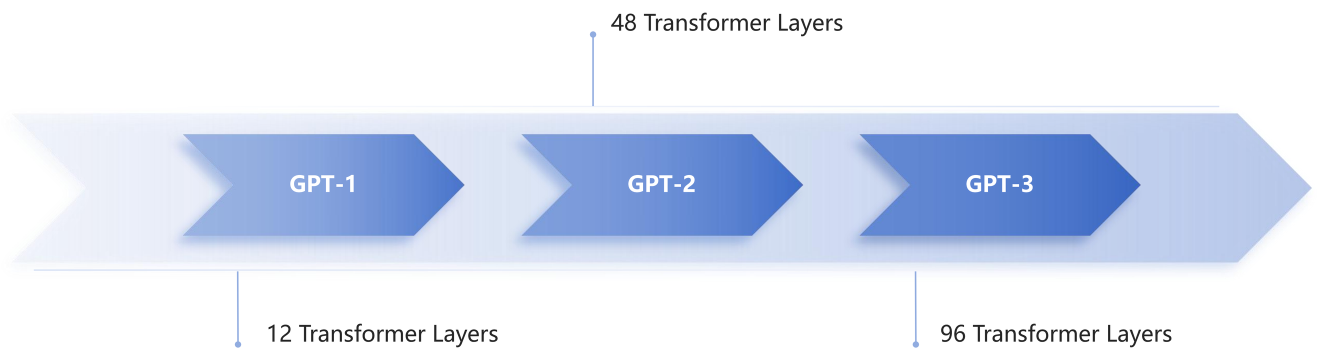
From Static to Dynamic: A Deeper, Faster, and Adaptive Language Modeling Approach

ARCIDS-LAB 档案智能开发与服务重点实验室

Jiajia Li, Qiwei Li and Ping Wang*

Motivation

A key embodiment of the depth language model's reasoning and understanding ability for language is the depth of the model.



Inference:

- Although the deeper the performance is better in overall performance, it is questionable whether each input instance requires the maximum number of layers.
- A short and simple sentence is obviously less difficult to encode than a complex or long sentence.
- It will risk overfitting if we use the same very deep structure for these sentences.
- The use of more structures than required for the encoding brings unnecessary computation overhead, thus increasing the service latency.

Architecture

(1) Vanilla Transformer Layer

For any i-th layer with input H_{i-1}, the encoding process for output H_i can be formalized as:

$$\tilde{H}_i = H_{i-1} + \textsc{MhAttn}(\textsc{Ln}(H_{i-1}))$$
$$H_i = \tilde{H}_i + \textsc{Ffn}(\textsc{Ln}(\tilde{H}_i))$$

The computation of one-head in the multi-head attention of the vanilla Transformer can be represented as follows:

$$\textbf{AS}(X) = \frac{X^T \mathbf{W}_Q^T \mathbf{W}_K X}{\sqrt{d_{head}}}$$
$$\textsc{Attn}(X) = \textsc{SoftMax}(\textbf{AS}(X))\mathbf{W}_V X$$

Specifically, we calculate two additional positional scores:

$$\textbf{RKS}(X) = \mathbf{W}_K X \times \mathbf{E}_{R_{i-j}}, \textbf{RQS}(X) = \mathbf{W}_Q X \times \mathbf{E}_{R_{i-j}}$$

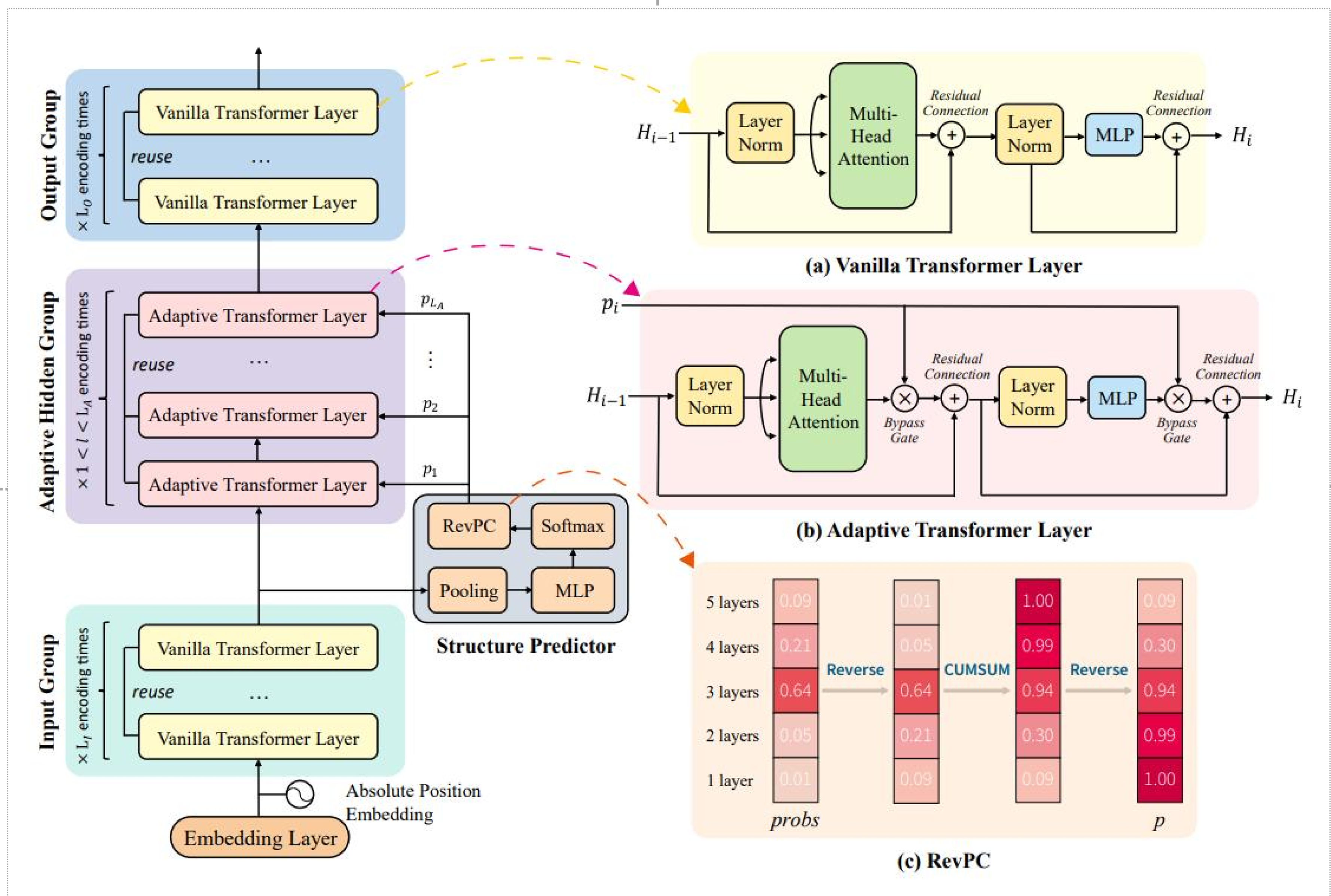
The new attention score:

$$\textbf{AS}^{rel}(X) = \textbf{AS}(X) + \textbf{RKS}(X) + \textbf{RQS}(X)$$

(2) Adaptive Transformer Layer

Challenge of changing the model structure by directly changing the number of layers during the training phase:

- *The operation of changing the model structure is not differentiable and cannot be directly optimized by the overall loss of the model.*
- *The training uses the mini-batch mechanism to take advantage of the parallel computing power of GPU devices and for the purpose of more stable optimization, however the model structure required for different examples within the mini-batch is not the same, which makes the input-dependent model structure unfeasible.*



The encoding process of adaptive Transformer is changed from the vanilla Transformer:

$$\tilde{H}_i = H_{i-1} + \textsc{MhAttn}(\textsc{Ln}(H_{i-1}))$$
$$H_i = \tilde{H}_i + \textsc{Ffn}(\textsc{Ln}(\tilde{H}_i))$$

to:

$$\tilde{H}_i = H_{i-1} + p_i \cdot \textsc{MhAttn}(\textsc{Ln}(H_{i-1}))$$
$$H_i = \tilde{H}_i + p_i \cdot \textsc{Ffn}(\textsc{Ln}(\tilde{H}_i))$$

(3) Structure Predictor

Perform first-token pooling on the outputs to obtain its sentence representation:

$$S = \textsc{Pooling}(H_{\mathrm{IG}}).$$

Predict the number of layers in the hidden group it needs to use:

$$probs = \textsc{SoftMax}(\textbf{MLP}(S))$$

RevPC:

1) Reverse the probabilities.

2) Perform a cumulative sum of the elements.

3) Reverses the result back.

Advantage:

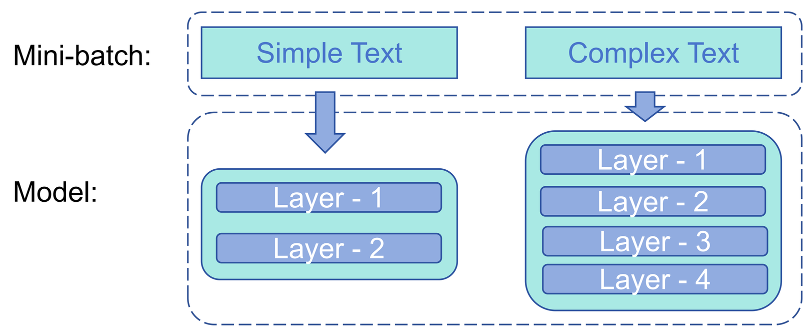
- *The complexity estimation and model structure changes are optimized by the loss of the language model during pre-training, achieving self-learning of the input complexity without additional complexity learning process.*
- *The model can continue to update the complexity estimation according to the downstream finetuning in conjunction with the specific task, thus providing more flexibility*

Process:

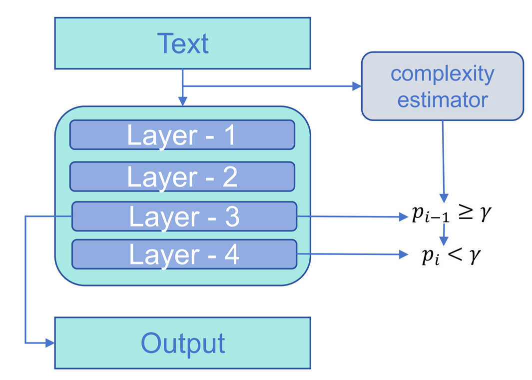
- *During the training stage, it is possible to use mini-batch strategy to encode different examples simultaneously while each example actually uses a different model structure.*
- *In the inference stage, early exiting mechanism is directly performed according to the structure predicted by the complexity estimator on the input to achieve the purpose of efficient inference.*

(4) Early Exiting

The training stage:



The inference stage:



Experiments

Main Results

(1) Development Results on GLU NLU Benchmark

Model	Param.	MNLI	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	Avg.	Speed
ALBERT-base	12M	84.6	89.6	91.7	92.8	58.9	89.5	89.5	78.6	84.4	(1.00×)
LayerDrop	12M	79.8	87.3	87.0	90.7	53.6	86.5	85.9	74.3	80.6	(1.96×)
HeadPrune	12M	80.3	88.0	86.8	90.5	54.1	87.4	86.2	75.1	81.1	(1.22×)
BranchyNet	12M	81.7	87.4	88.9	91.6	55.2	–	87.2	75.4	–	(1.88×)
Shallow-Deep	12M	81.5	87.8	89.2	91.7	55.5	–	87.1	75.2	–	(1.95×)
PABEE	12M	85.1	89.6	**91.8**	**93.0**	61.2	90.1	**90.0**	80.1	85.1	(1.57×)
ALBERTa-base	30M	**85.2**	**90.5**	91.6	**93.0**	**61.4**	**90.8**	89.9	**80.3**	**85.3**	(1.68×)

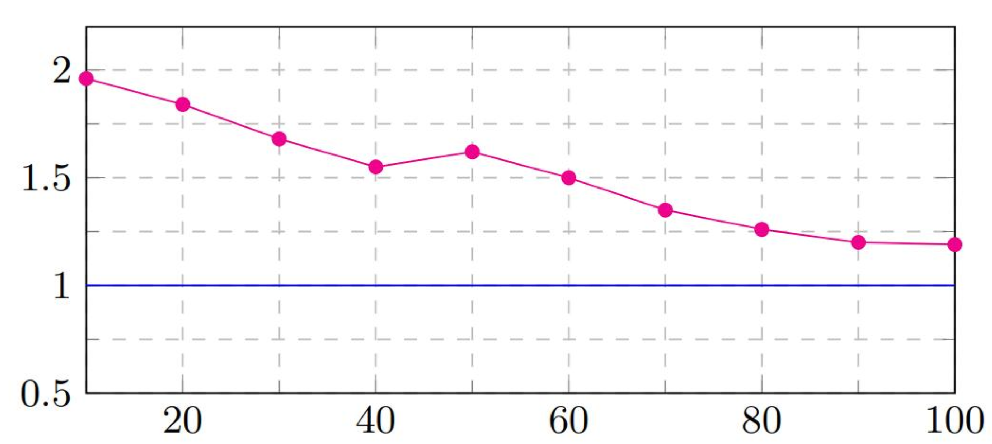
(2) Development Results on SQUAD v2.0 MRC Benchmark

Model	Param.	EM	F1	Speedup
ALBERT-base	13M	77.1	80.0	(1.00×)
ALBERTa-base	30M	79.3	82.6	(1.46×)
ALBERT-large	19M	79.4	82.3	(1.00×)
ALBERTa-large	48M	82.2	85.3	(1.50×)
ALBERT-xlarge	62M	83.1	86.1	(1.00×)
ALBERTa-xlarge	166M	84.2	87.3	(1.43×)
ALBERT-xxlarge	237M	85.1	88.1	(1.00×)
ALBERTa-xxlarge	630M	86.7	89.7	(1.55×)

(3) Development Results on CoNLL-2003 NER Benchmark

Model	P	R	F1	Speed
ALBERT-base	93.69	94.10	93.90	(1.00×)
ALBERTa-base	94.15	94.27	**94.21**	(1.81×)
ALBERT-large	94.15	94.66	94.41	(1.00×)
ALBERTa-large	94.56	94.61	**94.59**	(1.72×)
ALBERT-xlarge	94.88	94.19	94.53	(1.00×)
ALBERTa-xlarge	94.53	95.22	**94.87**	(1.70×)
ALBERT-xxlarge	95.06	95.64	95.35	(1.00×)
ALBERTa-xxlarge	95.26	95.91	**95.58**	(1.66×)

The Consumed Time Statistics



Speedup vs. sequence lengths on sampled sentences.

Ablation Study

Model	MNLI	QQP	QNLI	NER	Speed
ALBERT-base	84.6	89.6	91.7	93.90	(1.00×)
ALBERTa-base	85.2	90.5	91.6	94.21	(1.77×)
w/o EarlyExit	85.5	90.8	92.0	94.48	(0.28×)
$\gamma = 0.1$	85.2	90.7	92.0	94.37	(0.52×)
$\gamma = 0.5$	85.3	90.4	91.8	94.33	(1.22×)
$\gamma = 0.9$	80.4	87.5	88.2	91.23	(1.98×)
w/o RevPC	84.1	89.8	90.9	90.35	(1.71×)
w/o AbsPos	85.1	90.4	91.8	94.20	(1.78×)
w/o RelPos	84.9	90.0	91.5	94.05	(1.87×)