Improving Natural Language Understanding with Computation-Efficient Retrieval Representation Fusion

Shangyu Wu1, Ying Xiong2, Yufei Cui3,†, Xue Liu3, Buzhou Tang3, Tei-Wei Kuo4,5, Chun Jason Xue1
1 City University of Hong Kong 2 Harbin Institute of Technology, Shenzhen 3 MILA, McGill University 4 National Taiwan University 5 Mohamed bin Zayed University of Artificial Intelligence

Abstract

Retrieval-based augmentations that aim to incorporate knowledge from an external database into language models have achieved great success in various knowledge-intensive (KI) tasks, such as question-answering and text generation. However, integrating retrievals in non-knowledge-intensive (NKI) tasks, such as text classification, is still challenging. Existing works focus on concatenating retrievals to inputs as context to form the prompt-based inputs. Unfortunately, such methods require language models to have the capability to handle long texts. Besides, inferring such concatenated data would also consume a significant amount of computational resources. To solve these challenges, we propose ReFusion in this paper, a computation-efficient Retrieval representation Fusion with neural architecture search. The main idea is to directly fuse the retrieval representations into the language models. Specifically, ReFusion first retrieves the representations of similar sentences and uses Neural Architecture Search (NAS) to seek the optimal fusion structures. Experimental results demonstrate our ReFusion can achieve superior and robust performance on various NKI tasks.

1 Introduction

Recent advances in language models (Khandelwal et al., 2020; Borgeaud et al., 2022; Guu et al., 2020; Lewis et al., 2020; Li et al., 2022) have demonstrated that retrieval-based augmentations can achieve remarkable performance on a variety of knowledge-intensive (KI) tasks. The basic idea of retrieval-based augmentations is to first leverage a dense vector indexing to retrieve the top-k related knowledge from an external database, then incorporate the retrieved knowledge into language models. For KI tasks such as question-answering and text generation, they have an inherent retrieval-based property (Chen et al., 2017; Karpukhin et al., 2020) as answers can be sourced or deduced from external knowledge databases.

However, retrieval-based augmentations in non-knowledge-intensive (NKI) tasks, such as text classification, are still challenging. Different from KI tasks, NKI tasks often require understanding and categorizing given sentences rather than generating new sentences (Wang et al., 2019). Previous works (Guo et al., 2023; Izacard & Grave, 2021) treat retrievals as the context of inputs and concatenate retrievals with inputs. However, their methods demand language models to have the capability of handling long sequence data. Figure 1(a) shows that concatenating more retrievals would significantly

* Authors contributed equally to this research
† Corresponding author

increase the length of inputs, but the number of retrievals would be limited by the max sequence length of models. This limitation would result in a performance drop as shown in the red line in Figure 1(b). Besides, processing such long sequence inputs would also consume a substantial amount of computational resources as shown in the green line in Figure 1(b).

In this paper, we introduce ReFusion, a computation-efficient Retrieval representation Fusion framework with neural architecture search. Different from previous retrieval-based augmentations (Izacard & Grave, 2021; Guo et al., 2023), ReFusion directly fuses the representations of retrievals into models. ReFusion consists of three major modules, i.e., the retrieval module for retrieving neighbor representations, the fusion module for fusing the representations, and the search module for seeking the optimal combination of different fusion schemes. Experimental results on 15 NKI tasks show that ReFusion outperforms other comparisons and achieves superior and robust results. Codes are available at [3]

The main contributions of this paper are:

- We are the first to propose fusing the representations of retrievals directly into models to solve the performance and efficiency bottleneck of prompt-based techniques.

- Experimental results demonstrate that our ReFusion framework can significantly improve models’ understanding capability, and achieve a superior and robust performance.

2 ReFusion: A Computation-Efficient Retrieval Representation Fusion with Neural Architecture Search

As shown in Figure 2(b), we propose a computation-efficient retrieval representation fusion framework. Our framework can be adapted to any transformer-based architecture (Vaswani et al., 2017), or any architecture that contains the attention module. The ReFusion contains three modules, i.e., the retrieval module for retrieving the representations of $k$ similar sentences, the fusion module containing different fusion schemes, and the search module for seeking the optimal combination of different fusion schemes. Specifically, the retrieval module encodes the query texts and searches for the representations of top-$k$ similar sentences among billions of data. The fusion module in this work involves different ranking schemes (e.g., a reranker-based scheme and an ordered-mask-based scheme [Rippel et al., 2014], Cui et al. [2023, 2020, 2021], Mao et al. [2022]) to rerank the retrievals for different layers in LMs. Since it is difficult to tell which ranking scheme is better on each layer in LMs, the search module leverages neural architecture search (NAS) techniques to select the optimal ranking scheme or no ranking for each layer.

3 Experiment

3.1 Experimental Setting

Experimental Settings Our experimental setting mainly follows the settings in LM-BFF (Gao et al., 2021). We conduct comprehensive experiments across 15 NKI tasks, including 8 tasks from GLUE benchmark (Wang et al., 2019), SNLI, SST-5, MR, CR, MNLI, MNLI-mm, Subj and TREC. We measure the average performance of five different sampled $D_{train}$ for each task with a fixed set of seed $S_{seed} = \{13, 21, 42, 87, 100\}$, which follows the LM-BFF’s settings. Our models are based on RoBERTa-large for fair comparison with LM-BFF.

To validate the effectiveness of our method, we compared ReFusion with several other models: (1) LM-BFF: a prompt-based fine-tuning approach; (2) DART (Zhang et al., 2022): a differentiable prompt-based model, which can automatically search for the optimal prompt; (3) KPT (Hu et al., 2022): a prompt-based approach incorporating knowledge into the prompt verbalizer; and (4) CA-512: a retrieval-augmented prompt-based method concatenating retrievals with inputs.

3.2 Main Results

Table I presents the main experimental results of our ReFusion and comparisons on 15 NKI tasks. The results are shown in the form of means and variances, with the variance denoted by a subscript.

For tasks with single sentences (S-Task), ReFusion consistently demonstrates superior performance across almost all benchmarks. ReFusion achieves state-of-the-art performance on 5 tasks over 8 tasks. And ReFusion improves the average performance on the S-Task benchmark by about 2.1% than LM-BFF. Specifically, on the TREC task, ReFusion (90.3%) exhibits the maximum improvements over LM-BFF (84.8%).

For tasks consisting of pair sentences (P-Task), ReFusion continues to demonstrate strong performance. ReFusion also achieves the state-of-the-art on 5 tasks over 7 tasks. And ReFusion can improve the average performance on the P-Task benchmark by about 3.0% than LM-BFF. For instance, on the QNLI and SNLI benchmark, ReFusion (73% for QNLI, 80.6% for SNLI) significantly exceeds LM-BFF (64.5% for QNLI, 77.2% for SNLI).

The Avg-all represents the average performance of all 15 NKI tasks. For overall average performance, ReFusion achieves a score of 74.3%, marginally surpassing LM-BFF’s 71.8%. This further highlights...
Table 1: Our main results with RoBERTa-large.

<table>
<thead>
<tr>
<th>Methods</th>
<th>SST-2</th>
<th>SST-5</th>
<th>MR</th>
<th>CR</th>
<th>MPQA</th>
<th>SUBJ</th>
<th>TREC</th>
<th>CoLA</th>
<th>Avg-S</th>
</tr>
</thead>
<tbody>
<tr>
<td>LM-BFF</td>
<td>92.7</td>
<td>47.4</td>
<td>87.0</td>
<td>90.3</td>
<td>84.7</td>
<td>91.2</td>
<td>84.8</td>
<td>9.3</td>
<td>73.4</td>
</tr>
<tr>
<td>DART</td>
<td>93.5</td>
<td>88.2</td>
<td>91.8</td>
<td>88.8</td>
<td>90.3</td>
<td>87.7</td>
<td>74.3</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>KPT</td>
<td>90.3</td>
<td>86.8</td>
<td>88.3</td>
<td>88.3</td>
<td>90.2</td>
<td>88.0</td>
<td>82.2</td>
<td>7.4</td>
<td>70.7</td>
</tr>
<tr>
<td>CA-512</td>
<td>91.3</td>
<td>46.7</td>
<td>88.1</td>
<td>88.3</td>
<td>91.7</td>
<td>86.7</td>
<td>90.3</td>
<td>11.4</td>
<td>75.5</td>
</tr>
<tr>
<td>ReFusion</td>
<td>93.4</td>
<td>49.8</td>
<td>87.9</td>
<td>91.7</td>
<td>87.1</td>
<td>92.5</td>
<td>90.3</td>
<td>14.4</td>
<td>77.5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods</th>
<th>MNLI</th>
<th>MNLI-m</th>
<th>SNLI</th>
<th>QNLI</th>
<th>RTE</th>
<th>MRPC</th>
<th>QQP</th>
<th>Avg-P</th>
<th>Avg-all</th>
</tr>
</thead>
<tbody>
<tr>
<td>LM-BFF</td>
<td>68.3</td>
<td>70.5</td>
<td>65.4</td>
<td>84.5</td>
<td>64.1</td>
<td>74.5</td>
<td>65.5</td>
<td>69.9</td>
<td>71.8</td>
</tr>
<tr>
<td>DART</td>
<td>67.5</td>
<td>75.8</td>
<td>66.7</td>
<td>87.3</td>
<td>67.8</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>KPT</td>
<td>61.4</td>
<td>-</td>
<td>61.5</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CA-512</td>
<td>66.2</td>
<td>67.8</td>
<td>66.9</td>
<td>73.6</td>
<td>64.0</td>
<td>68.8</td>
<td>64.0</td>
<td>68.1</td>
<td>69.5</td>
</tr>
<tr>
<td>ReFusion</td>
<td>69.3</td>
<td>70.9</td>
<td>70.6</td>
<td>73.0</td>
<td>70.9</td>
<td>77.0</td>
<td>72.9</td>
<td>72.9</td>
<td>74.3</td>
</tr>
</tbody>
</table>

The results of LM-BFF, DART refer to their original paper. The results of KPT refer to Chen et al. (2022). The numbers are the average results. The subscript numbers are the standard deviation results.

ReFusion’s consistent and superior performance. Besides, ReFusion surpasses other models like DART, CA-512 and KPT, delivering superior or comparable results. Notably, the standard deviation of ReFusion is considerably smaller than that of other models, indicating that ReFusion produces stable results and offers superior robustness.

3.3 Ablation Study

We conduct ablation experiments on six representative tasks to show the contributions of each module to the overall performance. On all tasks, ReFusion tends to produce better results than those just applying the retrieval fusion module. The results of methods using NAS demonstrate that NAS can significantly boost performance. Specifically, compared to the baseline, two ranking schemes can bring different but significant improvements. This reveals that it is necessary to combine different ranking schemes on different tasks. After using NAS, the performance of each ranking scheme is also significantly improved. This suggests these two ranking schemes are not always suitable for every layer in LMs, thus we need to disable the fusion module at some layers. Finally, our ReFusion integrating all effective candidate fusion modules using NAS achieves the best performance on three tasks. We can infer that the combination of all candidate modules harnesses their strengths.

Table 2: Ablation studies on different modules.

<table>
<thead>
<tr>
<th>Methods</th>
<th>MPQA</th>
<th>SUBJ</th>
<th>TREC</th>
<th>SNLI</th>
<th>QNLI</th>
<th>RTE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Roberta-Large</td>
<td>83.6</td>
<td>90.3</td>
<td>83.8</td>
<td>73.5</td>
<td>65.0</td>
<td>64.1</td>
</tr>
<tr>
<td>Reranker</td>
<td>84.2</td>
<td>91.3</td>
<td>85.0</td>
<td>74.3</td>
<td>68.8</td>
<td>65.3</td>
</tr>
<tr>
<td>Ordered Mask</td>
<td>83.3</td>
<td>90.8</td>
<td>83.0</td>
<td>74.9</td>
<td>68.3</td>
<td>65.8</td>
</tr>
<tr>
<td>NAS with Reranker</td>
<td>86.9</td>
<td>92.4</td>
<td>90.8</td>
<td>83.0</td>
<td>73.5</td>
<td>69.2</td>
</tr>
<tr>
<td>NAS with Ordered Mask</td>
<td>87.0</td>
<td>92.4</td>
<td>90.7</td>
<td>83.0</td>
<td>73.0</td>
<td>70.4</td>
</tr>
<tr>
<td>ReFusion</td>
<td>86.7</td>
<td>92.5</td>
<td>90.3</td>
<td>80.6</td>
<td>73.0</td>
<td>70.9</td>
</tr>
</tbody>
</table>

The numbers are the average results. The subscript numbers are the standard deviation results.

4 Conclusion

In this paper, we aim to solve the bottleneck of prompt-based techniques by directly fusing retrieval representations into models. We propose a computation-efficient retrieval representation fusion framework with neural architecture search, ReFusion. ReFusion uses NAS to fuse retrievals refined by different ranking schemes on each layer in LMs. Experimental results demonstrate our fusion framework outperforms baselines and is robust on various tasks.
References


## A Templates on All Tasks

Table 3 provides an overview of the manual templates and selected label words used for each dataset in our experiments. These templates and label words were created following LM-BFF (Gao et al., 2021).

<table>
<thead>
<tr>
<th>Task</th>
<th>Prompts</th>
<th>Label word</th>
</tr>
</thead>
<tbody>
<tr>
<td>SST-2</td>
<td>[CLS] x It was [MASK]. [SEP]</td>
<td>“0”: “terrible”, “1”: “great”</td>
</tr>
<tr>
<td>MR</td>
<td>[CLS] x It was [MASK]. [SEP]</td>
<td>“0”: “terrible”, “1”: “great”</td>
</tr>
<tr>
<td>CR</td>
<td>[CLS] x It was [MASK]. [SEP]</td>
<td>“0”: “terrible”, “1”: “great”</td>
</tr>
<tr>
<td>MPQA</td>
<td>[CLS] x It was [MASK]. [SEP]</td>
<td>“0”: “terrible”, “1”: “great”</td>
</tr>
<tr>
<td>SUBJ</td>
<td>[CLS] x This is [MASK]. [SEP]</td>
<td>“0”: “subjective”, “1”: “objective”</td>
</tr>
<tr>
<td>CoLA</td>
<td>[CLS] x It was [MASK]. [SEP]</td>
<td>“0”: “incorrect”, “1”: “correct”</td>
</tr>
<tr>
<td>MNLI</td>
<td>[CLS] x1 ? [MASK], x2 [SEP]</td>
<td>“contradiction”: “No”, “entailment”: “Yes”, “neutral”: “Maybe”</td>
</tr>
<tr>
<td>MNLI-m</td>
<td>[CLS] x1 ? [MASK], x2 [SEP]</td>
<td>“contradiction”: “No”, “entailment”: “Yes”, “neutral”: “Maybe”</td>
</tr>
<tr>
<td>SNLI</td>
<td>[CLS] x1 ? [MASK], x2 [SEP]</td>
<td>“contradiction”: “No”, “entailment”: “Yes”, “neutral”: “Maybe”</td>
</tr>
<tr>
<td>QNLI</td>
<td>[CLS] x1 ? [MASK], x2 [SEP]</td>
<td>“not entailment”: “No”, “entailment”: “Yes”</td>
</tr>
<tr>
<td>RTE</td>
<td>[CLS] x1 ? [MASK], x2 [SEP]</td>
<td>“not entailment”: “No”, “entailment”: “Yes”</td>
</tr>
<tr>
<td>MRPC</td>
<td>[CLS] x1 [MASK], x2 [SEP]</td>
<td>“0”: “No”, “1”: “Yes”</td>
</tr>
<tr>
<td>QQP</td>
<td>[CLS] x1 [MASK], x2 [SEP]</td>
<td>“0”: “No”, “1”: “Yes”</td>
</tr>
</tbody>
</table>

## B Results on Full Training Set

We conduct experiments on several tasks under the prompt-based setting with the full training set. As shown in Table 4, across all datasets, ReFusion generally demonstrates either comparable or superior performance compared to LM-BFF. The average performance across all tasks in ReFusion surpasses that of LM-BFF by 1.4%. This suggests that ReFusion’s performance superiority is consistent and not dependent on the size of the dataset. This implies that ReFusion is robust and can generalize well across varying amounts of data.

<table>
<thead>
<tr>
<th>Methods</th>
<th>SST-2</th>
<th>SST-5</th>
<th>MR</th>
<th>CR</th>
<th>MPQA</th>
<th>SUBJ</th>
<th>TREC</th>
<th>CoLA</th>
<th>RTE</th>
</tr>
</thead>
<tbody>
<tr>
<td>LM-BFF</td>
<td>95.0</td>
<td>58.7</td>
<td>90.8</td>
<td>89.4</td>
<td>87.8</td>
<td>97.0</td>
<td>97.4</td>
<td>62.6</td>
<td>80.9</td>
</tr>
<tr>
<td>ReFusion</td>
<td><strong>95.6</strong></td>
<td><strong>61.0</strong></td>
<td><strong>92.3</strong></td>
<td><strong>91.4</strong></td>
<td><strong>84.4</strong></td>
<td><strong>97.1</strong></td>
<td><strong>97.6</strong></td>
<td><strong>62.8</strong></td>
<td><strong>85.2</strong></td>
</tr>
</tbody>
</table>

## C Technique Details

### C.1 The Online Retrieval Module

In the retrieval module, there is a query encoder for encoding query texts and a task-agnostic retriever built offline over billions of dense vectors. The retriever consists of an efficient indexing like FAISS (Johnson et al., 2019) or ScaNN (Guo et al., 2020), and a compressed key-value store database that contains all texts and embeddings. The retrieving process in our framework is online performed, which means that for every forward, the query encoder first passes the representation $h_x$ of the input.
text $x$ to the retriever, then the retriever returns the representations $H_Z = \{h_{z1}, \ldots, h_{zk}\}$ of top-$k$ similar sentences $Z = \{z_1, \ldots, z_k\}$ to the fusion module. For efficient retrieving, especially for the training, the retrieval module maintains an in-memory cache for the input text $x$ and corresponding representations $H_Z$ of similar sentences.

### C.2 The Retrieval Fusion Module

The retrieval fusion module can be wrapped with any modules in the language models (LMs). It takes the representations of top-$k$ similar sentences and the hidden representations of existing modules as inputs, and outputs the fused representations. Specifically, we introduce two effective ranking schemes as shown in Figure 3.

#### C.2.1 Reranking the Retrievals

In the retrieval module, the retrievals are ranked by a task-agnostic similarity metric, e.g., L2 norm. Directly adding the representations to the hidden representations would not improve LMs’ performance. That is because 1) The retrievals are not optimally ranked for the existing module in LMs, which may introduce noise or irrelevant information; 2) The models should pay different attention to those retrievals in case of overemphasizing less relevant information. Therefore, we aim to propose a learnable reranker to learn the ranking distribution tailored to each module in LMs. As shown in the top of Figure 3, the significance of retrievals is re-assigned after reranking.

Specifically, the reranker is a 1D learnable vector of $k$ dimensions, i.e., $R = \{r_1, \ldots, r_k\}$. It is first normalized and then multiplied by the retrievals. Finally, the averaged representation of all reranked retrievals is added to the sentence representation, e.g., [CLS] token in BERT-like models (Liu et al., 2019b; Devlin et al., 2019). The formal steps are as follows,

$$r_i = \frac{\exp(r_i)}{\sum_j \exp(r_j)} \tag{1}$$

$$h_{y[CLS]} = h_{x[CLS]} + \frac{1}{k} \sum_i r_i \cdot h_{z_i} \tag{2}$$

where $h_{x[CLS]}$, $h_{y[CLS]}$ are the sentence representations of inputs and outputs.

#### C.2.2 Ordered Mask Over Retrieval Representations

Rippel et al. ([Rippel et al., 2014] proposed a nested dropout that directly drops the representation units from the sampled index $I$, thus yielding an inherent importance ranking of the representation dimensions. This nested dropout can be implemented by a mask with leading $I$ ones then zeros. Based on the nested dropout, recent works (Cui et al., 2023, 2020, 2021; Mao et al., 2022) proposed the ordered mask that modeled the dropping process with a chain of Bernoulli variables and made it differentiable using the re-parameterization trick.

As shown in the bottom of Figure 3, we apply the ordered mask over $k$ retrievals on each representation dimension. This means that different from the reranker, the ordered mask trusts the ranking produced by the retriever.
by the retriever and refines the ranking with training data. Specifically, let \( h_{z_1}, \ldots, h_{z_k} \) be the top-\( k \) \( D \)-dimensional retrieval representations. For each dimension of retrieval representation (e.g., the dimension \( d \)), the ordered mask is modeled by a chain of Bernoulli variables \( V = \{ v_i^d, \ldots, v_k^d \} \), where \( v_i^d \sim \text{Bernoulli}(\pi_i) \) indicates whether drop the \( d \)-th representation unit of the \( i \)-th retrieval. Following the property of nested dropout, the variable \( v_i^d \) is conditioned on \( v_{i-1}^d \), thus we can obtain the marginal distribution \( p(v_i^d) \) of \( v_i^d \).

After that, the ordered mask uses the re-parameterization trick, e.g., choosing the Gumbel Softmax distribution \( \text{Gumbel}(\beta, \tau) \) as the tractable variational distribution \( q(v_i^d) \). With Gumbel Softmax distribution, if \( c^d \sim \text{Gumbel}(\beta, \tau) \), then \( v_i^d = 1 - \text{cumsum}(c^d) \), where \( c^d \) is a sample choice of the dropped index over \( k \) retrievals on the dimension \( d \), and \( \text{cumsum}_i(c^d) = \sum_{j=0}^{i-1} c_j^d \). In the Gumbel Softmax distribution, \( \beta \) is a learnable parameter in the differentiable function \( v_i^d = g(\epsilon_i; \beta) \) and \( \tau \) is the temperature variable that controls the smoothness of the step at the dropped index.

Finally, we obtain the different ordered mask \( V^1, \ldots, V^D \) over representation dimensions. We use it to mask the retrievals in a fine-grained way. Then, the masked retrievals would be fused into the sentence representations in the same way as Reranker. The formal steps are as follows,

\[
\begin{align*}
  c^d &\sim \text{Gumbel}(\beta, \tau) \\
  v_i^d &= 1 - \text{cumsum}(c^d) \\
  \hat{h}_z^d &= v_i^d \cdot \hat{h}_{z_i}^d \\
  h_{y(\alpha \beta)} &= h_{x(\alpha \beta)} + \frac{1}{k} \sum_i \hat{h}_{z_i} 
\end{align*}
\]

where \( \hat{h}_{z_i}^d \) is the \( d \)-th masked representation unit of \( i \)-th retrieval.

### C.3 The Architecture Search Module

As shown in Figure 3, it is difficult to tell which ranking scheme is better on each layer in LMs. Therefore, we propose an architecture search module, aiming to leverage neural architecture search (NAS) techniques to search to select the optimal ranking scheme.

#### C.3.1 Search Space

In this work, we do not search for a totally new neural network architecture like previous NAS works (Liu et al., 2019a) do. Instead, we keep the main structure of transformer-based models unchanged and only replace several modules with our search modules.

A search module consists of multiple fusion modules with different ranking schemes and the original module. For example, taking the linear module in LMs as an example, we replace the linear module with our linear search module, which includes three modules, the fusion module with reranker-based scheme, the fusion module with ordered-mask-based scheme, and the original linear module.

Although the number of candidate modules in the search module is small, the whole search space is quite large. Given a transformer-based language model with \( N \) hidden layers, assume that we only replace the linear module for the key and value in every attention module, we have at least \( 3 \times 3 = 9 \) candidate modules and thus at least \( 9^N \) different retrieval-augmented transformer-based language models. Taking the RoBERTa-large as an example, which has 24 layers, the search space can be septillion-level large.

#### C.3.2 Searching Details

We follow the same searching strategies used in DARTS (Liu et al., 2019a). Specifically, let \( \alpha = \{ \alpha_1, \ldots, \alpha_l \} \) be the architectural weights, where \( l \) is the number of candidate modules in each search module. To make the search space continuous, we also relax the categorical choice of a particular candidate module to a softmax over all possible candidate modules within the search module,

\[
\hat{o}(h) = \frac{\sum_i \exp(\alpha_i) o_i(h)}{\sum_j \exp(\alpha_j)}
\]
where $o_i(h)$ represents the output of the $i$-th candidate module $o_i(\cdot)$ taking the hidden states $h$ as input, $\hat{o}(\cdot)$ indicates the output of the search module.

The goal of architecture searching is to jointly optimize the architectural weights $\alpha$ and the weights $\omega$ of all modules with LMs. We update the weights $\omega$ based on the training loss, and the architectural weights based on the validation loss. The updates of these two types of weights are done alternatively. After training, we only choose the candidate module with the largest architectural weights for the inference.