) dataset (Khot, Sabharwal, and Clark 2018).

MedNLI It is a NLI dataset annotated by doctors, grounded in the medical history of patients (Romanov and Shivade 2018).

CoNLL2003 This dataset consists of 200k training words which have been annotated as Person, Organization, Location, Miscellaneous, or Other (non-named entity).

Our experimental protocols follow (Liu, Simonyan, and Yang 2018). Experiments on each task consist of two stages, architecture search and architecture evaluation. In the search stage, we search for the cells on the train set, and determine the best pair of encoder cell and aggregator cell based on the performance on validation set. In the second stage, we derive discrete architectures from the cells, train them from scratch, and report their performance on the test set.

Architecture search protocols

Initially the search space consists of all the operations in Section **Search Space Design**. For every 3 epochs we carry out search space reduction once, till the search space is halved. The number of intermediate nodes N for the encoder cell ranges from 1 to 3. With N equal to 3, the search cell takes up 9G GPU memory. Note that the number of nodes for the aggregator cell can only be 1. Similar with ENAS (Pham et al. 2018) and DARTS (Liu, Simonyan, and Yang 2018), we enable layer normalization in each node to prevent gradient explosion during architecture search, and disable it during architecture evaluation.

For architecture search, both the embedding and hidden size are set to 300. Word embedding is initialized from pre-trained Glove (Pennington, Socher, and Manning 2014). We randomly initialize word vectors for words that do not appear in Glove. The batch size is 32 for the classification

[3]Because this part of experiment is focused on the NER task, we only test with the NER tags of CoNLL2003's English part.

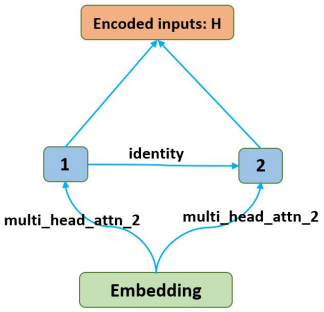


Figure 4: Searched neural architecture for NLI on SciTail. The corresponding aggregator obtained in the same search is average pooling.

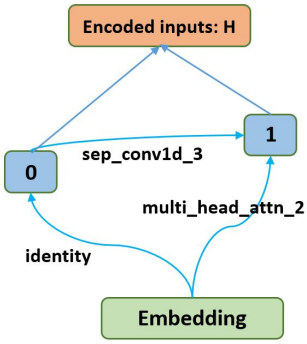


Figure 5: Searched neural architecture for NLI on MedNLI. The corresponding aggregator obtained in the same search is dynamic routing.

tasks and 16 for the others. The learning rate is 1e-4 for both network parameters and architecture parameters, and the weight decay is set to 1e-5, and the dropout rate is set to 0.5. Dropout is applied after embedding look-up, after encoding layer, and to the output layer. The learning rate is warmed up for 1000 steps and then it decreases linearly. The max number of epochs is set 60. We use Adam (Kingma and Ba 2015) to optimize both the network parameters and architecture parameters. The search takes around 1.5 GPU day (Tesla P100) for SST-1, and 0.3 for the SciTail tasks.

Architecture evaluation protocols

We run each search configurations twice with different random seeds and pick the best cell based on the validation performance, and each search run results in at most 3 different dicrete NN architectures. The hyper-parameters are the same with those in the search stage.

Results and discussion

Results on SST Results on SST-1 and SST-2 datasets are listed in Table 2. On the SST-1, DARTS generate a network architecture (DARTS-SST-1-V0) that performs better than most of the traditional NN models. Not that the encoder cell of DARTS-SST-1-V0 contains only RNN and CNN operations, but the exact details of combination of different level of features are impossible to design manually. The best architecture (DARTS-SST-2-V0) we obtained on the SST-2 dataset involves a star-transformer operation and an identity map. Note that since (Guo et al. 2019) did not provide results on SST-2, we use the code from fastNLP[4] to run the transformer and the original star-transformer on SST-2. The results given by us are all the average of 10 different runs. We can see that DARTS-SST-2-V0 can obtain results comparable to the SOTA on SST-2.

We also experiment on the transferability of the learned architectures. From Table 2, we can see that DARTS-SST-2-V0 performs worse than DARTS-SST-1-V0 on SST-1 with a significant margin, but DARTS-SST-1-V0 also performs competitively on SST-2.

Results on NLI tasks Among the architecture candidates derived from the search on SciTail, we find that the one obtained by accepting the null operation when it gets the highest score (DARTS-SciTail-V0) performs best. In addition, this search run gives the average pooling as the aggregator instead of dynamic-routing. The results are presented in

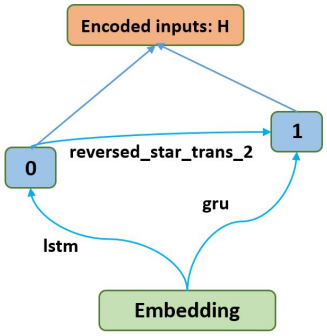


Figure 6: Searched neural architecture for NER on CoNLL2003.

Table 2: Test accuracy (%) on the SST-1 and SST-2 datasets.

Model	SST-1	SST-2
CNN-non-static (Kim 2014)	48.0	87.2
Paragraph-Vec (Le and Mikolov 2014)	48.7	87.8
MT-LSTM (F2S) (Liu et al. 2015)	49.1	87.2
Tree-LSTM (Tai, Socher, and Manning 2015)	51.0	88.0
CNN-Tensor (Lei, Barzilay, and Jaakkola 2015)	51.2	-
BiLSTM + max pooling (Gong et al. 2018)	48.0	87.0
BiLSTM + average pooling (Gong et al. 2018)	46.2	85.2
BiLSTM + self-att (Gong et al. 2018)	48.2	86.4
BiLSTM + dynamic routing (Gong et al. 2018)	50.5	87.6
Emb + self-att (Shen et al. 2018)	48.9	-
DiSAN (Shen et al. 2018)	51.7	-
BiLSTM + self-att (Yoon, Lee, and Lee 2018)	50.4	88.2
CNN + self-att (Yoon, Lee, and Lee 2018)	50.6	88.3
Dynamic self-att (Yoon, Lee, and Lee 2018)	50.6	88.5
Transformer (Guo et al. 2019)	50.4	87.0 *
star-transformer (Guo et al. 2019)	52.9	87.5 *
DARTS-SST-1-V0 (ours)	51.9	88.3
DARTS-SST-2-V0 (ours)	50.8	88.5

* Results obtained by us via running the code at https://github.com/fastnlp/fastNLP/.

[4]https://github.com/fastnlp/fastNLP/

Table 3: Test accuracy (%) on the SciTail dataset.

Model	ACC
600D ESIM (Khot, Sabharwal, and Clark 2018)	70.6
Decomposable Attention (Khot, Sabharwal, and Clark 2018)	72.3
DGEM (Khot, Sabharwal, and Clark 2018)	72.3
AdvEntuRe (Kang et al. 2018)	79.0
HCRN (Tay, Luu, and Hui 2018)	80.0
DeIsTe (Yin, Schütze, and Roth 2018)	82.1
CAFE (Yin, Schütze, and Roth 2018)	83.3
MIMN (Liu et al. 2018)	84.0
ConSeqNet (Wang et al. 2019)	85.2
HBMP (Mihaylov et al. 2018)	86.0
star-transformer (Guo et al. 2019)	79.2*
transformer (Vaswani et al. 2017)	78.6*
DARTS-MedNLI-V0 (ours)	80.3
DARTS-SciTail-V0 (ours)	80.9

* Results obtained by us via running the code at https://github.com/fastnlp/fastNLP/.

Table 4: Test accuracy (%) on the MedNLI dataset.

Model	ACC
ESIM (Romanov and Shivade 2018)	73.1
InferSent (Romanov and Shivade 2018)	73.5
star-transformer (Guo et al. 2019)	75.8*
transformer (Vaswani et al. 2017)	74.3*
DARTS-SciTail-V0 (ours)	74.6
DARTS-MedNLI-V0 (ours)	74.8

* Results obtained by us via running the code at https://github.com/fastnlp/fastNLP/.

Table 3. DARTS-SciTail-V0 achieves a competitive performance on the test set, outperforming the baseline models such as ESIM and decomposable attention by a large margin. It also outperforms the results of the star-transformer and transformer even after extensively parameters tuning. Our model is actually the best one that has no inter-sentence attentions other than the final interaction before the prediction layer, and uses no outside resources, no manually designed features and no extra training mechanism like adversarial training.

As we can see from Figure 5 that, on the MedNLI dataset, the search gives out a architecture (DARTS-MedNLI-V0) that quite resembles the original implementation of the multi-head attention inside the transformer block, except the residual connection is replaced by a sep conv with kernel size 3. DARTS-MedNLI-V0 performs worse than the original star-transformer, but it is better than the original transformer, and the baseline ESIM and InferSent.

We also look into the transferability between the two task. We find that although the datasets are from different domains, the architecture searched on one performs comparable on the other.

Results on CoNLL2003 For the NER task CoNLL2003, since our goal is to compare the previous models and the models discovered by NAS, we do not include the CRF layer which is proven to be a standard component of the best NER models (Lample et al. 2016). We do not use any outside resources like gazetteers, or manually designed features like suffix features and capitalization features, or character embedding. To eliminate implementation discrepancies, we re-run all the results ourselves for this task. Figure 6 gives the searched architecture (DARTS-CoNLL2003-V0) and Table 5 gives out the results. The LSTM sets a strong baseline, and the star-transformer and GRU performs significantly worse. Our DARTS-CoNLL2003-V0 works slightly better than LSTM.

Table 5: Test F1 (%) on the CoNLL2003 dataset.

Model	F1
LSTM (Lample et al. 2016)	84.7*
GRU (Cho et al. 2014)	71.1*
Star-transformer (Guo et al. 2019)	69.4 *
Transformer (Vaswani et al. 2017)	68.9 *
DARTS-CoNLL2003-V0 (ours)	85.3

* Results obtained by us via running the code at https://github.com/fastnlp/fastNLP/.

Conclusion

This paper addresses NAS for a series of NLU tasks. Corresponding to the encoder-aggregator architecture of typical NN models for NLU (Gong et al. 2018), we redefine the search space, by splitting it into encoder search space and aggregator search space. Our search strategy is based on DARTS (Liu, Simonyan, and Yang 2018) and P-DARTS (Chen et al. 2019). Experiments shows that architectures discovered by NAS achieves results that are comparable to the previous SOTA models. In the further, we would like to investigate one-shot architecture search on more large-scale NLU tasks.

References

Ahn, N.; Kang, B.; and Sohn, K.-A. 2018. Fast, accurate, and lightweight super-resolution with cascading residual network. In *ECCV*.

Bender, G.; Kindermans, P.-J.; Zoph, B.; Vasudevan, V.; and Le, Q. 2018. Understanding and simplifying one-shot architecture search. In *ICML*.

Cai, H.; Zhu, L.; and Han, S. 2018. Proxylessnas: Direct neural architecture search on target task and hardware. *arXiv preprint arXiv:1812.00332*.

Chen, Q.; Zhu, X.; Ling, Z.; Wei, S.; Jiang, H.; and Inkpen, D. 2016. Enhanced lstm for natural language inference. *arXiv preprint arXiv:1609.06038*.

Chen, X.; Xie, L.; Wu, J.; and Tian, Q. 2019. Progressive differentiable architecture search: Bridging the depth gap between search and evaluation. In *arXiv preprint arXiv:1904.12760*.

Child, R.; Gray, S.; Radford, A.; and Sutskever, I. 2019. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*.

Cho, K.; Van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Chollet, F. 2017. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1251–1258.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Ghiasi, G.; Lin, T.-Y.; and Le, Q. V. 2019. Nas-fpn: Learning scalable feature pyramid architecture for object detection. In *CVPR*.

Gong, J.; Qiu, X.; Wang, S.; and Huang, X. 2018. Information aggregation via dynamic routing for sequence encoding. *arXiv preprint arXiv:1806.01501*.

Guo, Q.; Qiu, X.; Liu, P.; Shao, Y.; Xue, X.; and Zhang, Z. 2019. Star-transformer. *arXiv preprint arXiv:1902.09113*.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Kang, D.; Khot, T.; Sabharwal, A.; and Hovy, E. 2018. AdvEntuRe: Adversarial Training for Textual Entailment with Knowledge-Guided Examples. *arXiv e-prints* arXiv:1805.04680.

Khot, T.; Sabharwal, A.; and Clark, P. 2018. Scitail: A textual entailment dataset from science question answering. In *AAAI*.

Kim, Y. 2014. Convolutional Neural Networks for Sentence Classification. *arXiv e-prints* arXiv:1408.5882.

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

Lample, G.; Ballesteros, M.; Subramanian, S. e.; Kawakami, K.; and Dyer, C. 2016. Neural Architectures for Named Entity Recognition. *arXiv e-prints* arXiv:1603.01360.

Le, Q. V., and Mikolov, T. 2014. Distributed Representations of Sentences and Documents. *arXiv e-prints* arXiv:1405.4053.

Lei, T.; Barzilay, R.; and Jaakkola, T. 2015. Molding CNNs for text: non-linear, non-consecutive convolutions. *arXiv e-prints* arXiv:1508.04112.

Liu, P.; Qiu, X.; Chen, X.; Wu, S.; and Huang, X. 2015. Multi-timescale long short-term memory neural network for modelling sentences and documents. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Liu, C.; Jiang, S.; Yu, H.; and Yu, D. 2018. Multi-turn inference matching network for natural language inference. In *CCF International Conference on NLPCC*.

Liu, C.; Chen, L.-C.; Schroff, F.; Adam, H.; Hua, W.; Yuille, A. L.; and Fei-Fei, L. 2019. Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In *CVPR*.

Liu, H.; Simonyan, K.; and Yang, Y. 2018. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*.

Mihaylov, T.; Clark, P.; Khot, T.; and Sabharwal, A. 2018. Can a Suit of Armor Conduct Electricity? A New Dataset for Open Book Question Answering. *arXiv e-prints* arXiv:1809.02789.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*.

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

Peters, M. E.; Neumann, M.; Iyyer, M.; Gardner, M.; Clark, C.; Lee, K.; and Zettlemoyer, L. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Pham, H.; Guan, M.; Zoph, B.; Le, Q.; and Dean, J. 2018. Efficient neural architecture search via parameter sharing. *arXiv preprint arXiv:1802.03268*.

Romanov, A., and Shivade, C. 2018. Lessons from natural language inference in the clinical domain. *arXiv preprint arXiv:1808.06752*.

Sang, E. F., and De Meulder, F. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. *arXiv preprint cs/0306050*.

Shen, T.; Zhou, T.; Long, G.; Jiang, J.; Pan, S.; and Zhang, C. 2018. Disan: Directional self-attention network for rnn/cnn-free language understanding. In *AAAI*.

So, D. R.; Liang, C.; and Le, Q. V. 2019. The evolved transformer. *arXiv preprint arXiv:1901.11117*.

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

Srivastava, R. K.; Greff, K.; and Schmidhuber, J. 2015. Highway networks. *arXiv preprint arXiv:1505.00387*.

Tai, K. S.; Socher, R.; and Manning, C. D. 2015. Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks. *arXiv e-prints* arXiv:1503.00075.

Tay, Y.; Luu, A. T.; and Hui, S. C. 2018. Hermitian co-attention networks for text matching in asymmetrical domains. In *IJCAI*.

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

Wang, X.; Kapanipathi, P.; Musa, R.; Yu, M.; Talamadupula, K.; Abdelaziz, I.; Chang, M.; Fokoue, A.; Makni, B.; Mattei, N.; et al. 2019. Improving natural language inference using external knowledge in the science questions domain. In *AAAI*.

Xie, S.; Zheng, H.; Liu, C.; and Lin, L. 2018. Snas: stochastic neural architecture search. *arXiv preprint arXiv:1812.09926*.

Xu, Y.; Xie, L.; Zhang, X.; Chen, X.; Qi, G.-J.; Tian, Q.; and Xiong, H. 2019. PC-DARTS: Partial Channel Connections for Memory-Efficient Differentiable Architecture Search. *arXiv e-prints* arXiv:1907.05737.

Yang, Z.; Dai, Z.; Salakhutdinov, R.; and Cohen, W. W. 2017. Breaking the Softmax Bottleneck: A High-Rank RNN Language Model. *arXiv e-prints* arXiv:1711.03953.

Yin, W.; Schütze, H.; and Roth, D. 2018. End-task oriented textual entailment via deep explorations of inter-sentence interactions. *arXiv preprint arXiv:1804.08813*.

Yoon, D.; Lee, D.; and Lee, S. 2018. Dynamic self-attention: Computing attention over words dynamically for sentence embedding. *arXiv preprint arXiv:1808.07383*.

Yu, A. W.; Dohan, D.; Luong, M.-T.; Zhao, R.; Chen, K.; Norouzi, M.; and Le, Q. V. 2018. Qanet: Combining local convolution with global self-attention for reading comprehension. *arXiv preprint arXiv:1804.09541*.

Zeng, D.; Liu, K.; Chen, Y.; and Zhao, J. 2015. Distant supervision for relation extraction via piecewise convolutional neural networks. In *EMNLP*.

Zhao, W.; Ye, J.; Yang, M.; Lei, Z.; Zhang, S.; and Zhao, Z. 2018. Investigating Capsule Networks with Dynamic Routing for Text Classification. *arXiv e-prints* arXiv:1804.00538.

Zoph, B., and Le, Q. V. 2016. Neural Architecture Search with Reinforcement Learning. *arXiv e-prints* arXiv:1611.01578.

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

Zoph, B.; Vasudevan, V.; Shlens, J.; and Le, Q. 2017. Learning transferable architectures for scalable image recognition. *arXiv preprint arXiv:1707.07012*.