Compression Methods for Transformer-based Models

韦阳

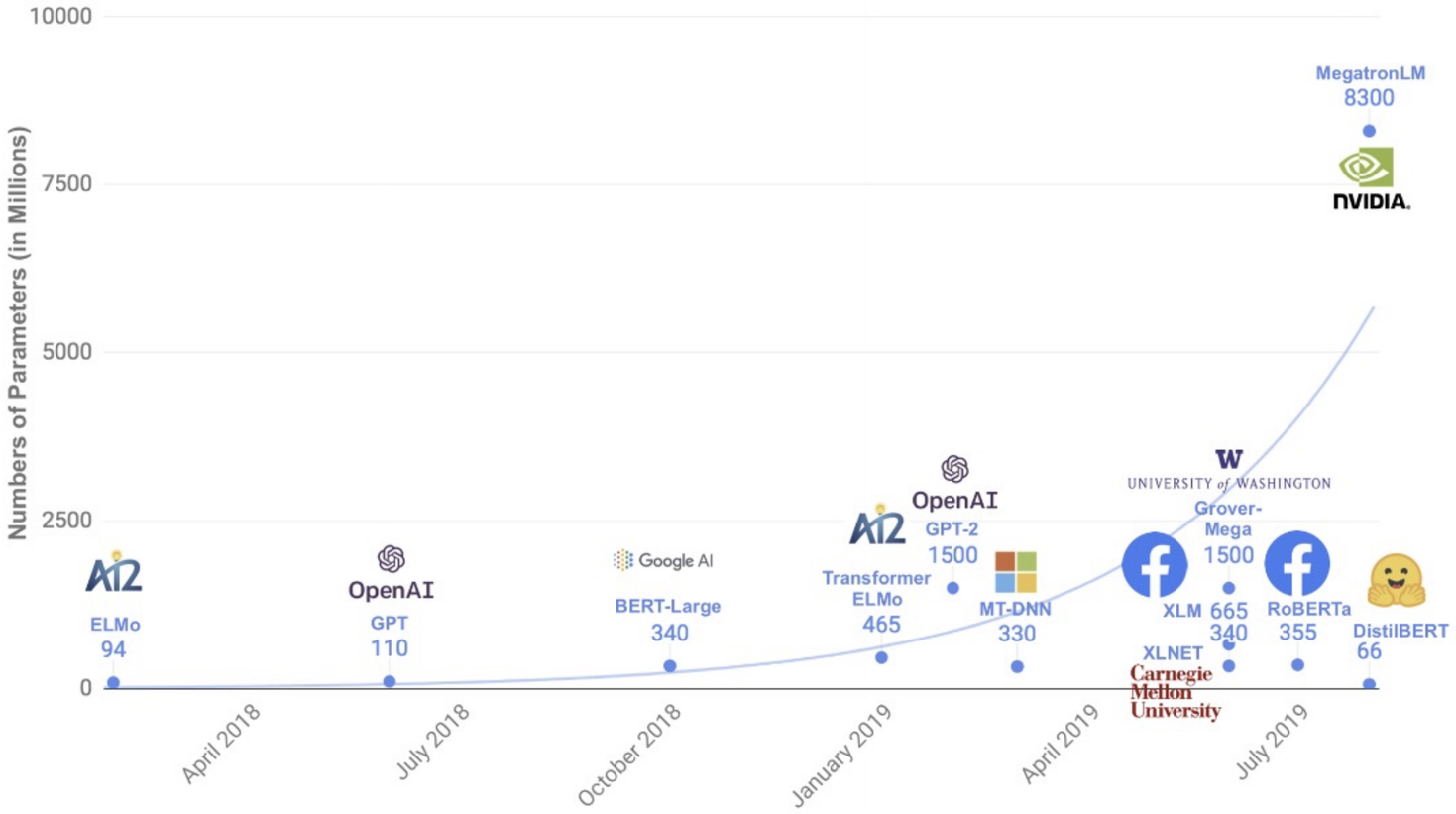
weiyang.god@bytedance.com
AI Lab - MLNLC

Outline

- Background
- Embedding Compression
- Attention and FFN Compression
- Layer-wise Compression
- Conclusion

Why Compression?

- Growing usage of Transformer-based models.
- Increasing size of pre-trained language models.
- Demand for AI-powered devices.
- Transformer is more complex than CNN, etc.



App	Size/MB	Service
谷歌翻译	45	文本/图片
腾讯翻译君	85	文本
网易有道词典	95	文本/图片
百度翻译	113	文本/图片
有道翻译君	225	文本/图片/语音/同传
微软翻译	98	文本/图片
搜狗翻译	96	文本/图片

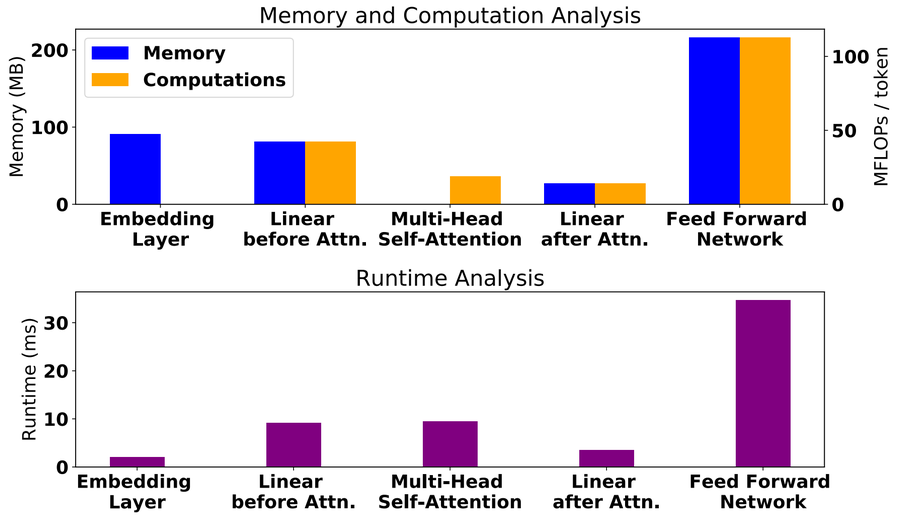
Compression Methods

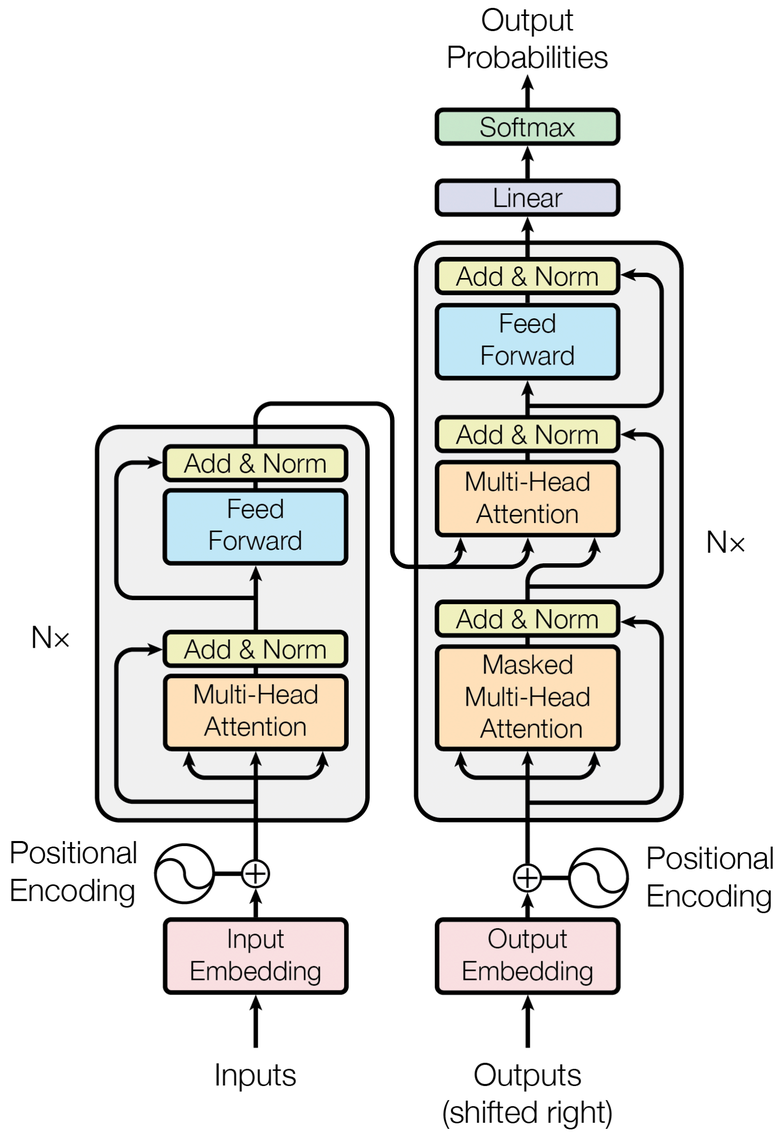
- Pruning (structured, unstructured)
- Quantization (QAT, post-training)
- Knowledge Distillation
- Architecture-Invariant Compression

Transformer Architecture

- E.g. Transformer-big (4w vocab+1024 dmodel+6e6d+4096d ffn)
  - Embedding: 4w*1024*2=81m
  - Attention: 1024*1024*4*3*6=75m
  - FFN: 1024*4096*2*2*6=100m
  - Total: 258m (984MB on disk)

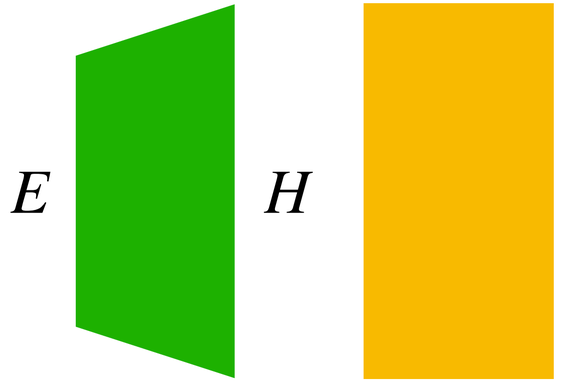
e.g. BERT





Factorized Embedding Parameterization

- Context-dependent representations (H) are more important than context-independent representations (E).
- Reducing E is beneficial to increasing H ($VE + EH$).
- If $E = H$, then the embeddings size (VE) increases as H increases.
- $E \ll H$ (e.g. $E = 256, H = 1024$).



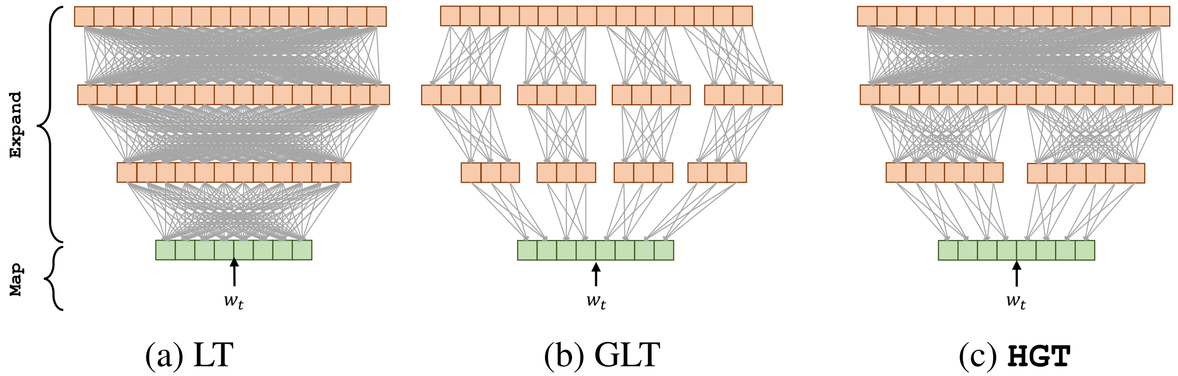
Effect of Embedding Size

Model	E	Parameters	SQuAD1.1	SQuAD2.0	MNLI	SST-2	RACE	Avg
ALBERT base not-shared	64	87M	89.9/82.9	80.1/77.8	82.9	91.5	66.7	81.3
	128	89M	89.9/82.8	80.3/77.3	83.7	91.5	67.9	81.7
	256	93M	90.2/83.2	80.3/77.4	84.1	91.9	67.3	81.8
	768	108M	90.4/83.2	80.4/77.6	84.5	92.8	68.2	82.3
ALBERT base all-shared	64	10M	88.7/81.4	77.5/74.8	80.8	89.4	63.5	79.0
	128	12M	89.3/82.3	80.0/77.1	81.6	90.3	64.0	80.1
	256	16M	88.8/81.5	79.1/76.3	81.5	90.3	63.4	79.6
	768	31M	88.6/81.5	79.2/76.6	82.0	90.6	63.3	79.8

Table 3: The effect of vocabulary embedding size on the performance of ALBERT-base.

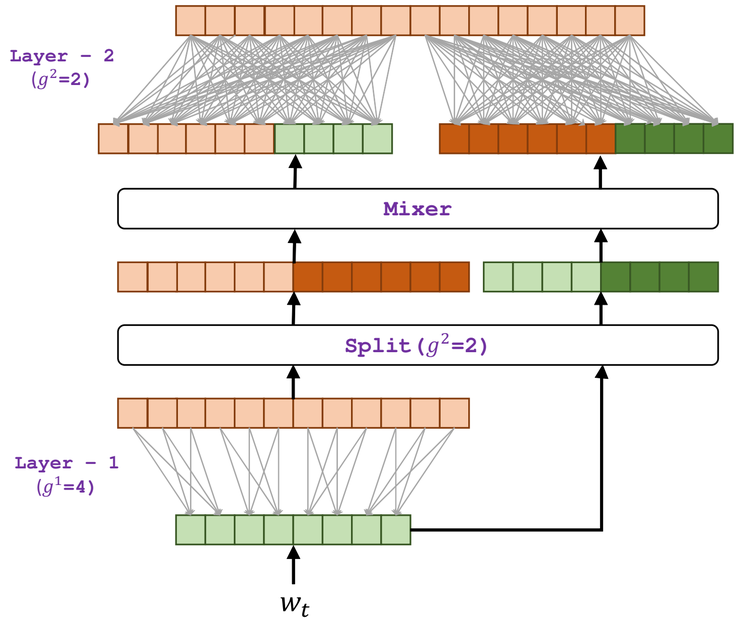
- Cross-layer parameters not shared: larger E is better.
- Cross-layer parameters shared: $E = 128$ is the best (In our practice, $E = 256$ is the best).
- Simple but effective method (little loss, fewer parameters, more operations).

Deep Factorized Input Token Embeddings (DeFINE)



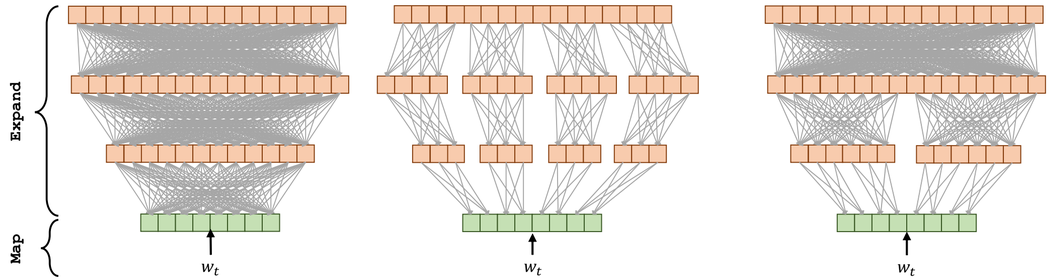
(a) LT (b) GLT (c) **HGT**

- Motivation: deeper is better.
- Map-Expand-Reduce (MER) + hierarchical group transformation (HGT) + residual connection.



HGT

- Use group transformation for fewer parameters.
- Different numbers of groups for global information.
- If $d_i \ll d_o$, parameters are need to be stored (will increase the operations in inference), otherwise only need to store the results after reducing for inference.
- If $d_i = d_o$, maybe FFN is a better choice (just need to cache the final results).



(a) LT (b) GLT (c) **HGT**

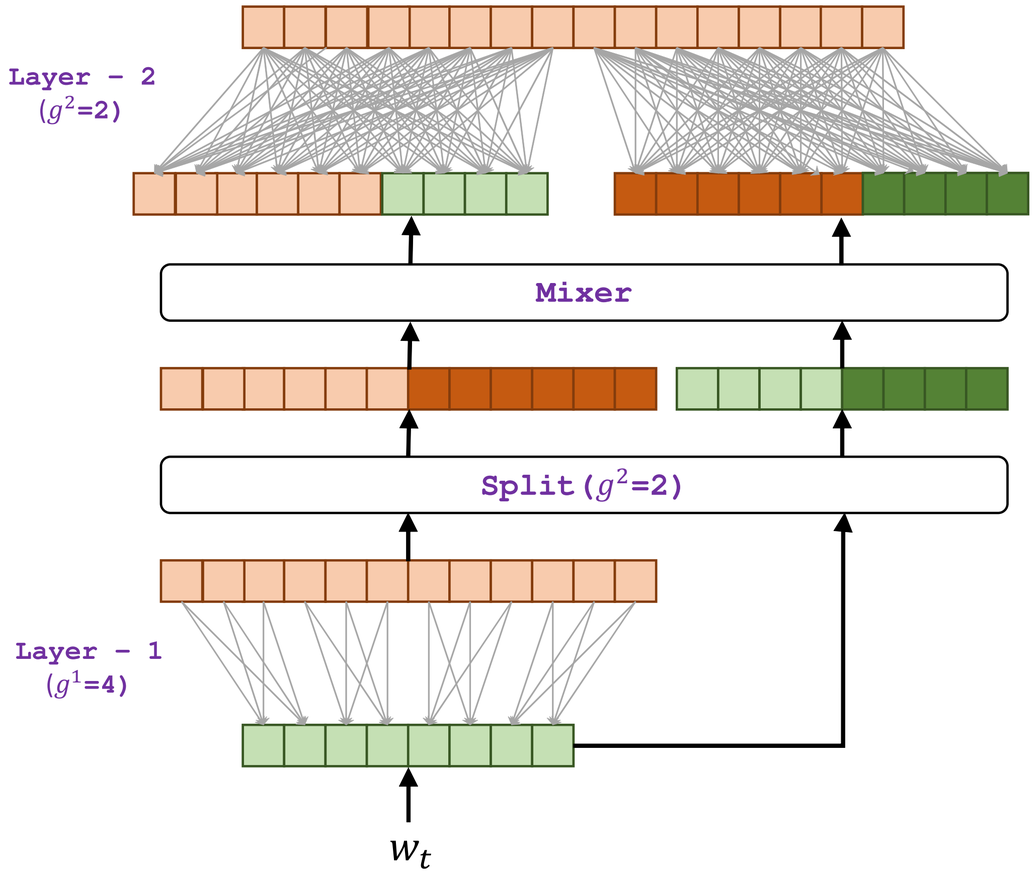
$$\hat{\mathbf{e}}_i^l = \begin{cases} \mathcal{F}_G\left(\mathbf{e}_i, \mathbf{W}^l, g^l\right), & l = 1 \\ \mathcal{F}_G\left(\hat{\mathbf{e}}_i^{l-1}, \mathbf{W}^l, g^l\right), & 1 < l \leq N \end{cases}$$

$$g^l = \max\left(\lfloor \frac{g_{max}}{2^{l-1}} \rfloor, 1\right)$$

$Vd_i + P_{expand}$ Vd_o

Residual Connection

- Concatenate the input and output of any layers using Split and Mixer.
- Can not directly add them (different dimensions).



Experiments on NMT

Model	Checkpoint Averaging?	Parameters (in millions)	BLEU (EN-DE) newstest2014	BLEU (EN-DE) newstest2017
Transformer (Vaswani et al., 2017)	✓	–	**27.30**	–
Transformer + SRU (Lei et al., 2018)	✓	90 M	27.1	**28.30**
Transformer (OpenNMT impl.) (Klein et al., 2017)	✓	92 M	26.89	28.09
Transformer	✗	92 M	25.01	25.81
Transformer + **DeFINE**	✗	**68 M**	27.01	28.25

Table 3: Results of Transformer-based model (with and without **DeFINE**) on the task of neural machine translation. **DeFINE** attains similar performance to checkpoint averaging, but with fewer parameters.

$$d_i = 128$$
$$d_o = 512$$

- Parameters are reduced by 26% with little loss on BLEU.
- Factorized embedding parameterization also reduces about 24m parameters. Which is better?

Impact of Depth and Width

Depth of	Dimensions of			# Parameters	Perplexity	
DeFINE (N)	$\mathbf{e}_i$ (n)	$\mathbf{e}_o$ (m)	$\hat{\mathbf{e}}_i$ (k)	(in millions)	Val	Test
3	256	256	1024	**33.02**	39.99	41.17
			1536	33.15	40.08	41.25
			2048	33.29	40.23	41.37
7	384	384	1024	40.73	36.95	38.01
			1536	41.86	36.85	37.81
			2048	43.19	36.95	37.84
11	512	512	1024	49.55	**34.94**	35.94
			1536	52.02	35.25	35.98
			2048	55.02	35.00	**35.92**

(a) Depth (N) vs width (k)

- Larger N, n and m lead to better performance.
- Larger k is useless for fixed N, n, m (redundant for learning from the same group).

Correlation Map Visualization

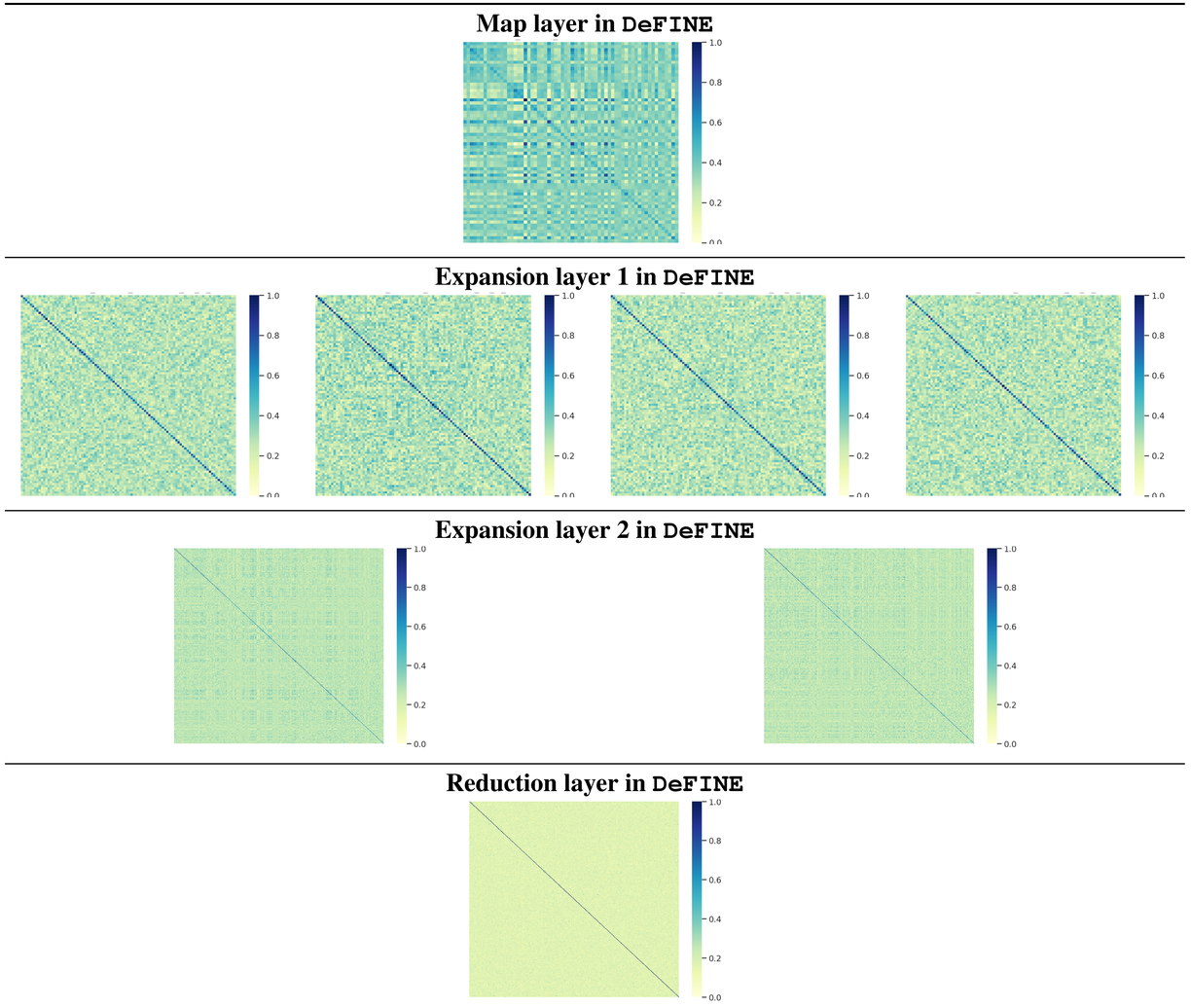


Figure 8: Layer-wise visualization of correlation maps of **DeFINE** embeddings when $n = 64$.

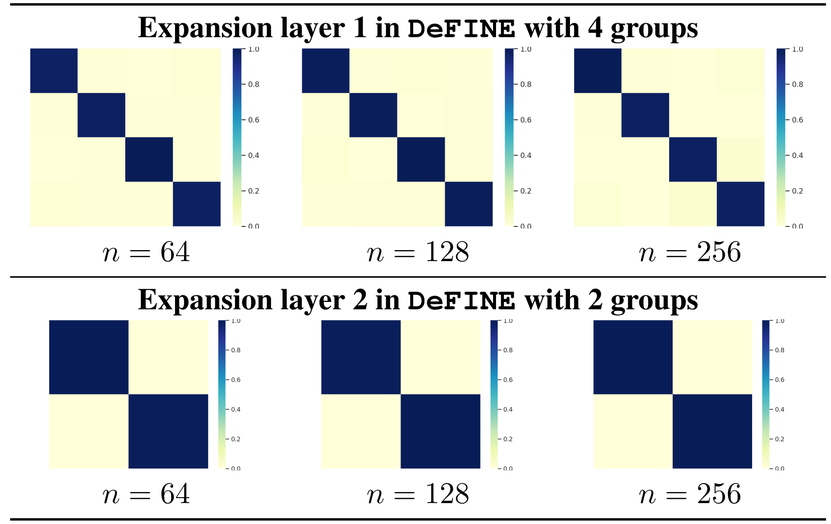
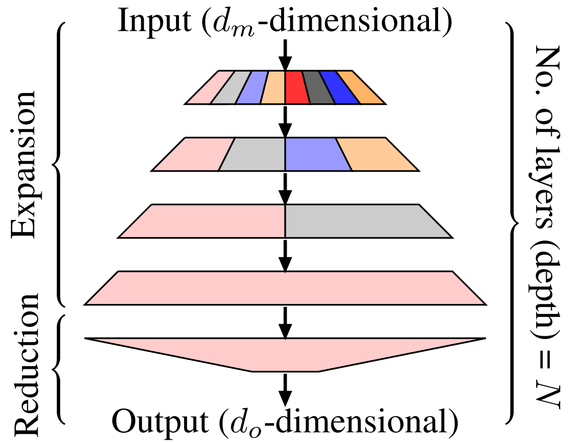


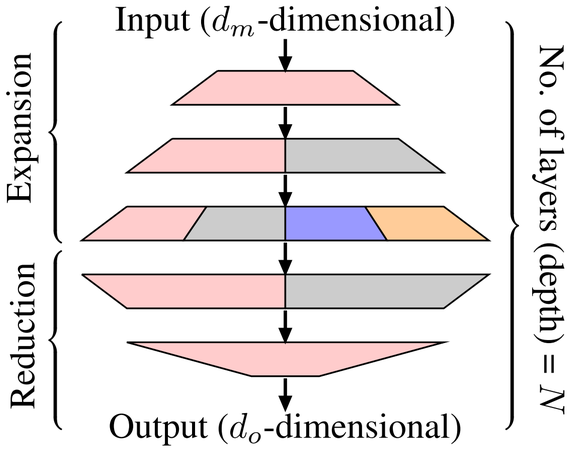
Figure 11: Groups in expansion layers of **DeFINE** are orthogonal, suggesting these matrices are learning different representations of their input.

Deep and Light-weight Expand-Reduce Transformation (DExTra)

- Differences from DeFINE:
  - Same layer numbers of expand and reduce (learn smoothly).
  - Wider layers, more groups (fewer parameters).
  - $d_m > d_o$ (fewer calculations on attention).
  - Feature shuffling (global information).
  - No residual connections (no explanation).



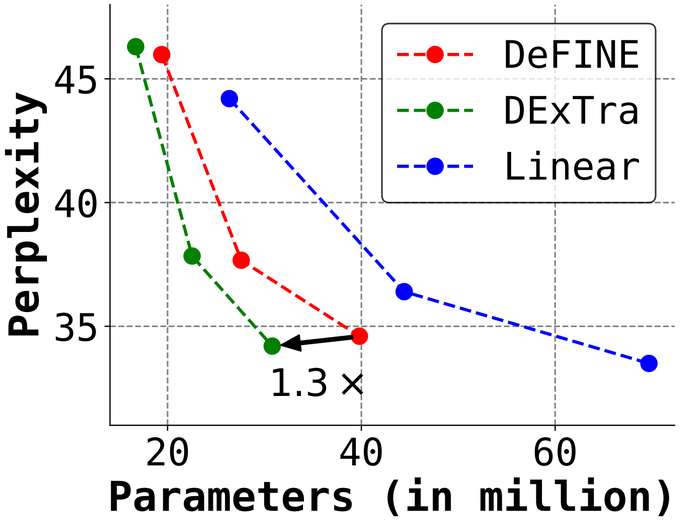
(a) DeFINE [32]



(b) `DExTra` (Ours)

Compared with Other Methods

- DExTra can achieve comparable results with fewer parameters.
- DExTra can be used in the entry of each layer (reduce the dimension of attention).



Inspirations (Embedding)

- Complex structure is better (deeper and wider layers, group transformation, etc).
- Limit $d_i > d_o$, so that the outputs can be cached, and finally apply a linear transformation to them ($d_o \rightarrow d_m$).
- Special design for different tokens (e.g. different dimensions for tokens with different frequencies).

Importance of Attention Heads

- Removing one head from one layer has no impact.
- Preserving only one head in each layer has small impact.
- Enc-Dec heads are very important.
- These phenomenons are universal in most datasets.

Layer \ Head	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	0.03	0.07	0.05	-0.06	0.03	**-0.53**	0.09	**-0.33**	0.06	0.03	0.11	0.04	0.01	-0.04	0.04	0.00
2	0.01	0.04	0.10	**0.20**	0.06	0.03	0.00	0.09	0.10	0.04	**0.15**	0.03	0.05	0.04	0.14	0.04
3	0.05	-0.01	0.08	0.09	0.11	0.02	0.03	0.03	-0.00	0.13	0.09	0.09	-0.11	**0.24**	0.07	-0.04
4	-0.02	0.03	0.13	0.06	-0.05	0.13	0.14	0.05	0.02	0.14	0.05	0.06	0.03	-0.06	-0.10	-0.06
5	**-0.31**	-0.11	-0.04	0.12	0.10	0.02	0.09	0.08	0.04	**0.21**	-0.02	0.02	-0.03	-0.04	0.07	-0.02
6	0.06	0.07	**-0.31**	0.15	-0.19	0.15	0.11	0.05	0.01	-0.08	0.06	0.01	0.01	0.02	0.07	0.05

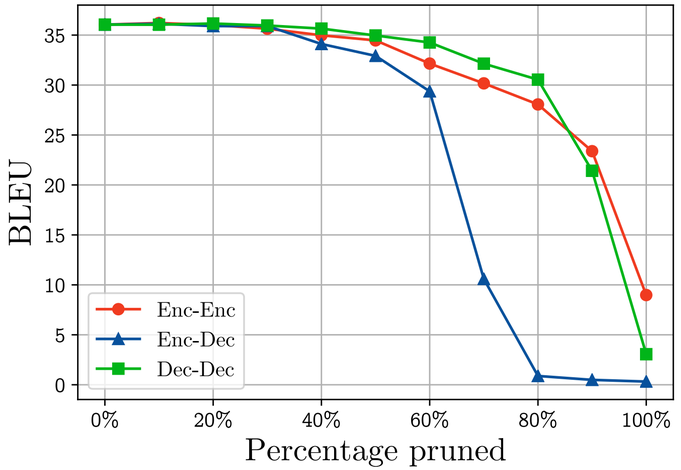
Table 1: Difference in BLEU score for each head of the encoder's self attention mechanism. Underlined numbers indicate that the change is statistically significant with $p < 0.01$. The base BLEU score is 36.05.

Layer	Enc-Enc	Enc-Dec	Dec-Dec
1	**-1.31**	**0.24**	-0.03
2	-0.16	0.06	0.12
3	0.12	0.05	0.18
4	-0.15	-0.24	0.17
5	0.02	**-1.55**	-0.04
6	**-0.36**	**-13.56**	0.24

Table 2: Best delta BLEU by layer when only one head is kept in the WMT model. Underlined numbers indicate that the change is statistically significant with $p < 0.01$.

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 $p < 0.01$.



Iterative Pruning of Attention Heads

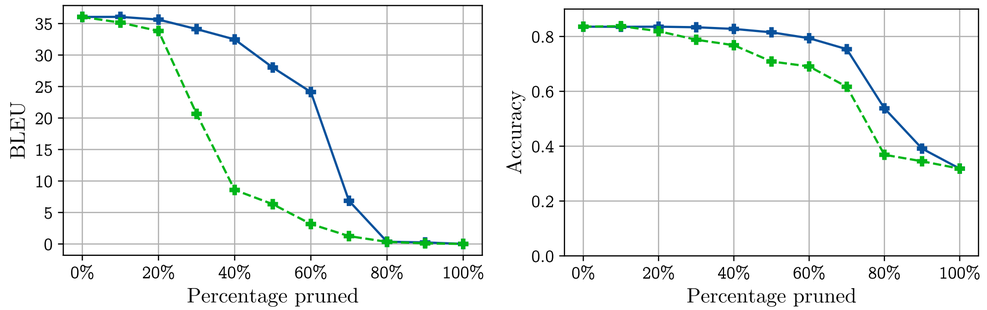
- In the early stages of training, the importance of heads is basically determined.
- Fewer heads in lower layers, more heads in higher layers.
- More heads for Enc-Dec attention.
- Hand-designed importance scores are not optimal and other methods can be considered.

Importance Score

$$\mathrm{MHAtt}(\mathbf{x}, q) = \sum_{h=1}^{N_h} \xi_h \mathrm{Att}_{W_k^h, W_q^h, W_v^h, W_o^h}(\mathbf{x}, q)$$

$$I_h = \mathbb{E}_{x \sim X} \left| \frac{\partial \mathcal{L}(x)}{\partial \xi_h} \right|$$

$$I_h = \mathbb{E}_{x \sim X} \left| \mathrm{Att}_h(x)^T \frac{\partial \mathcal{L}(x)}{\partial \mathrm{Att}_h(x)} \right|$$



(a) Evolution of BLEU score on `newstest2013` when heads are pruned from WMT.

(b) Evolution of accuracy on the MultiNLI-matched validation set when heads are pruned from BERT.

Figure 3: Evolution of accuracy by number of heads pruned according to I_h (solid blue) and individual oracle performance difference (dashed green).

Fixed Self-Attention Patterns

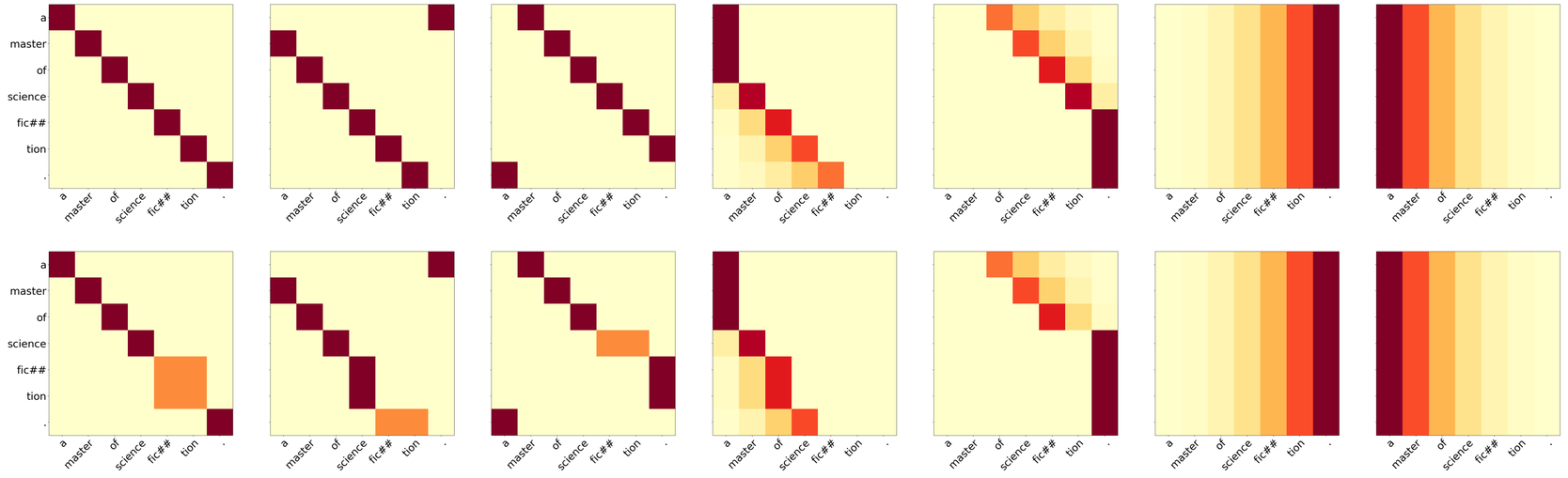


Figure 1: Token-based (upper row) and word-based (lower row) fixed attention patterns for the example sentence "*a master of science fic## tion .*". The word-based patterns treat the subwords "*fic##*" and "*tion*" as a single token.

- Fix seven heads using above patterns and only preserve one learnable head.
- Only used in Encoders, not Decoders or Enc-Dec (above patterns may not suitable, e.g. 5th pattern).

Given the $i\text{-}th$ word within a sentence of length n, we define the following patterns:

1. the current token: a fixed attention weight of 1.0 at position i,
2. the previous token: a fixed attention weight of 1.0 at position $i-1$,
3. the next token: a fixed attention weight of 1.0 at position $i+1$,
4. the larger left-hand context: a function f over the positions 0 to $i-2$,
5. the larger right-hand context: a function f over the positions $i+2$ to n,
6. the end of the sentence: a function f over the positions 0 to n,
7. the start of the sentence: a function f over the positions n to 0.

$$\xi_{i,j}^{(4)} = \begin{cases} f^{(4)}(j) & \text{if } j \leq i-2 \\ 0 & \text{otherwise} \end{cases}$$

where

$$f^{(4)}(j) = \frac{(j+1)^3}{\sum_{j=0}^{i-2}(j+1)^3}$$

Experiments on NMT

- 7F1L is similar to 8L in most cases (except DE-EN).
- 7F1L is better than 1L (fixed heads are indeed helpful).
- More decoder layers, better results.
- Token-based is better than word-based.
- 8F is even better than 8L in some language pairs (the 8th head focuses on the last word).

		Encoder+Decoder layers						
	Encoder heads	1+1	2+1	3+1	4+1	5+1	6+1	6+6
EN–DE	8L	20.61	21.68	22.63	23.02	23.18	23.36	25.02
	$7F_{token}$+1L	20.61	21.58	22.38	23.15	23.10	23.07	24.63
	$7F_{word}$+1L	19.72*	21.43	21.81*	22.83	22.74	22.88*	24.85
	1L	18.14*	19.88*	21.42*	21.71*	22.63*	22.29*	23.87*
DE–EN	8L	25.66	27.28	27.88	28.62	28.71	29.31	30.99
	$7F_{token}$+1L	24.90*	27.01	26.84*	28.09*	28.43	28.61*	30.61
	$7F_{word}$+1L	25.03*	26.72*	27.38*	27.78*	27.82*	28.40*	30.69
	1L	23.76*	25.75*	26.96*	27.34*	27.44*	27.56*	30.17*

Table 1: BLEU scores for the German ↔ English (DE ↔ EN) standard scenario, for different configurations of learnable (L) and fixed (F) attention heads. Scores marked in gray with * are significantly lower than the respective *8L* model scores, at $p < 0.05$. Statistical significance is computed using the *compare-mt* tool (Neubig et al., 2019) with paired bootstrap resampling with 1000 resamples (Koehn, 2004).

Enc. heads	#Param.	EN–DE	DE–EN	EN–VI	VI–EN
8L	91.7M	25.02	30.99	29.85	26.15
$7F_{token}$+1L	88.9M	24.63	30.61	31.05	29.16
$8F_{token}$	88.5M	24.64	30.56	31.45	28.97

Table 5: BLEU scores for the experiments with eight fixed attention heads and 6+6 layers. "#Param." denotes the number of parameters for the EN–DE model.

reduced parameters: $512^2 \cdot 2 \cdot 6 = 3m$

Experiments on Low-Resource Scenarios

- 4k to 1k tokens, 0.1 to 0.3 dropout and tied embeddings.
- 7F1L is better than 8L, 1L and prior work (RNN).

Enc. heads	DE–EN	KO–EN	EN–VI	VI–EN
8L	30.86	6.67	29.85	26.15
$7F_{token}$+1L	**32.95**	8.43	31.05	**29.16**
$7F_{word}$+1L	32.56	**8.70**	**31.15**	28.90
1L	30.22	6.14	28.67	25.03
Prior work	† 33.60	† 10.37	⊎ 27.71	⊎ 26.15

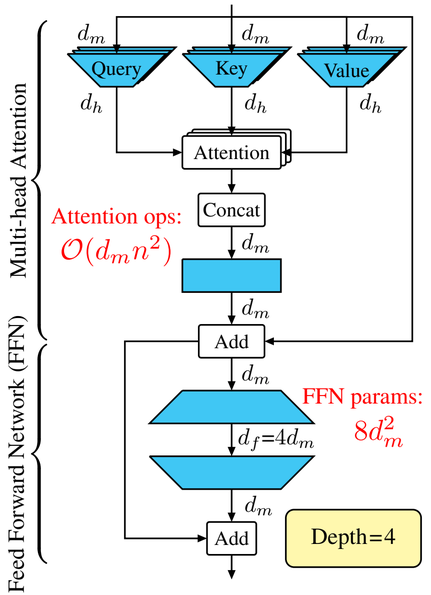
Table 2: BLEU scores obtained for the low-resource scenarios with 6+6 layer configuration. Results marked with † are taken from Sennrich and Zhang (2019), those marked with ⊎ from Kudo (2018).

Pros & Cons

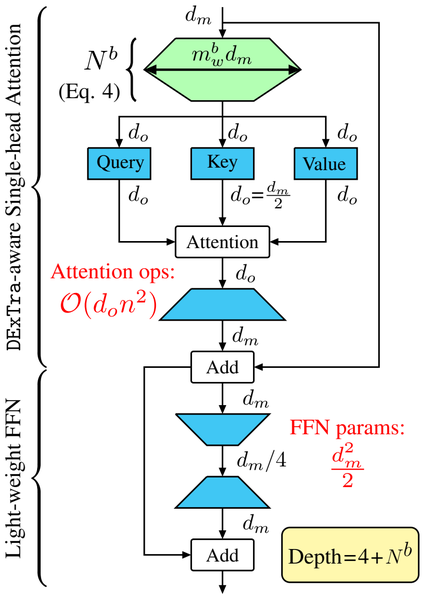
- Simple, fast and fewer parameters (not obvious).
- Better than directly removing the heads.
- Only suitable for Encoders.
- Decoders need to be very deep to enhance the representation.

DeLighT

- Main differences:
  - One DExTra block in the beginning of each layer.
  - One attention head ($d_o = d_m/2$), which will decrease the calculations of attention.
  - One linear transformation before FFN.
  - Decrease FFN hidden dimension.



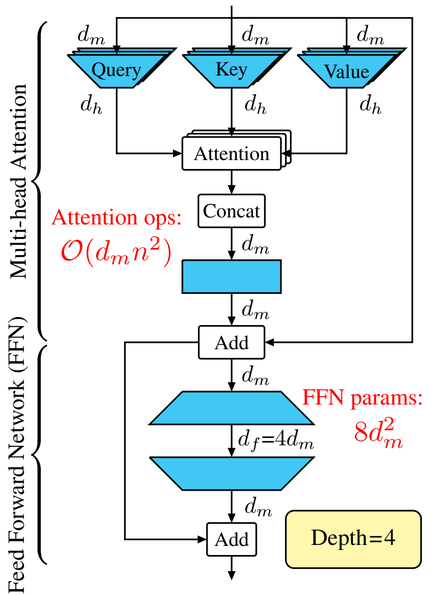
(c) Transformer block [1]



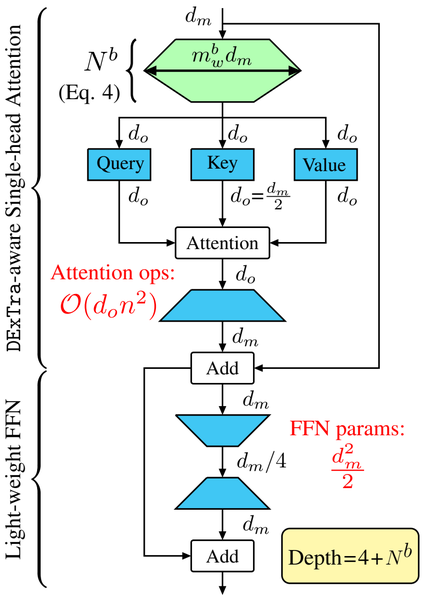
(d) `DeLighT` block (Ours)

DExTra is Helpful

- This is consistent with the previous view that most heads are redundant.
- The parameters of attentions and FFN before are transferred to DExTra block.

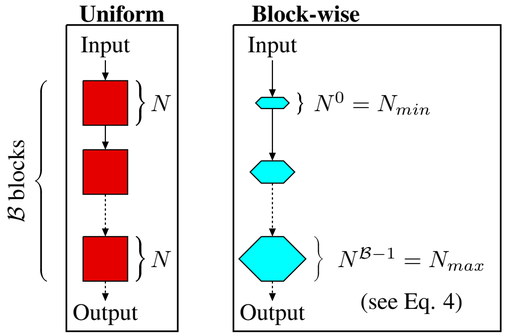


(c) Transformer block [1]

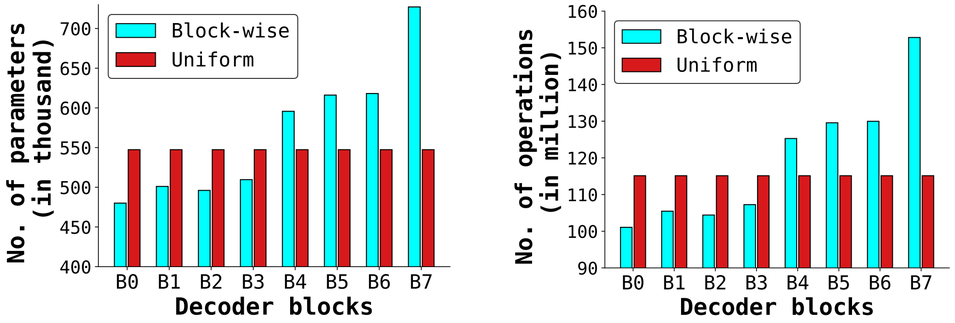


(d) `DeLighT` block (Ours)

Block-wise Scaling



(a) Uniform vs. block-wise



(b) Distribution of parameters and operations within each block

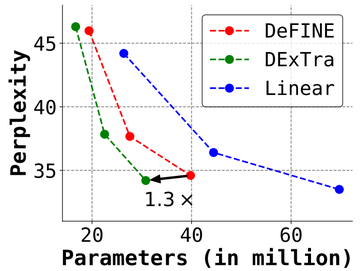
- Deeper layers need more parameters to learn representations.
- Standard Transformer blocks are hard to implement above idea.
- DExTra blocks are useful here (deeper and wider in latter layers).

Ablations

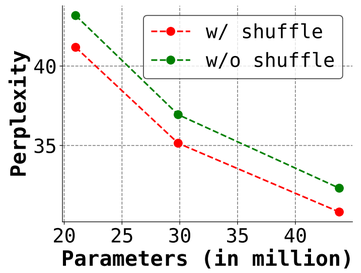
(a)

Row #	N_{min}	N_{max}	m_w	d_m	Depth	Parameters	MACs	Perplexity
Uniform vs. block-wise scaling								
R1	4	4	2	256	43	14.1 M	2.96 B	56.19
R2	8	8	2	256	115	16.6 M	3.49 B	48.58
R3	8	8	4	256	115	22.1 M	4.64 B	45.10
R4	4	8	2	256	92	16.7 M	3.51 B	46.30
R5	4	12	2	256	158	21.0 M	4.41 B	41.18
Varying depth (N_{min} and N_{max} (Eq. 4)								
R6	4	8	2	256	92	16.7 M	3.51 B	46.30
R7	6	8	2	256	102	16.5 M	3.46 B	46.68
R8	4	12	2	256	158	21.0 M	4.41 B	41.18
R9	6	12	2	256	172	20.0 M	4.20 B	42.26
Varying `DExTra`'s width m_w (Eq. 4)								
R10	4	12	2	256	158	21.0 M	4.41 B	41.18
R11	4	12	3	256	158	23.8 M	4.99 B	39.92
R12	4	12	4	256	158	27.1 M	5.69 B	39.10
Varying model width d_m								
R13	4	12	2	256	158	21.0 M	4.41 B	41.18
R14	4	12	2	384	158	29.9 M	6.28 B	35.14
R15	4	12	2	512	158	43.8 M	9.20 B	30.81
Deeper and wider near the Input								
R16	12	4	2	256	158	21.0 M	4.41 B	43.10

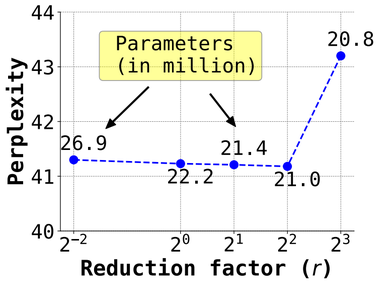
(b)



(c)

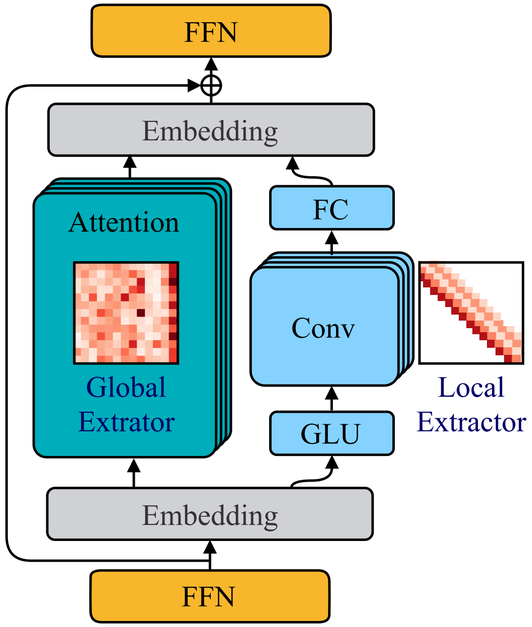


(d)



Long-Short Range Attention (LSRA)

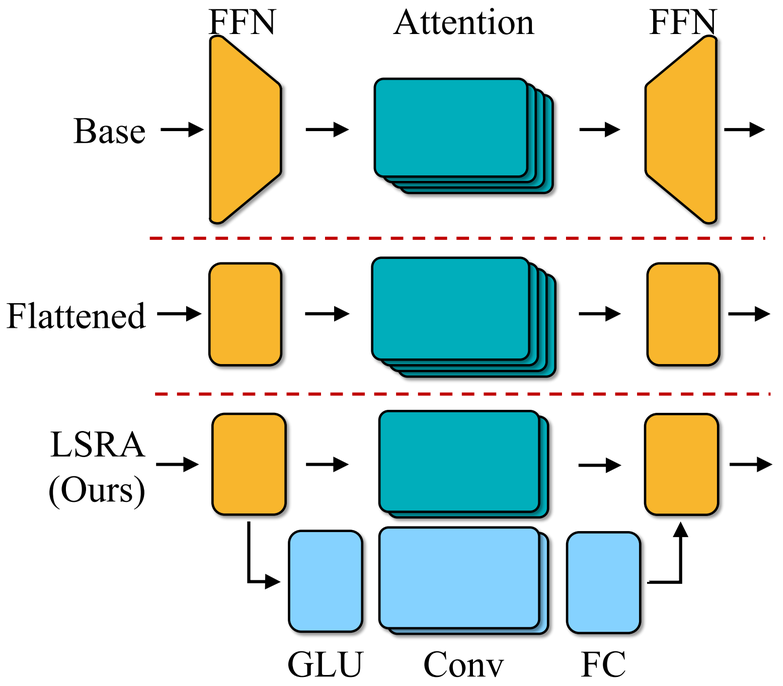
- Attention is redundant which encodes too much local information.
- Half of the dimension is used for attention (global) and the other half for convolution (local).



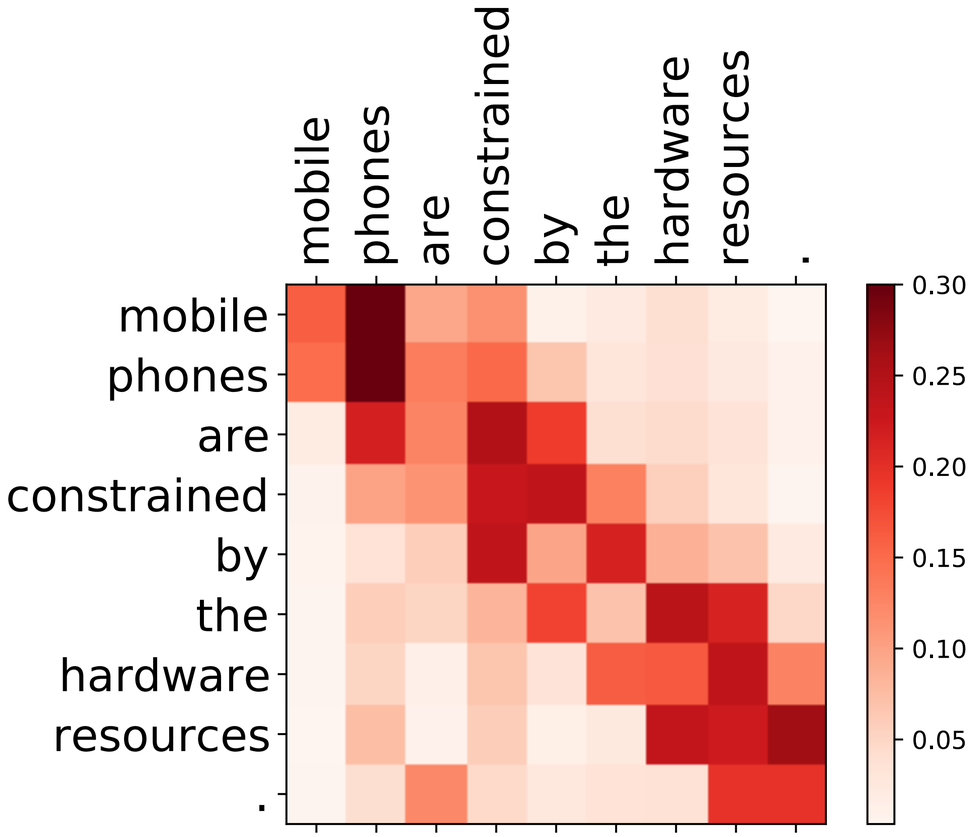
(a) Lite Transformer block

Flattened FFN

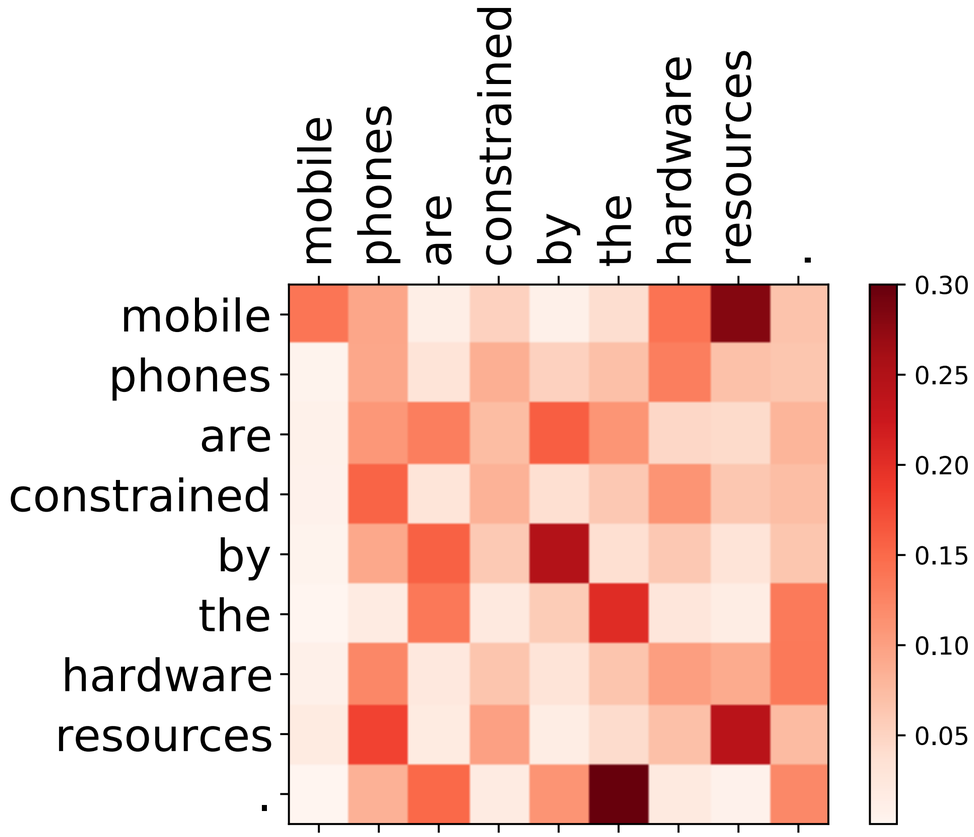
- When sequence length is short (e.g. in NMT), FFN bottleneck is needless (attention $\propto N^2$ and FFN $\propto N$).
- FFN is not used for encoding context information, so large hidden dimension is meaningless (in doubt).
- The proportion of FFN calculations is higher than attention, so flattening FFN is beneficial to optimizing attention.



Visualization of Attention



(a) Conventional Attention.



(b) Attention in LSRA.

Performance

	#Parameters	#Mult-Adds	WMT'14 En-De BLEU	WMT'14 En-De ΔBLEU	WMT'14 En-Fr BLEU	WMT'14 En-Fr ΔBLEU
Transformer (Vaswani et al., 2017)	2.8M	87M	21.3	–	33.6	–
Lite Transformer (Ours)	2.9M	90M	**22.5**	**+1.2**	**35.3**	**+1.7**
Transformer (Vaswani et al., 2017)	11.1M	338M	25.1	–	37.6	–
Lite Transformer (Ours)	11.7M	360M	**25.6**	**+0.5**	**39.1**	**+1.5**
Transformer (Vaswani et al., 2017)	17.3M	527M	26.1	–	38.4	–
Lite Transformer (Ours)	17.3M	527M	**26.5**	**+0.4**	**39.6**	**+1.2**

Table 2: Results on WMT'14 En-De and WMT'14 En-Fr. Our Lite Transformer improves the BLEU score over the transformer under similar Mult-Adds constraints.

- Lite-Transformer can achieve better performance with the same parameters and calculations.
- Convolution is faster than attention.
- However, in our implementation, Lite-Transformer is the same as Transformer-big in WMT-14 (maybe because they use gradient accumulate).
- Totally convolutional seq-to-seq model may be better.

Inspirations (Attention & FFN)

- Most attention heads are useless, which can be removed or replaced by fixed patterns.
- FFN is important, but can be reduced by increasing other blocks' parameters.
- Deeper layers need more parameters, so we can distinguish different layers.
- Enc-Dec attention is important, so we'd better preserve it.
- Goodbye attention. Hello convolution.

Cross-Layer Parameter Sharing

- Share the parameters of all encoders and decoders.
- Simple but effective, reducing the parameters from 258m to 110m (81m embedding and 29m layers) with little loss on performance.
- It may be better if we only share unimportant blocks (e.g. transformations for attention)?

LayerDrop

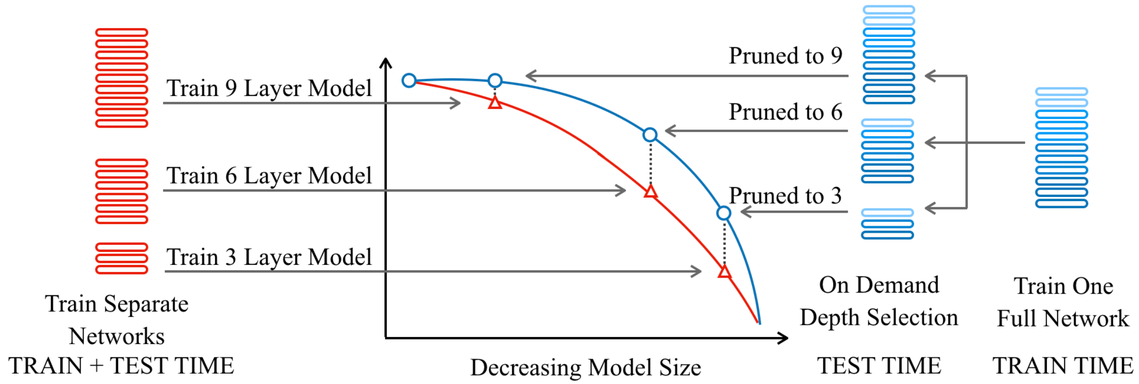


Figure 1: **LayerDrop** (right) randomly drops layers at training time. At test time, this allows for sub-network selection to any desired depth as the network has been trained to be robust to pruning. In contrast to standard approaches that must re-train a new model from scratch for each model size (left), our method trains only one network from which multiple shallow models can be extracted.

- Training: drop layers with probability p.
- Testing: three methods for selecting the network.
  - Drop every $1/p$ layers (chosen).
  - Select best network using development dataset (complex).
  - Using p_d as activations after each layer in training (best).

Performance on NMT (as Regularizer)

Model	Enc Layers	Dec Layers	BLEU
Transformer (Vaswani et al., 2017)	6	6	28.4
Transformer (Ott et al., 2018)	6	6	29.3
DynamicConv (Wu et al., 2019)	7	6	29.7
Transformer (Ott et al., 2018) + LayerDrop	6	6	29.6
Transformer (Ott et al., 2018) + LayerDrop	12	6	**30.2**

Table 1: **Results on WMT en-de Machine Translation** (newstest2014 test set)

- LayerDrop can boost the performance without increasing the parameters and calculations.

Performance (as Pruning)

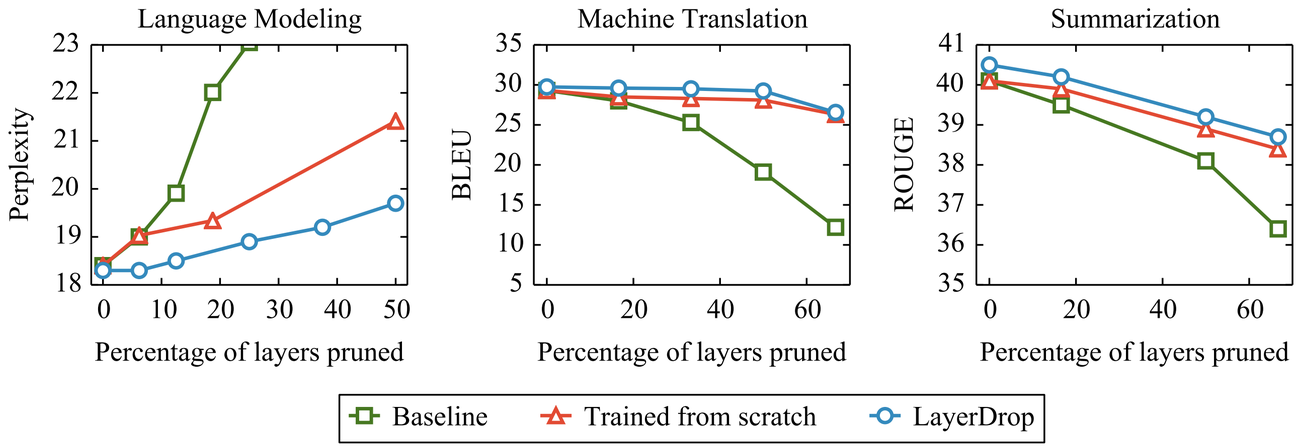


Figure 2: **Performance as a function of Pruning** on various generation tasks (test set), compared to training smaller models from scratch and pruning a Transformer baseline trained without LayerDrop. Pruning networks with LayerDrop performs strongly compared to these alternatives.

- No need to fine-tune.
- Only need to train once for pruning models with different sizes.
- On MT task, LayerDrop can reduce parameters by half with no loss.

Ablations

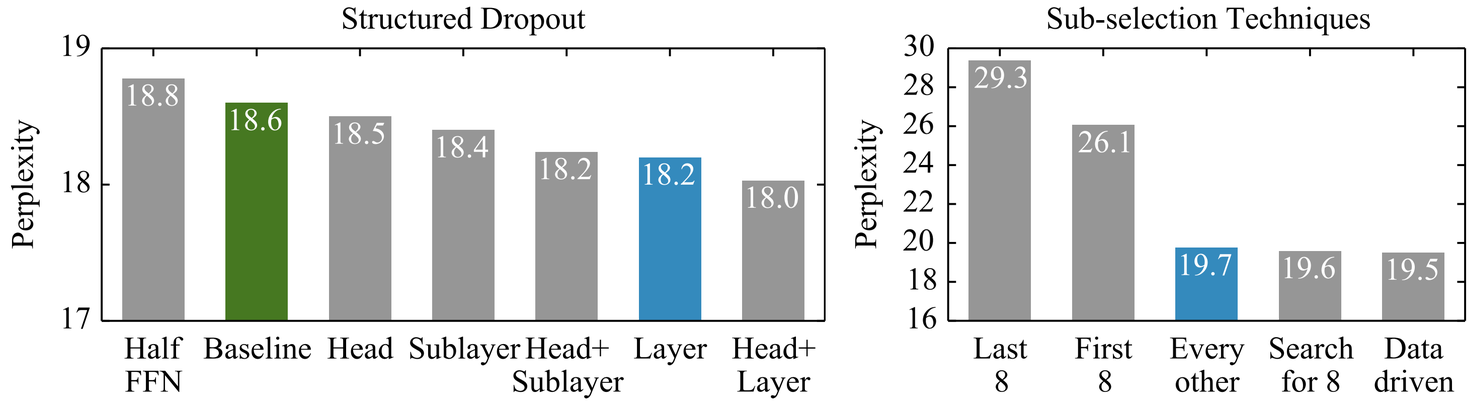
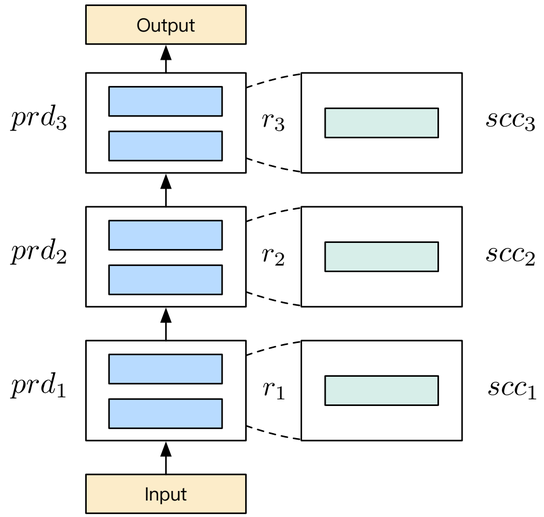


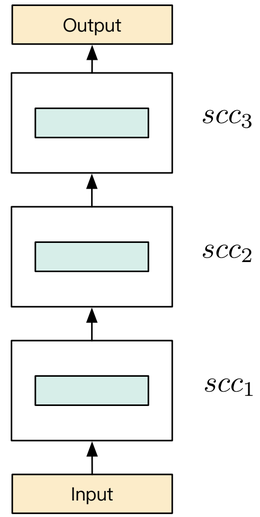
Figure 4: (left) **Impact of Various Structured Dropouts** on Wikitext-103 Valid. Dropping Layers is straightforward and has strong performance. (right) **Comparison of Pruning Strategies** on Wikitext-103 Valid. Marginal gains can be achieved, but dropping every other layer is hard to beat.

- Dropping sublayers should be better than dropping layers, but here not.
- Dropping layers is simple and effective.
- Selecting continuous layers in inference is harmful.

Bert-of-Theseus



(a) Compression Training



(b) Successor Fine-tuning and Inference

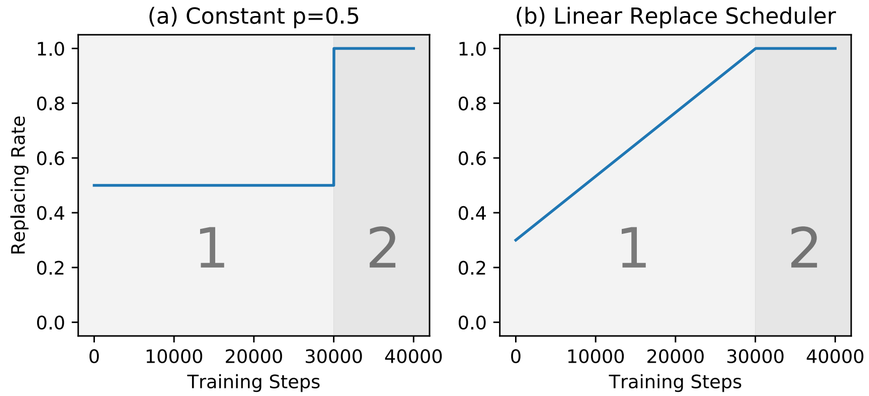
- Training: replace each predecessor with a successor with probability p.

$$y_{i+1} = r_{i+1} * scc_i(\mathbf{y}_i) + (1 - r_{i+1}) * prd_i(\mathbf{y}_i)$$

- Fine-tune and inference: use all successors.

[arXiv] Bert-of-theseus: Compressing bert by progressive module replacing

Curriculum Replacement



$$p_d = \min(1, \theta(t)) = \min(1, kt + b)$$

$$lr' = (np_d/n)lr = (kt + b)lr$$

Figure 2: The replacing curves of a constant module replace rate and a replacement scheduler. We use different shades of gray to mark the two phases of Theseus Compression: (1) Module replacing. (2) Successor fine-tuning.

- Constant replacement rate is enough.
- Linear replacement rate can help smooth the learning process and combine training/fine-tune process.
- Warm-up of learning rate can also help training.

[arXiv] Bert-of-theseus: Compressing bert by progressive module replacing

Experiments

Table 2: Experimental results on the dev set of GLUE. The numbers under each dataset indicate the number of training samples.

Method	CoLA (8.5K)	MNLI (393K)	MRPC (3.7K)	QNLI (105K)	QQP (364K)	RTE (2.5K)	SST-2 (67K)	STS-B (5.7K)	Macro Score
Teacher / Predecessor Model									
BERT-base [4]	54.3	83.5	89.5	91.2	89.8	71.1	91.5	88.9	82.5
Models Compressed with External Data									
DistillBERT [28]	43.6	79.0	87.5	85.3	84.9	59.9	90.7	81.2	76.5
PD-BERT [37]	-	**83.0**	87.2	89.0	89.1	66.7	91.1	-	-
Models Compressed without External Data									
Fine-tuning	43.4	80.1	86.0	86.9	87.8	62.1	89.6	81.9	77.2
Vanilla KD [14]	45.1	80.1	86.2	88.0	88.1	64.9	90.5	84.9	78.5
BERT-PKD [33]	45.5	81.3	85.7	88.4	88.4	66.5	91.3	86.2	79.2
BERT-of-Theseus	**51.1**	82.3	**89.0**	**89.5**	**89.6**	**68.2**	**91.5**	**88.7**	**81.2**

- Bert-of-Theseus beats most KD methods.
- Compared with KD methods, Bert-of-Theseus is better.
  - No need for additional loss.
  - Model-agnostic (general method besides Transformer).
  - Closer connection between predecessors and successors.

Inspirations (Layer-wise)

- Layer-wise compression methods are simple to implement compared to other model-specific methods (e.g. KD).
- We need to carefully design the training strategies to help the models learn smoothly.
- Fine-grained drop methods may be better.
- There's no need for layer-wise methods to design the detailed model structure, and you can just decide to whether to drop some modules in existing models.

Conclusions

- Embedding and FFN are important, attention heads are not.
- Deeper, wider and more complex embedding networks are better and would not increase parameters if caching final outputs.
- Parameters of attention heads and FFN can be transferred to other modules (e.g. embedding) for performance.
- Different layers can be shared or pruned without performance loss.

Our Attempt

- Cross-layer sharing and embedding decomposition (≥256d) has little impact.
- Vocabulary reduction has a great impact?
- FFN and model dimension reduction has a great impact.
- Layer numbers impact for compressing models are less than Transformer-big.

Model		Bleu	Param
Transformer-big		30.69	258
ls+ed(128)+dp01		29.42	40
ls+ed(256)+dp01	/	30.48	50
	emb share	29.91	40
	emb(6k)	27.42	33
	ffn(2048)	29.57	42
	12e3d	30.33	50
	12e6d+dm(768)+head(12)	30.09	41
ls+hierarchical emb		<25	30

What's Next

- Improve the structure of Transformer.
- Try other faster and smaller structures (LSTM, CNN, etc) and retrofit them to increase the performance.
- Network architecture search (NAS).
- …

Reference

- [ICLR20] ALBERT: A Lite BERT for Self-supervised Learning of Language Representations
- [ICLR20] DeFINE: Deep Factorized Input Token Embeddings for Neural Sequence Modeling
- [arXiv] DeLighT: Very Deep and Light-weight Transformer
- [NIPS19] Are Sixteen Heads Really Better than One?
- [arXiv] Fixed Encoder Self-Attention Patterns in Transformer-Based Machine Translation
- [ICLR20] Lite Transformer with Long-Short Range Attention
- [ICLR19] Reducing Transformer Depth on Demand with Structured Dropout
- [arXiv] Bert-of-theseus: Compressing bert by progressive module replacing

Q & A

Thanks