LYU2002

Study Neural Architecture Search

[Neural Architecture Search on BERT for Network Compression]

ESTR4998 2020/21 Term 1 Oral Presentation

Yau Chung Yiu, Oscar

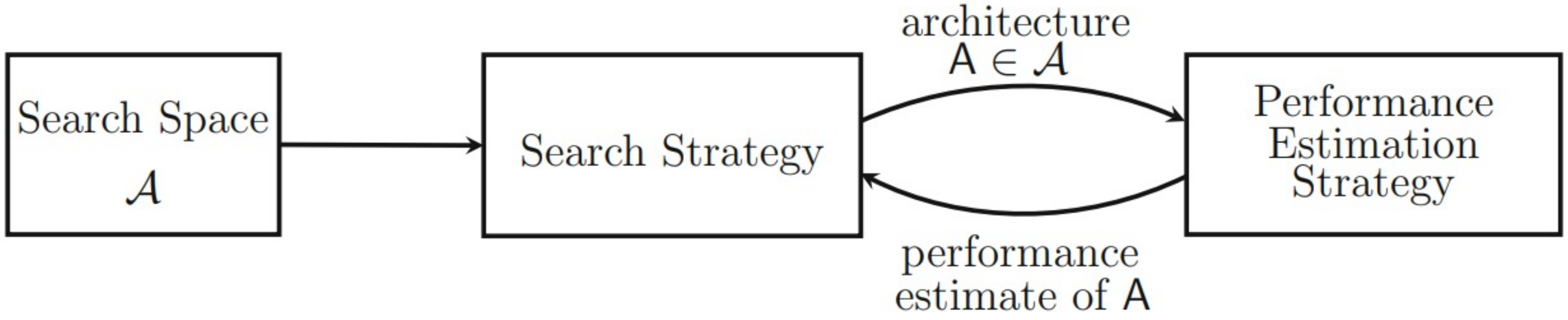
1155109029

30 min present, 15 min Q&A

7-12-2020

Introduction to Neural Architecture Search (NAS)

- What defines an NAS algorithm?

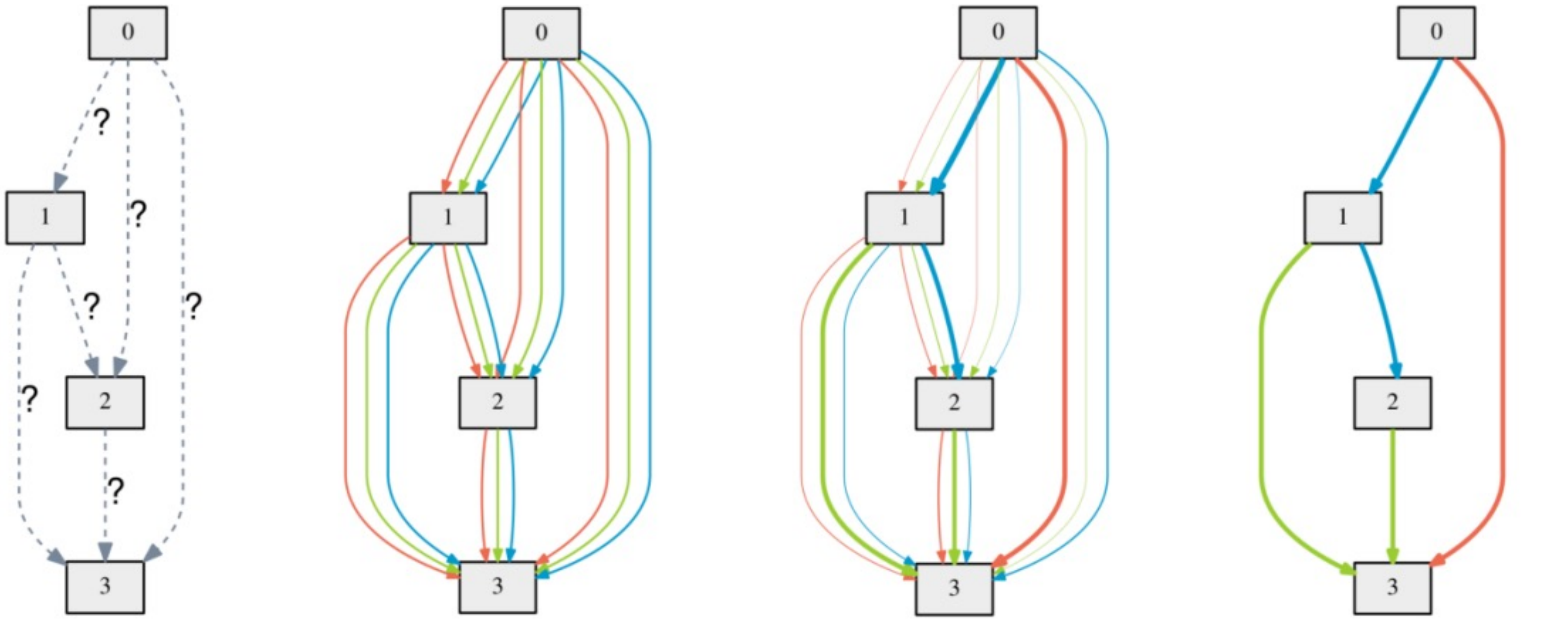


What operations are allowed
e.g. in AdaBERT [11]
- Convolution
- Pooling
- Skip connection
- Zero operation

How to pick the next architecture
e.g.
- Reinforcement Learning
- Genetic Algorithm
- Bayesian Optimization
- Gradient-based

How to evaluate the performance of the architecture
e.g.
- Full training evaluation
- Bayesian Optimization estimation

Example - DARTS: Differentiable Architecture Search



(a) (b) (c) (d)

Train all connection at the same time (Supergraph)

Learn the alpha weight by backpropagation (Learn sub architecture)

Take the maximum alpha as result (Final subgraph)

Example - DARTS

Algorithm 1: DARTS – Differentiable Architecture Search

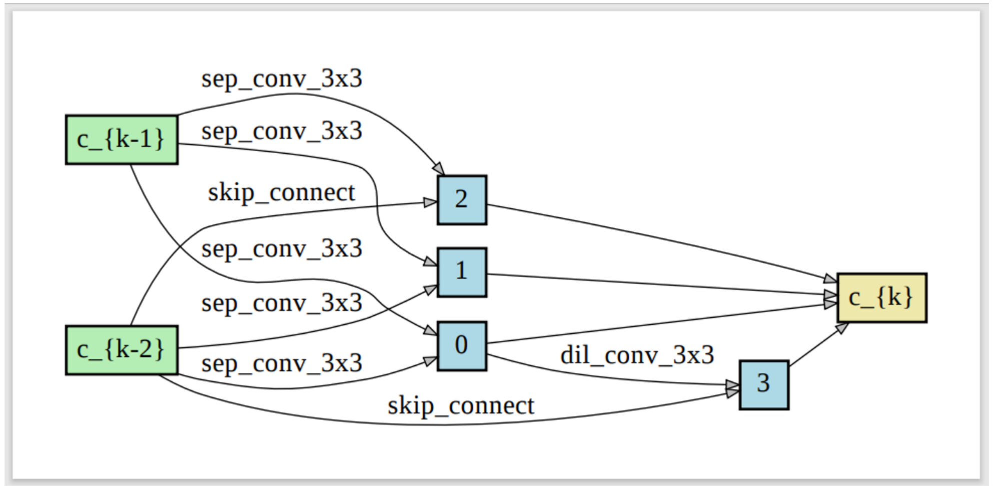
Create a mixed operation $\bar{o}^{(i,j)}$ parametrized by $\alpha^{(i,j)}$ for each edge (i, j)
while *not converged* **do**
  1. Update architecture α by descending $\nabla_{\alpha}\mathcal{L}_{val}(w - \xi\nabla_{w}\mathcal{L}_{train}(w, \alpha), \alpha)$
     ($\xi = 0$ if using first-order approximation)
  2. Update weights w by descending $\nabla_{w}\mathcal{L}_{train}(w, \alpha)$

Derive the final architecture based on the learned α.

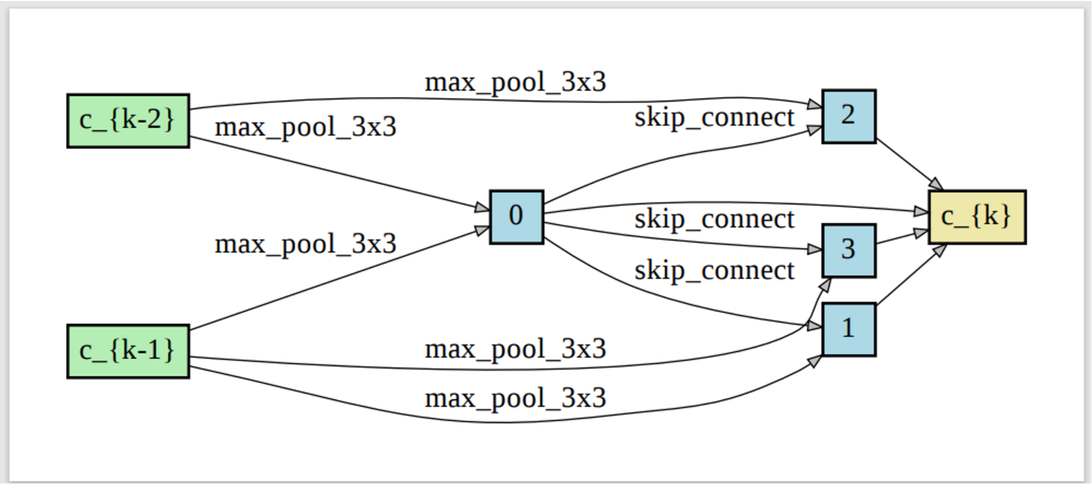
Step 1. Learn the architecture parameter α
Step 2. Learn the supergraph model parameter (weights) w

Example - DARTS

Normal cell

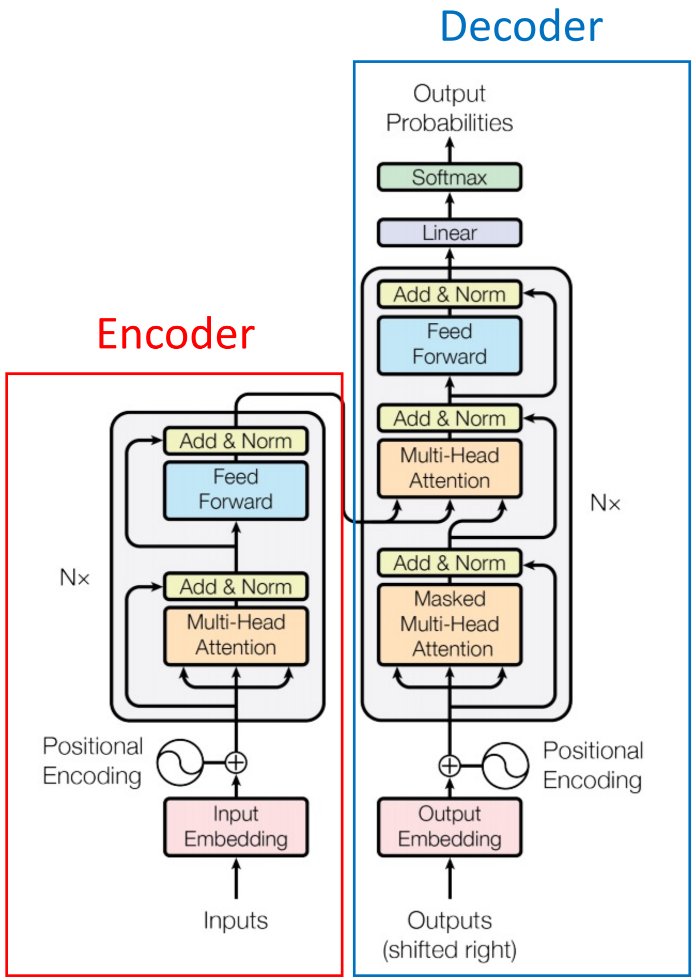


Reduction cell



Each cell receive two input from previous two cell.
Reduction cell is put at 1/3 and 2/3 of the total depth of the network.

Transformer [17]



Transformer, attention mechanism for sequence transduction

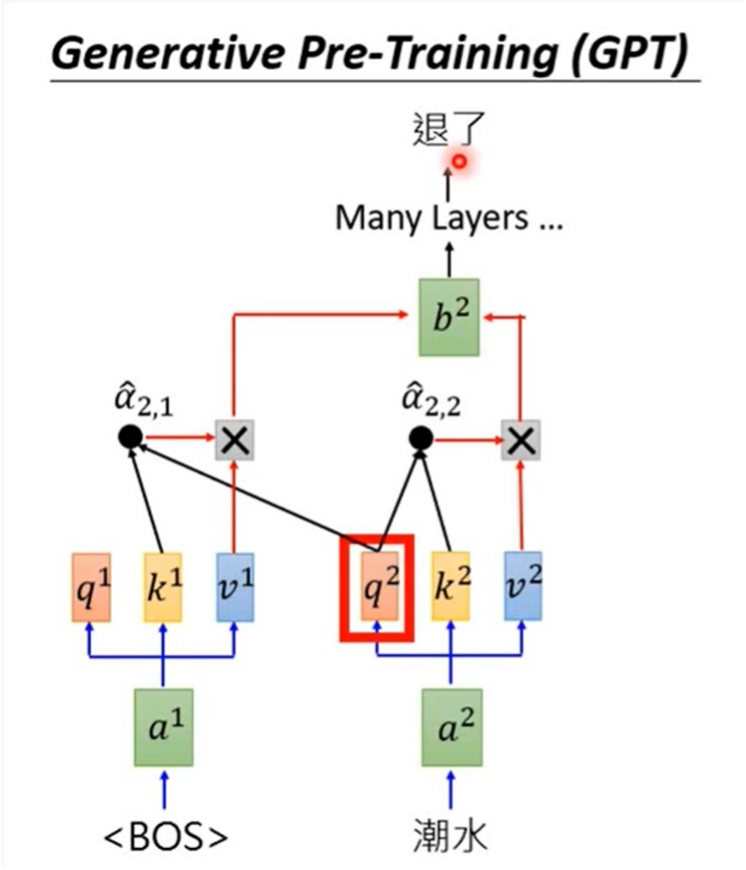
- Sequence-to-sequence model
- $O(1)$ path length between long-range dependencies (across the words, within a sentence)
- Parallelizable, unidirectional (compared to RNN)
  - *O(1) vs O(n)* sequential operations
- Application: Translation (English-German, English-French)[17]

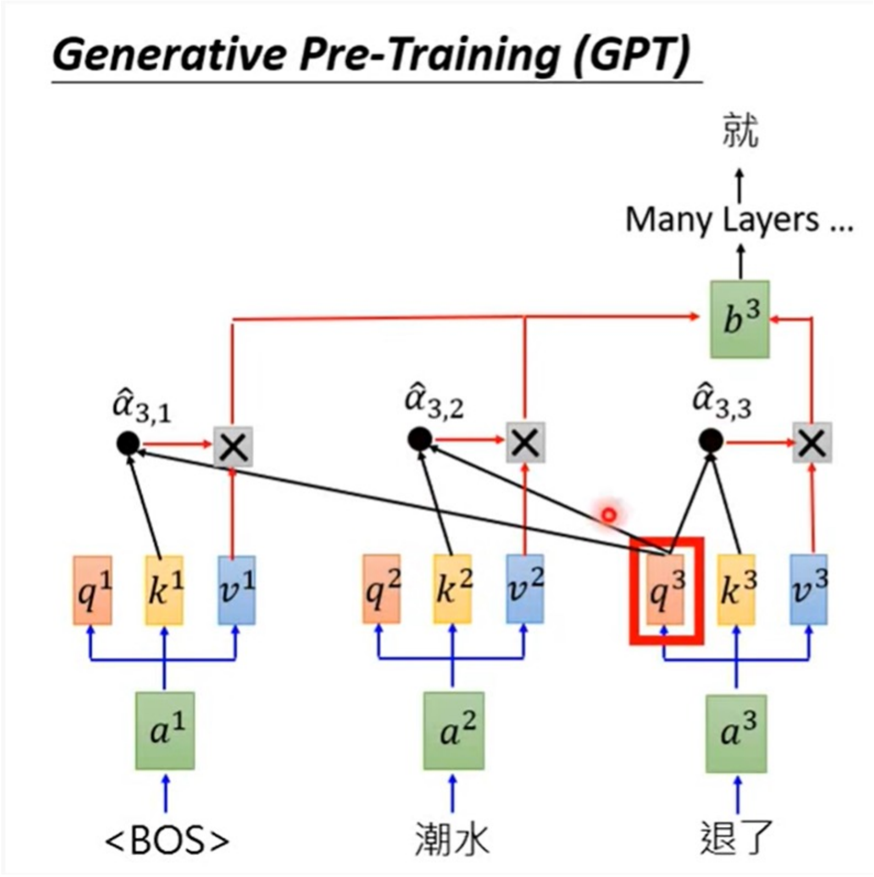
Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(log_k(n))$
Self-Attention (restricted)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

Attention explained

GPT: left-to-right generative
BERT: unidirectional, predictive

$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$





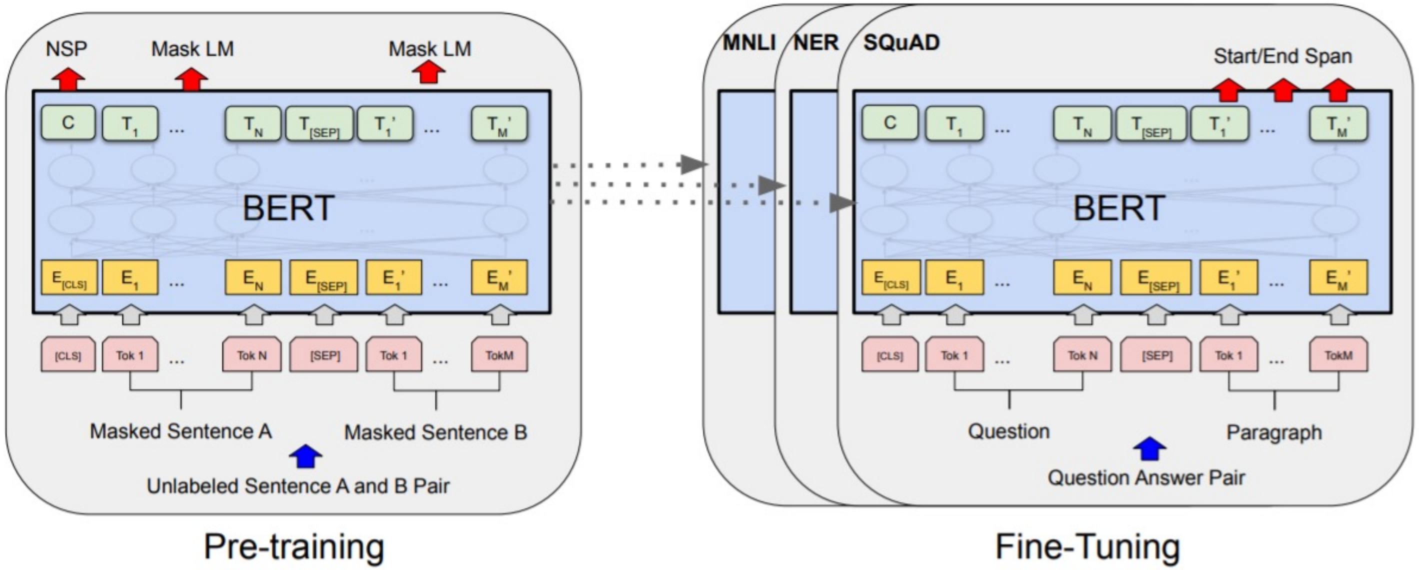
https://www.youtube.com/watch?v=UYPa347-DdE&ab_channel=Hung-yiLee

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding [21]

- Essentially is the encoder of Transformer
- Large model: $\text{BERT}_{\text{BASE}}$: 110M, $\text{BERT}_{\text{LARGE}}$: 340M parameters
- Pre-training: To understand/learn the language (self-supervised learning)
- Fine-Tuning: To learn achieving specific task

Pre-training task (on large corpus, 800M+2500M words):

1. Masked LM: Predict masked words by softmax
2. Next Sentence Prediction: Predict sentence relation



Downstream task:

1. Classification
2. Question answering

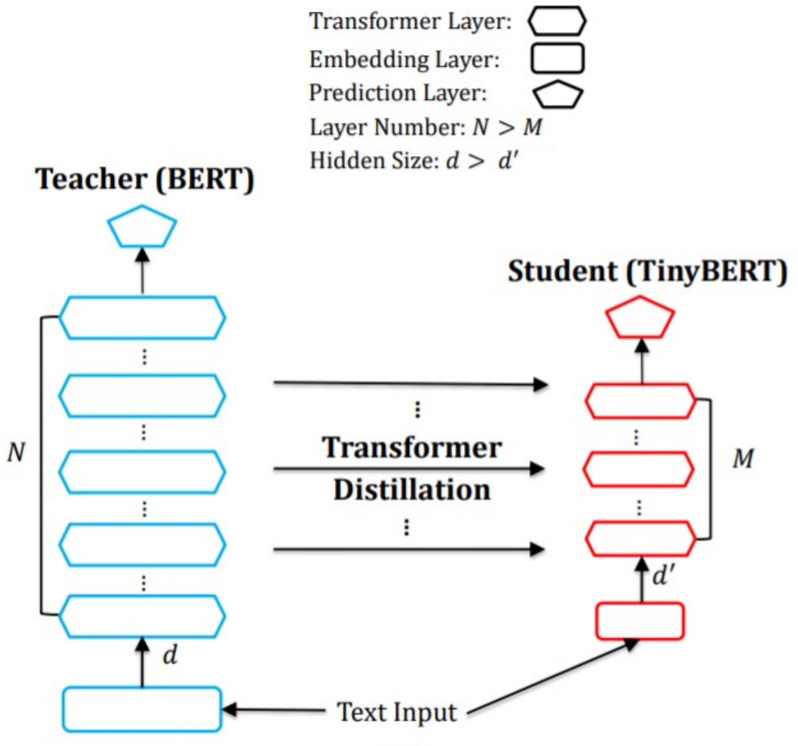
....

(GLUE dataset), *classification*

TinyBERT: Distilling BERT for Natural Language Understanding [6]

- Knowledge distillation on BERT
  - General distillation (pre-train the student) (feed large corpus)
  - Task-specific distillation (fine-tune the student) (feed task-specific dataset)

$N = 12$
bert-base-uncased
(pretrained model)



(a)

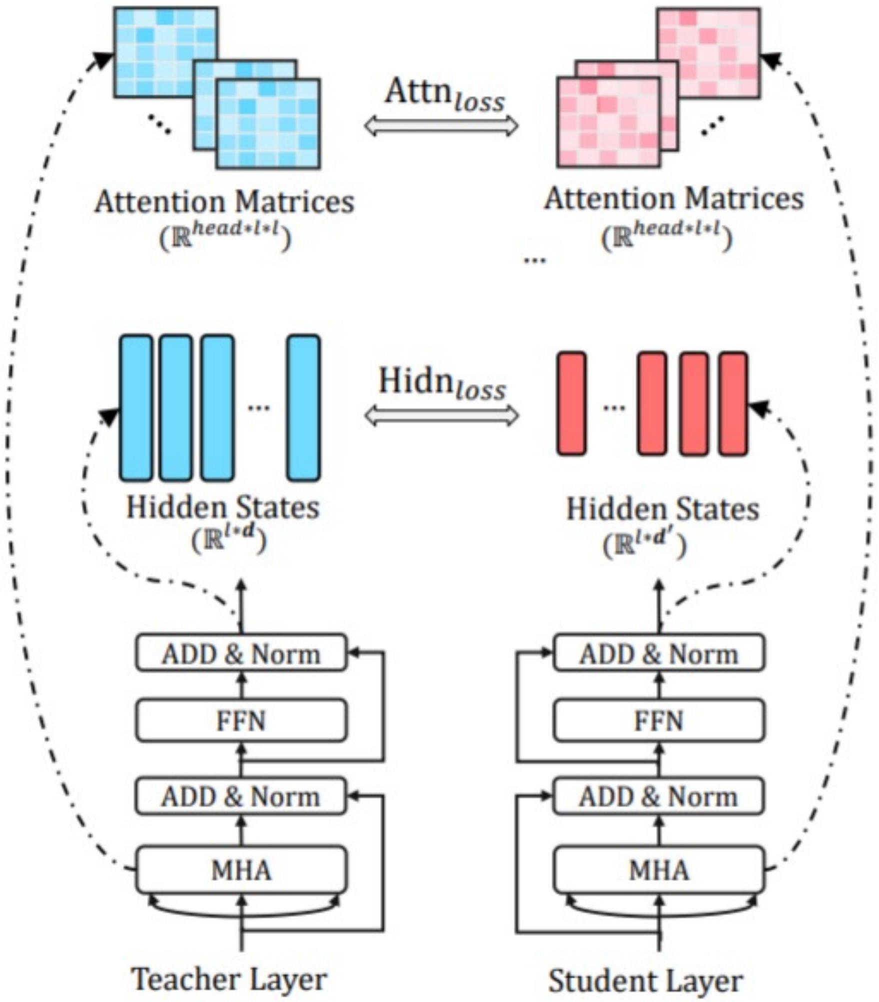
$M = 4$

TinyBERT

- Distillation objectives

(ranked by importance towards the final performance)

1. Attention matrices (Transformer-layer)
2. Hidden states (Embedding-layer)
3. Softmax outputs (Prediction-layer)



(b)

General Language Understanding Evaluation benchmark [23] (GLUE)

CoLA

- Acceptability judgement of grammatical correctness
- Correct or incorrect

Example

Correct:
They made him angry.

Incorrect:
They caused him to become angry by making him.

Small

SST-2

- Sentiment classification, from movie reviews
- Positive or negative

Example

Positive:
that loves its characters and communicates something rather beautiful about human nature

Negative:
contains no wit , only labored gags

Medium

RTE

- Textual entailment classification of a pair of sentence
- Entailment or not entailment

Example

Not entailment:
No Weapons of Mass Destruction Found in Iraq Yet.

Weapons of Mass Destruction Found in Iraq.

Entailment:
Valero Energy Corp., on Monday, said it found "extensive" additional damage at its 250,000-barrel-per-day Port Arthur refinery.

Valero Energy Corp. produces 250,000 barrels per day.

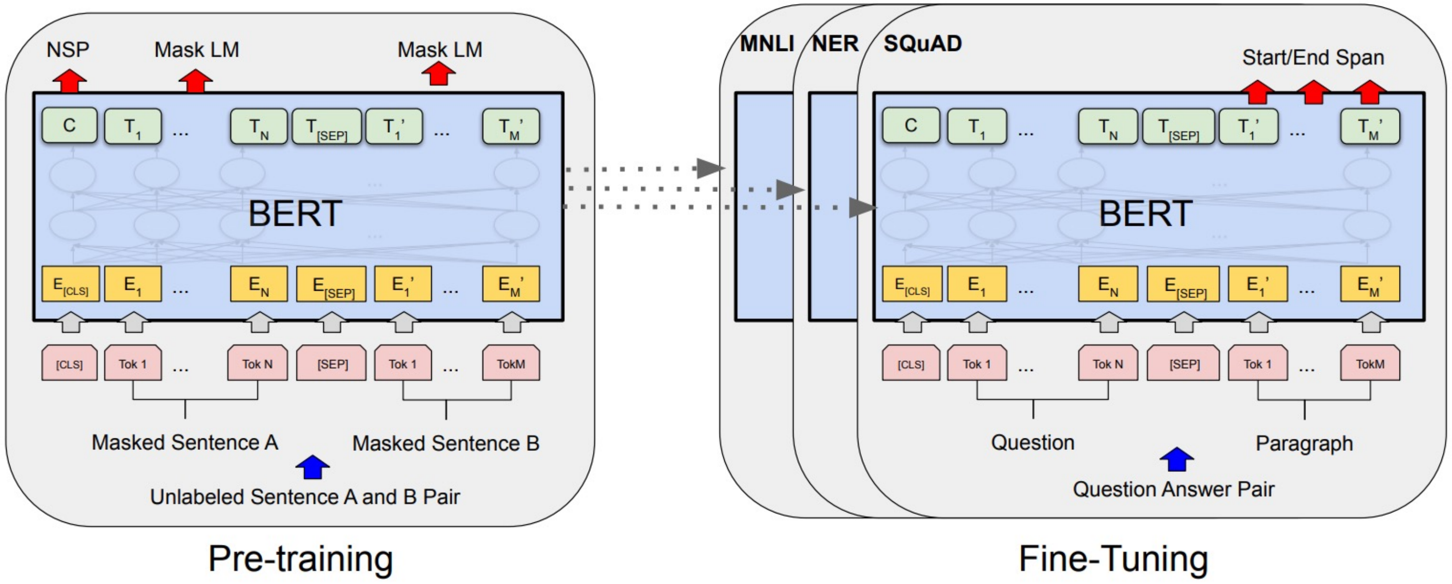
Large

Motivation – Expected Redundancy

- Why BERT?

Large Text Corpus
- Learn language feature

Downstream Task Dataset
- Learn skill to accomplish tasks



<u>The resulting model for downstream task requires less language knowledge.</u>
<u>Redundancy is remained in the model after fine-tuning.</u>

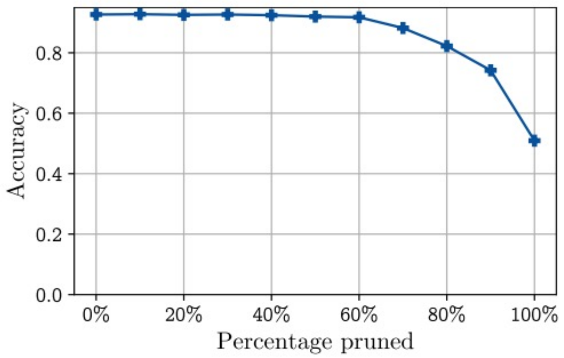
Motivation – Redundancy in Multi-head

- [1905.10650] Are Sixteen Heads Really Better than One? (arxiv.org)

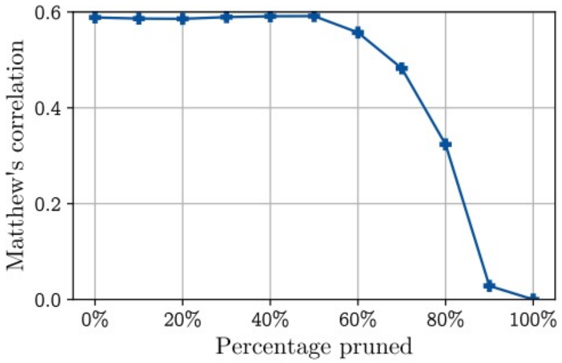
Layer		Layer	
1	-0.01%	7	0.05%
2	0.10%	8	-0.72%
3	-0.14%	9	-0.96%
4	-0.53%	10	0.07%
5	-0.29%	11	-0.19%
6	-0.52%	12	-0.12%

Table 3: Best delta accuracy by layer when only one head is kept in the BERT model. None of these results are statistically significant with $p < 0.01$.

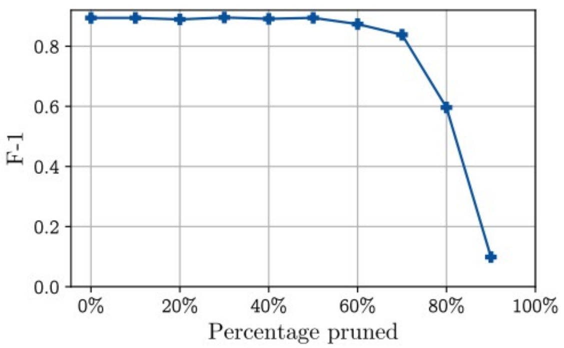
Keep one significant head at each layer



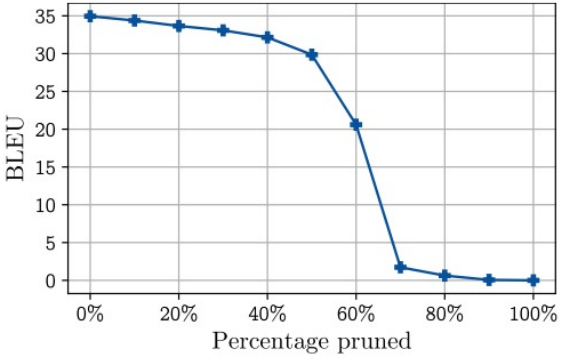
(a) Evolution of accuracy on the validation set of **SST-2** when heads are pruned from BERT according to I_h.



(b) Evolution of Matthew's correlation on the validation set of **CoLA** when heads are pruned from BERT according to I_h.



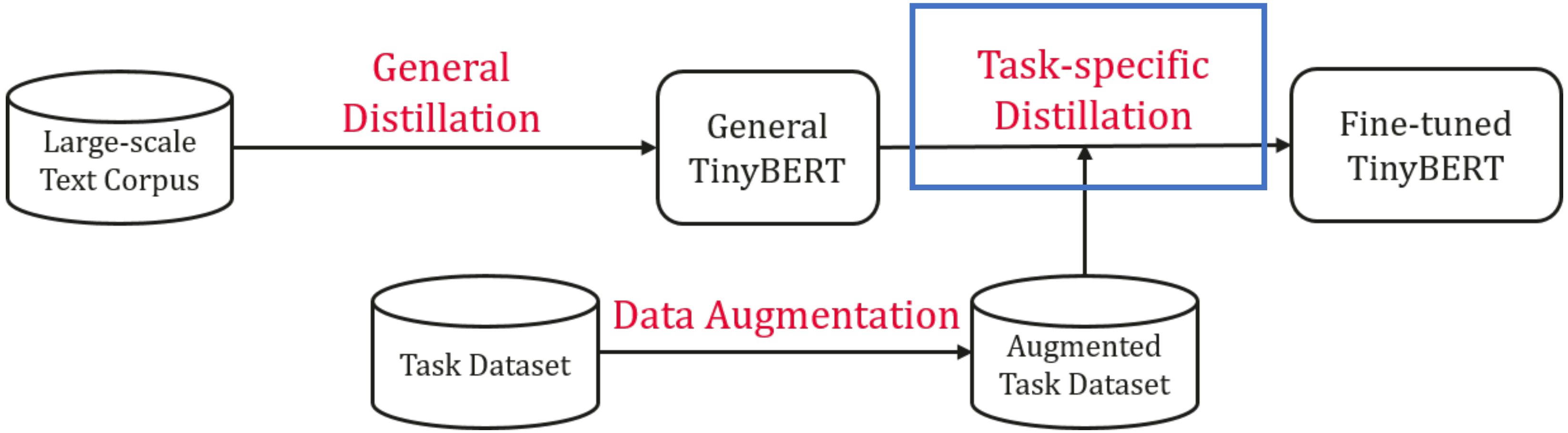
(c) Evolution of F-1 score on the validation set of **MRPC** when heads are pruned from BERT according to I_h.



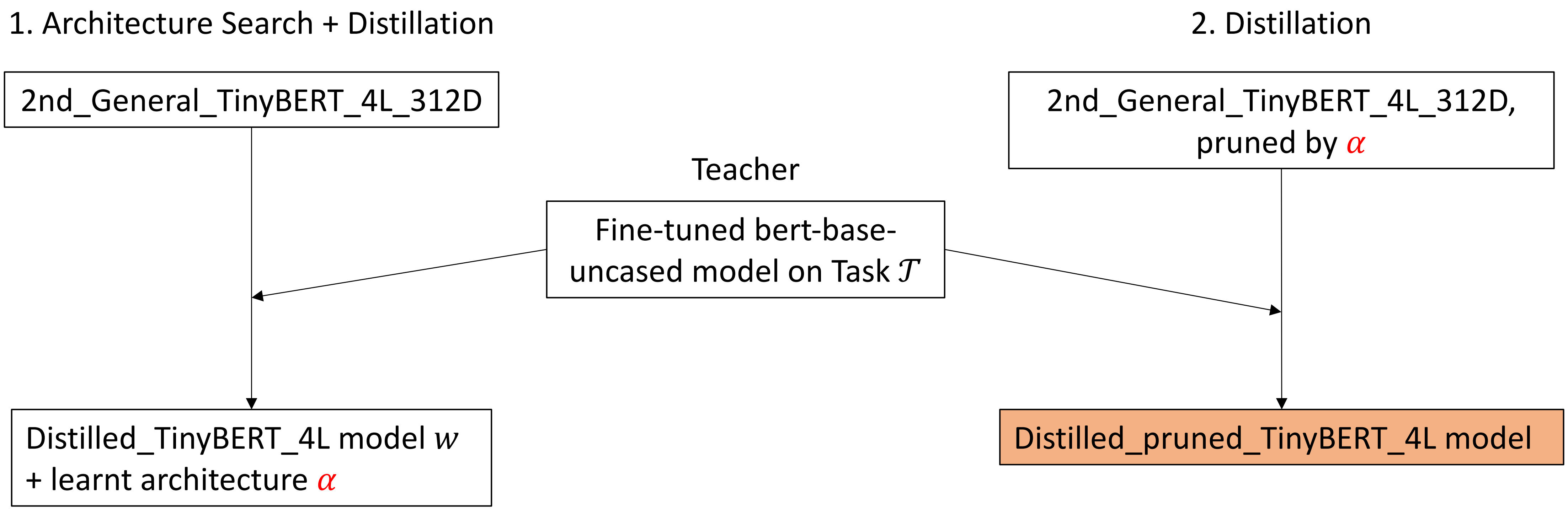
(d) Evolution of the BLEU score of our **IWSLT** model when heads are pruned according to I_h (solid blue).

Experiment Setup

- Inherit the results of TinyBERT
- Focus on Task-specific Distillation



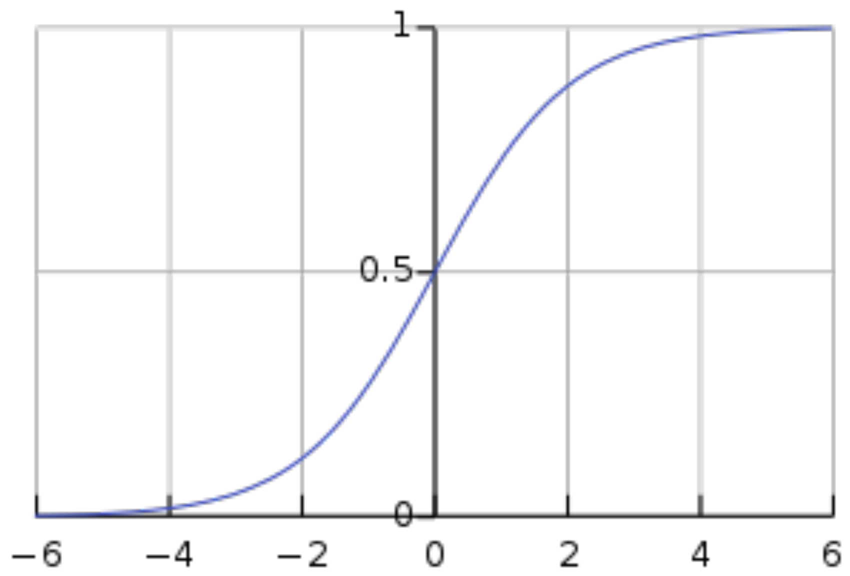
Experiment Procedure

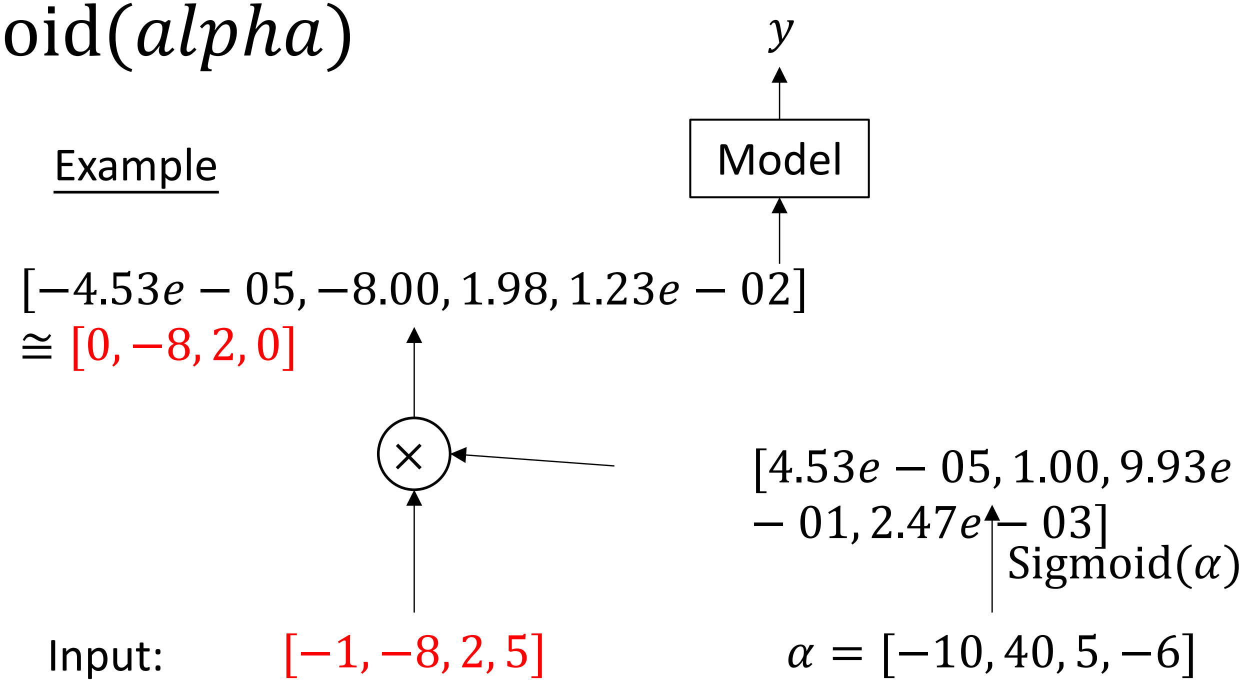


5.2 Experiment Procedure in the report

Search Method for Hidden Representation Dimensions

- $alpha = [\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_n]$, n is the hidden representation size
- $\text{forward\_mask} = \text{torch.sigmoid}(alpha)$





$$\text{grad} = \frac{\text{grad}}{10^{\log_{10}|\text{grad}|}}$$

Scale down the gradient to (-10, 10)

5.5.1 Search Method for Representation Dimension in the report

```
.register_hook(lambda grad: grad / 10**torch.log10(torch.abs(grad)+1e-9))
```

Search Method for Hidden Representation Dimensions (Animation)

Example

iter 1

Input: $[-1, -8, 2, 5]$

$\alpha = [\ 5\ ,\ 5\ , 5,\ 5\ ]$

Search Objective

Adopt similar approach like TAS[3]

$$\mathcal{L}_{arch} = \underbrace{-\log\left(\frac{\exp(z_y)}{\sum_{j=1}^{|z|}\exp(z_j)}\right)}_{\substack{\textit{cross-entropy} \\ \textit{classification loss}}} + \underbrace{\lambda_{cost}\mathcal{L}_{cost}}_{\substack{\textit{computational} \\ \textit{cost loss}}}$$

$$\mathcal{L}_{cost} = \begin{cases} \log(F_{cost}(\mathbb{A})) & \text{when } F_{cost}(\mathbb{A}) > (1+t)\times R \\ 0 & \text{when } (1-t)\times R < F_{cost}(\mathbb{A}) < (1+t)\times R \\ -\log(F_{cost}(\mathbb{A})) & \text{when } F_{cost}(\mathbb{A}) < (1+t)\times R \end{cases}$$

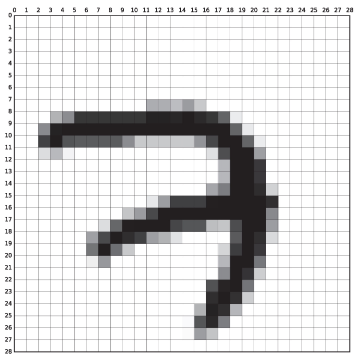
t – target ratio
R – tolerance
$F_{cost}(\mathbb{A})$ – computational cost metric, e.g., FLOPS

- Cross-entropy classification loss encourages the model to learn the useful architecture
- Computation cost loss encourages the model to minimize the model size

5.3 Search Objective in the report

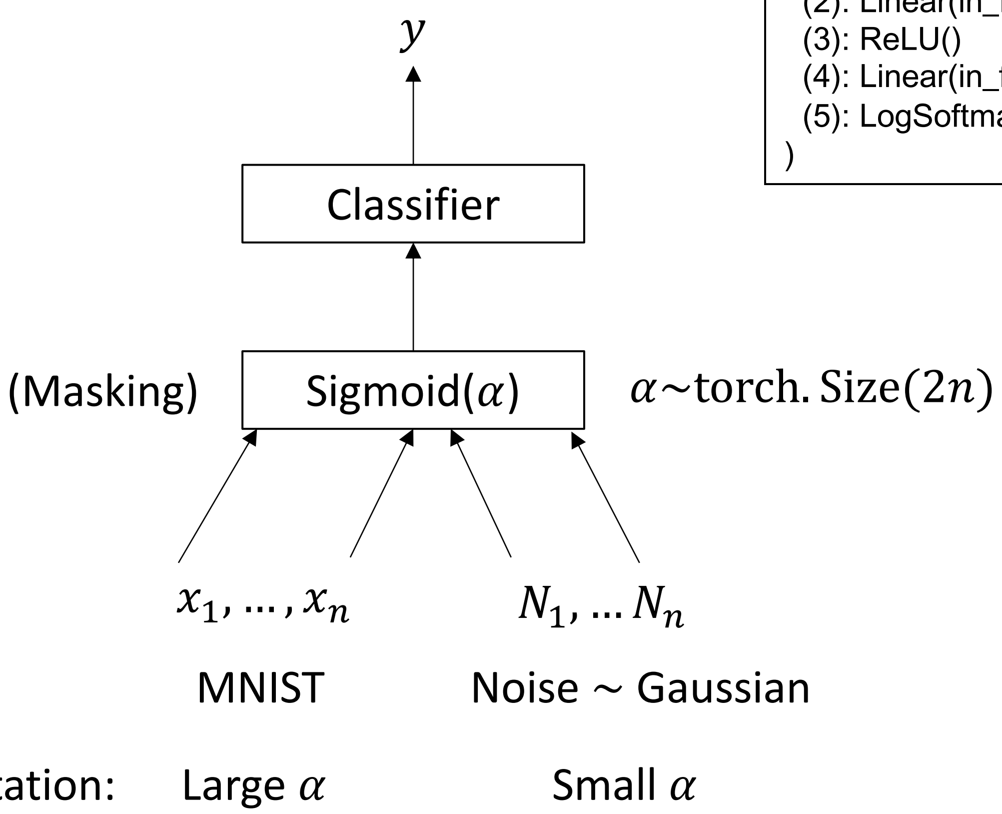
Verifying the Effectiveness of the Search Method

```
Sequential(
  (0): Linear(in_features=1568, out_features=128, bias=True)
  (1): ReLU()
  (2): Linear(in_features=128, out_features=64, bias=True)
  (3): ReLU()
  (4): Linear(in_features=64, out_features=10, bias=True)
  (5): LogSoftmax(dim=1)
)
```



(a) MNIST sample belonging to the digit '7'.

$$n = 28^2 = 784$$



Verifying the Effectiveness of the Search Method

3L Experiment (15 epochs)

The basic model is evaluated to have accuracy 0.974.

Searching without FLOPS loss

Evaluation (Accuracy)	Search Target Size Ratio	Search Result Size Ratio	Search Result Split
0.975	/	0.589	[560, 363]

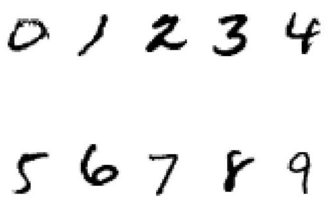
==> $[act(\alpha_{1:n}), act(\alpha_{n:2n})]$

$$act(\alpha) = count(sigmoid(\alpha_i) > 0.01)$$

Searching with FLOPS loss

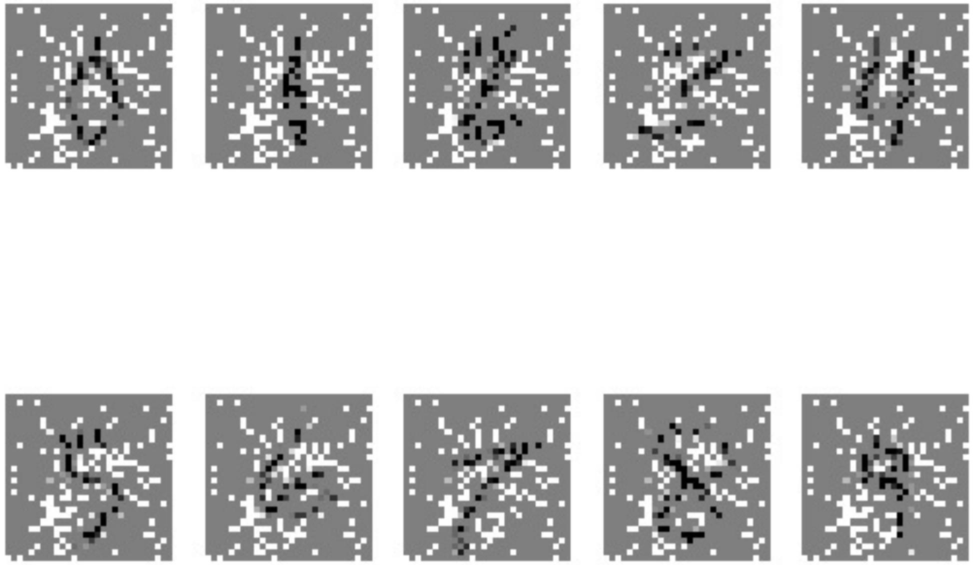
Evaluation (Accuracy)	Search Target Size Ratio	Search Result Size Ratio	Search Result Split
0.758	0.01	0.012	[20, 0]
0.937	0.04	0.040	[63, 0]
0.952	0.05	0.050	[78, 0]
0.970	0.10	0.100	[154, 3]
0.976	0.30	0.265	[336, 79]
0.977	0.50	0.452	[453, 255]
0.975	0.75	0.703	[588, 514]
0.976	1.0	0.951	[726, 765]

==> Suitable compression ratio without significant performance drop

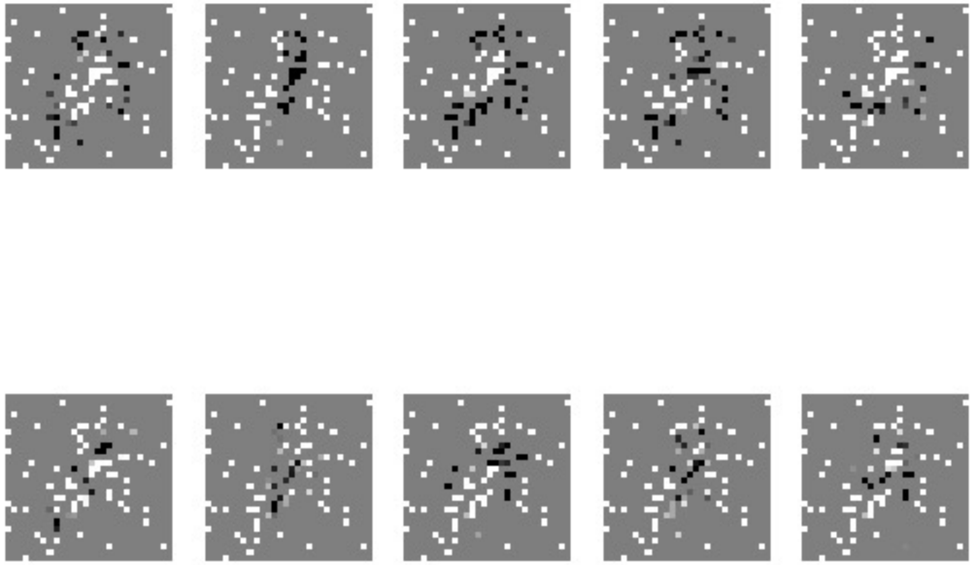


Verifying the Effectiveness of the Search Method

0.10 ratio (~20% of the original image)



0.05 ratio (~10% of the original image)



Grey cells represent pruned dimensions

All digits combined covers ~50% of the grids (28*28)

Model learns to read dotted lines of writing

Verifying the Effectiveness of the Search Method

11L Experiment (20 epochs)

The basic model is evaluated to have accuracy 0.969.

Searching without FLOPS loss

Evaluation (Accuracy)	Search Target Size Ratio	Search Result Size Ratio	Search Result Split
0.961	/	0.467	[425, 308]

Searching with FLOPS loss

Evaluation (Accuracy)	Search Target Size Ratio	Search Result Size Ratio	Search Result Split
0.113	0.01	0.010	[16, 0]
0.794	0.04	0.036	[51, 7]
0.834	0.05	0.046	[69, 4]
0.916	0.10	0.100	[125, 32]
0.946	0.30	0.262	[288, 124]
0.960	0.50	0.456	[429, 287]
0.964	0.75	0.701	[539, 561]
0.940	1.0	0.963	[732, 778]

```
Sequential(
  (0): Linear(in_features=1568, out_features=119, bias=True)
  (1): ReLU()
  (2): Linear(in_features=119, out_features=95, bias=True)
  (3): ReLU()
  (4): Linear(in_features=95, out_features=76, bias=True)
  (5): ReLU()
  (6): Linear(in_features=76, out_features=61, bias=True)
  (7): ReLU()
  (8): Linear(in_features=61, out_features=48, bias=True)
  (9): ReLU()
  (10): Linear(in_features=48, out_features=39, bias=True)
  (11): ReLU()
  (12): Linear(in_features=39, out_features=31, bias=True)
  (13): ReLU()
  (14): Linear(in_features=31, out_features=25, bias=True)
  (15): ReLU()
  (16): Linear(in_features=25, out_features=20, bias=True)
  (17): ReLU()
  (18): Linear(in_features=20, out_features=16, bias=True)
  (19): ReLU()
  (20): Linear(in_features=16, out_features=10, bias=True)
  (21): LogSoftmax(dim=1)
)
```

Verifying the Effectiveness of the Search Method

Ratio	3L	4L	5L	11L 15 epoch	11L 20 epoch
0.01	0	0	0	2	0
0.04	0	0	5	4	7
0.05	0	1	8	5	4
0.1	3	9	9	24	32
0.3	79	92	113	140	124
0.5	255	248	256	306	287
0.75	514	519	514	549	561
1	765	762	769	768	778

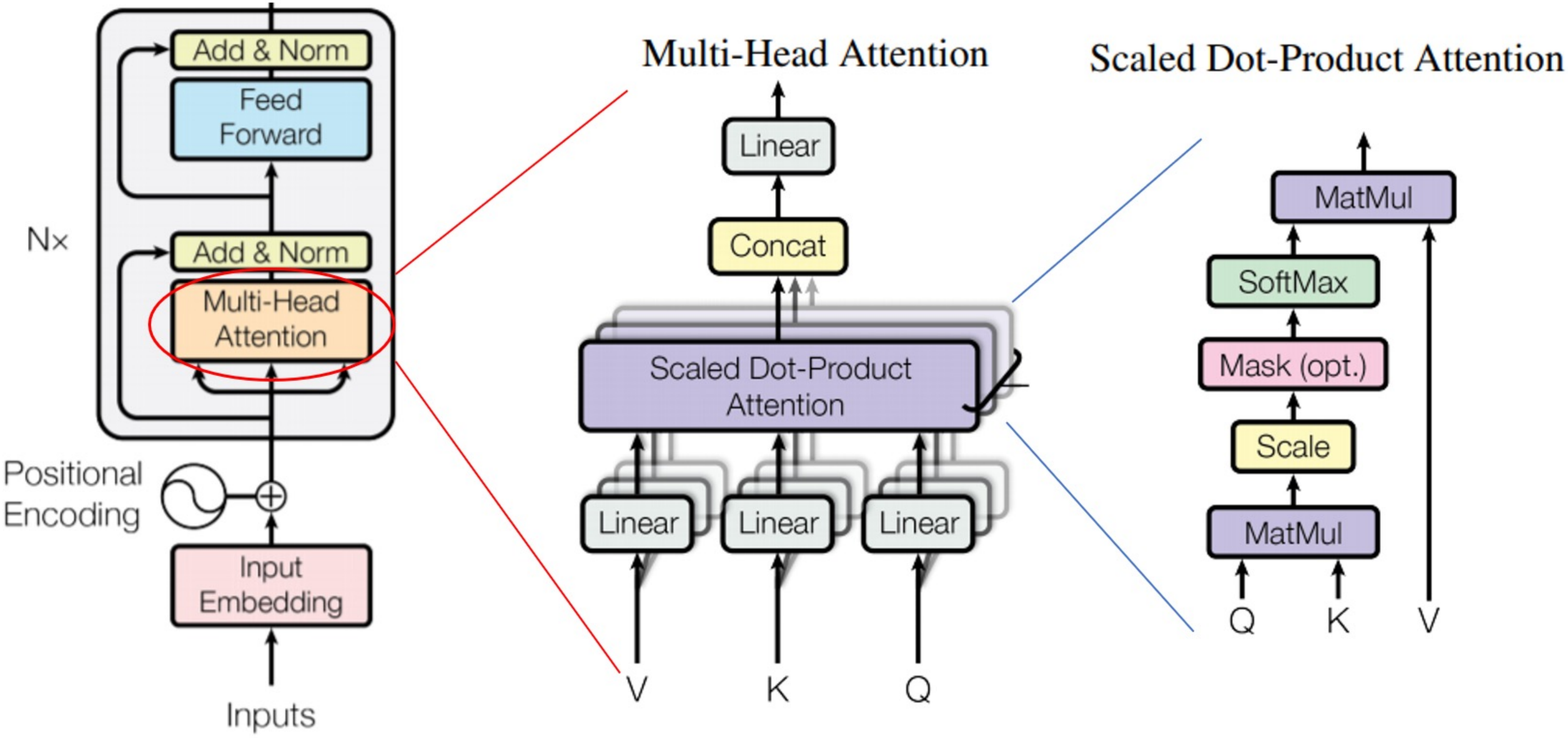
The number of noise dimensions used in the resulting model

Ratio	3L	4L	5L	11L 15 epoch	11L 20 epoch
0.01	0.758	0.765	0.633	0.113	0.113
0.04	0.937	0.914	0.924	0.797	0.794
0.05	0.952	0.935	0.937	0.865	0.834
0.1	0.97	0.965	0.963	0.925	0.916
0.3	0.976	0.972	0.974	0.957	0.946
0.5	0.977	0.976	0.973	0.967	0.96
0.75	0.975	0.972	0.975	0.962	0.964
1	0.976	0.971	0.972	0.954	0.94

The accuracy of the resulting model

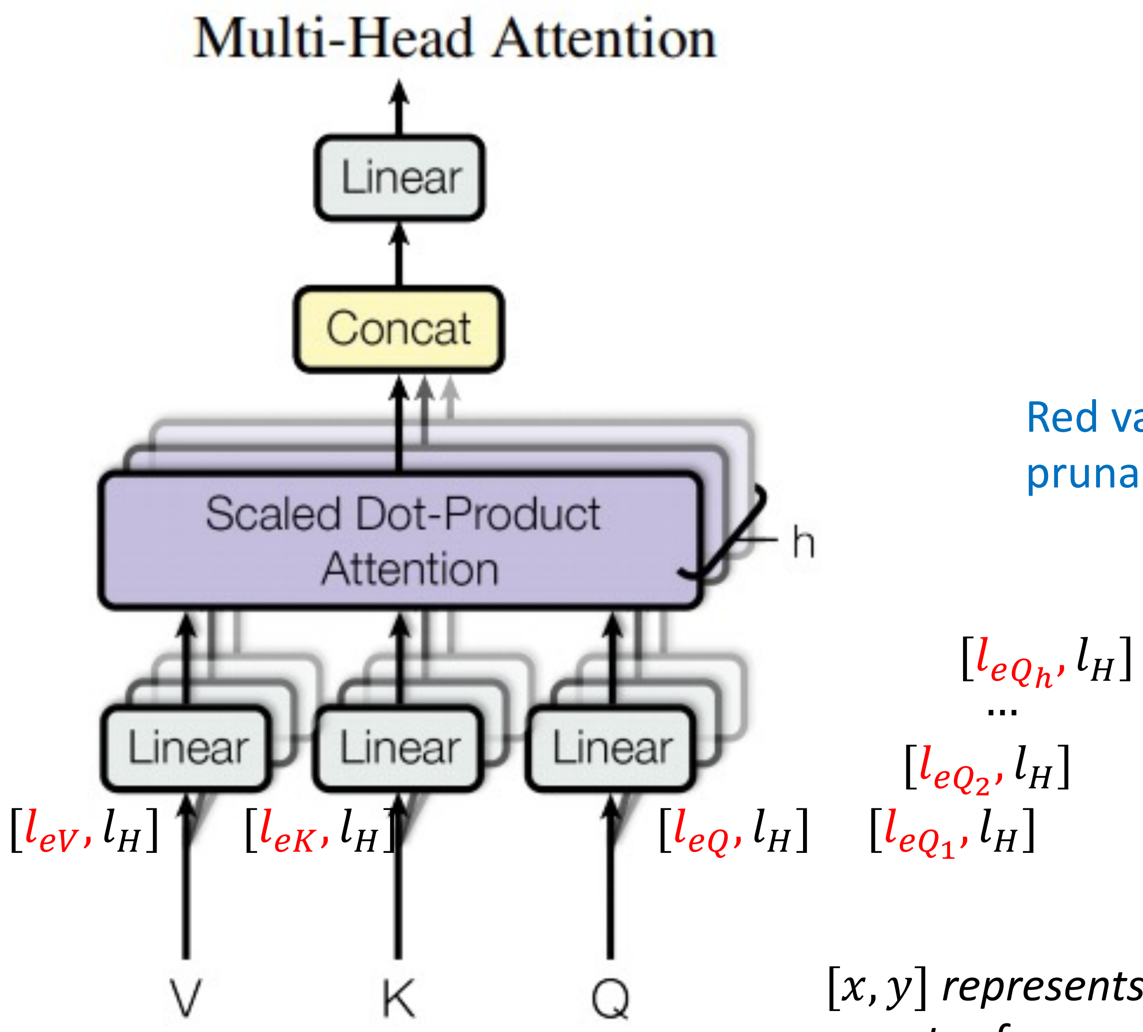
The search method performance is not good in deeper model

Search Space – Overview



5.4 Search Space in the report

Search Space - Input Embedding Dimensions



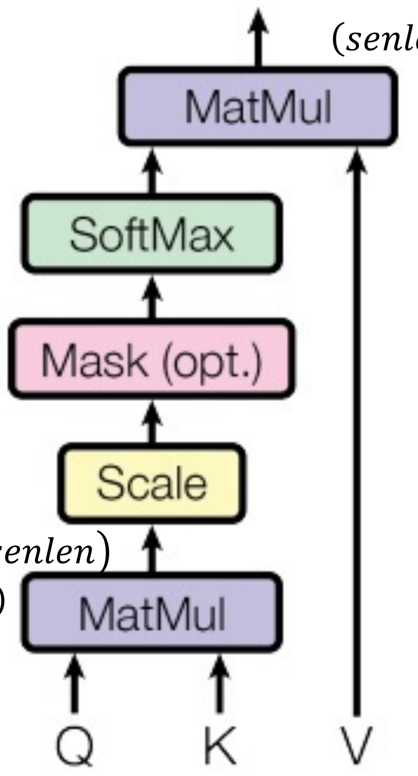
Red variable represents the prunable dimensions

$[x, y]$ represents a linear transformation of a vector from x dimensions to y dimensions.

5.4.1 Input Embedding in the report

Search Space – QKV Hidden Representation Dimensions (Low rank Multi-head attention)

Scaled Dot-Product Attention



$(senlen, senlen)\times(senlen, \sum l_{V_i}) \Rightarrow (senlen, \sum l_{V_i})$

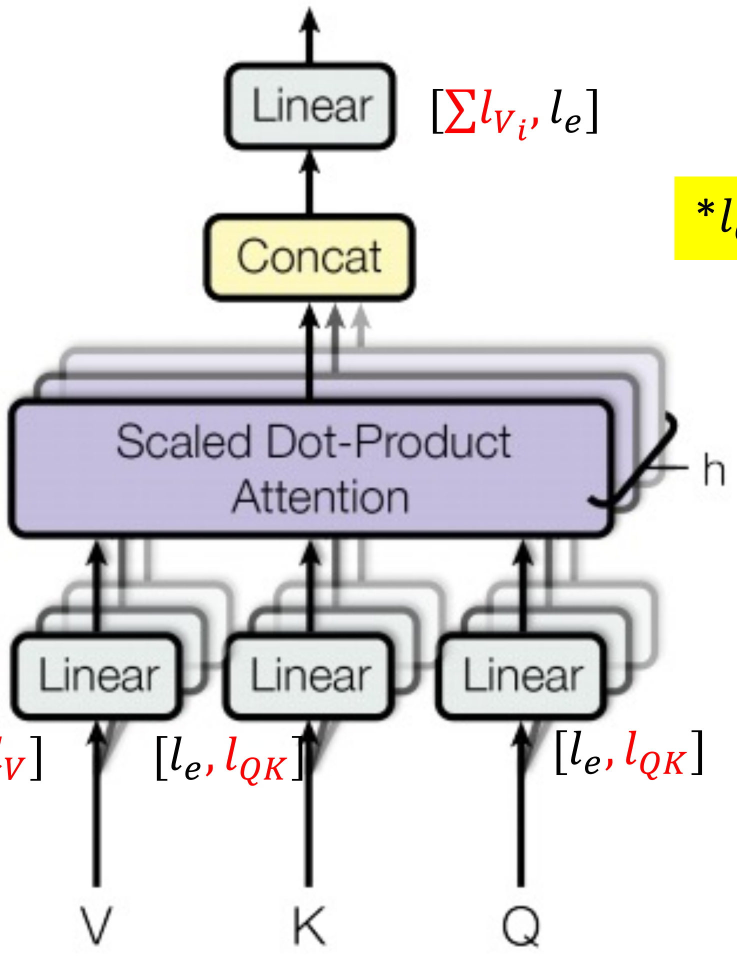
$\mathrm{softmax}(\frac{QK^T}{\sqrt{d_k}})V$

$(senlen, \sum l_{QK_i})\times(\sum l_{QK_i}, senlen) \Rightarrow (senlen, senlen)$

$[l_e, l_{QK_1}]_1, [l_e, l_{QK_2}]_2, \dots, [l_e, l_{QK_h}]_h \Rightarrow [l_e, \sum l_{QK_i}]$

Multihead attention forward requires $l_{Q_i} = l_{K_i} = l_{QK_i}$

Multi-Head Attention



$[\sum l_{V_i}, l_e]$

$*l_{QKV_i} \ge n^{[32]}$

$[l_e, l_{QK_h}]$

...

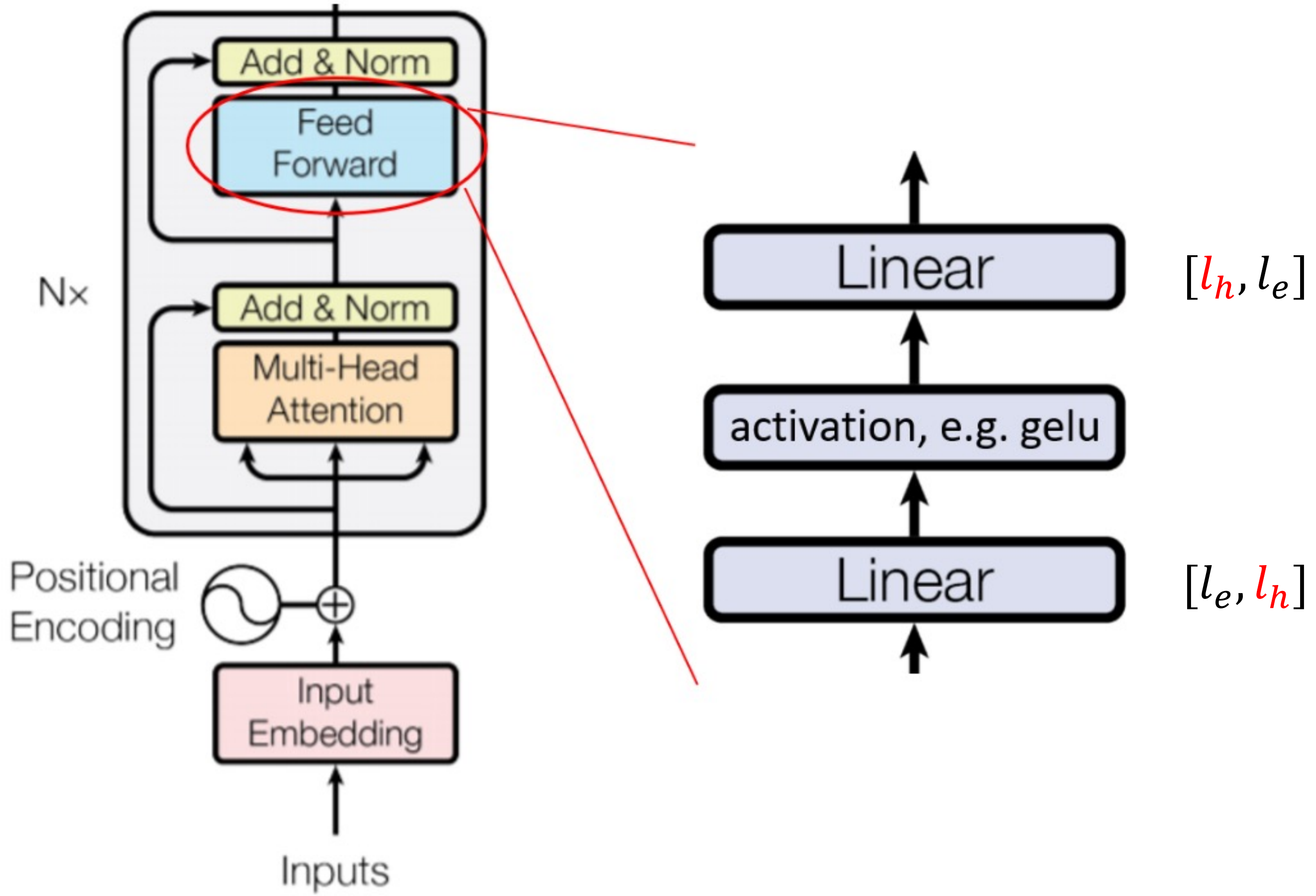
$[l_e, l_{QK_2}]$

$[l_e, l_{QK_1}]$

$[l_e, l_V]$ $[l_e, l_{QK}]$ $[l_e, l_{QK}]$

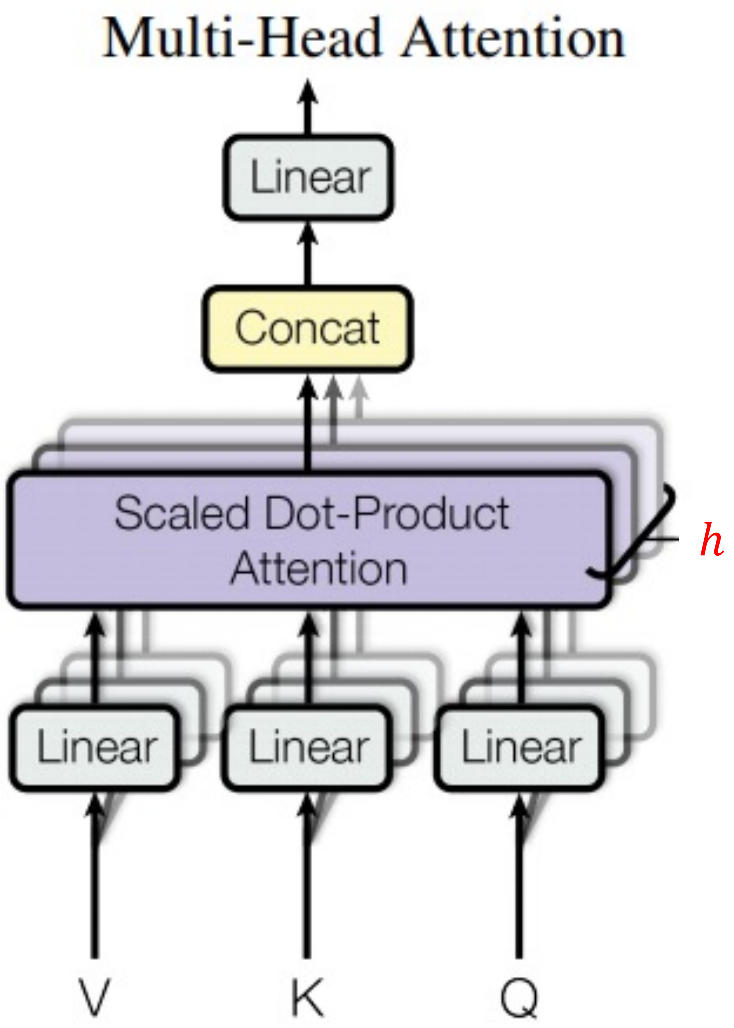
5.4.2 QKV Hidden Representation in the report

Search Space – Feed Forward Intermediate Representation Dimensions



5.4.3 Feed Forward Intermediate Representation in the report

Search Space – Multi-head



5.4.4 Multi-heads Pruning in the report

Result: Searching on Input Token Embedding

Searching without FLOPS loss:

Task	Evaluation	Search Result Size Ratio
CoLA	0.236 mcc (decreased by 0.19)	0.406
RTE	0.621 acc (-6.89%)	0.456
SST-2	0.894 acc (-2.50%)	0.455

TinyBERT distilled model for comparison:

	CoLA (mcc)	RTE (accuracy)	SST-2 (accuracy)
reproduced 4layer-312dim TinyBERT performance (10, 10)	0.426	0.667	0.917

5.6.1 Input Embedding Pruning in the report

Searching with FLOPS loss:

Task	Evaluation	Search Target Size Ratio	Search Result Size Ratio
CoLA	0.267 mcc (decreased by 0.159)	0.5	0.660
CoLA	0.289 mcc (decreased by 0.137)	0.75	0.828
CoLA	0.355 mcc (decreased by 0.071)	1.0	0.974
RTE	0.646 acc (-3.14%)	0.5	0.663
RTE	0.646 acc (-3.14%)	0.75	0.825
RTE	0.653 acc (-2.09%)	1.0	0.975
SST-2	0.905 acc (-1.30%)	0.5	0.662
SST-2	0.906 acc (-1.19%)	0.75	0.852
SST-2	0.909 acc (-0.872%)	1.0	0.974

Why Large Dataset is less vulnerable to pruning?

1. Training involves more global steps? (10 epochs of large set > 10 epochs of small set)
2. Training data is more diverged, more general

Result: Model Size

	Number of parameters	Ratio to original model
Bert-base-uncased	110074370	1.0
TinyBERT 4L	14591258	0.132
TinyBERT 4L input pruned 0.5	14112026	0.128

Conclusion: it is not efficient to prune away the input dimensions for compression, there are little redundancy in the dimensions of the input embeddings.