Supplementary Material: Implementation and Experiments for GAU-based Model

Zhenjie Liu
School of Computer Science and Technology, Xidian University
santosocy18@gmail.com

Abstract

In February this year Google proposed a new Transformer variant called FLASH(Hua et al., 2022), which has a faster speed, lower VRAM footprint and better performance. This is achieved by designing a performant layer named GAU (Gated Attention Unit), which combines the Attention layer and FFN. In this paper, some implementation details are re-analyzed both theoretically and practically. We then propose a novel GAU-based model and pre-train it model on a Chinese corpus. Results of the CLUE benchmark show that our model achieves a dev average score of 73.95, 1% higher than RoFormerV1 and being 45% faster, which is also competitive with RoFormerV2.

1 Introduction

These days have witnessed the great success of pre-trained Transformer-based models (Vaswani et al., 2017) (Devlin et al., 2018) (Radford et al., 2019). The self-attention mechanism is the key defining characteristic of Transformer models, however, it's also blamed for it's quadratic time and memory complexity, which can hinder model scalability especially in processing long sequences. There has been a lot of variants proposed to address this problem by modifying the model architecture, and most of these methods fall into two categories: *"sparsification"* and *"linearization"*. The former (Child et al., 2019) (Beltagy et al., 2020) (Zaheer et al., 2020) introduces sparsity into attention matrix by limiting the view of each token to reduce the computation of token-to-token associativity. Further more, it separates the input sequence into several chunks so that the computation of attention matrix takes place in every single chunk rather than the complete sequence. To reduce the performance loss caused by reducing the view of each token, some models propose the global attention to capture the long-term dependences. But obviously, this approach has two drawbacks: 1. How to choose the area of attention to be retained is highly subjective. 2. It requires specific design optimization in programming and therefore, it's hard to generalize.

Another kind of models start by using the associative law of matrix multiplication to theoretically approximate the softmax function in the attention matrix. Some works (Choromanski et al., 2020) (Kasai et al., 2021) (Qin et al., 2022) design effective unbiased estimation of the original softmax with linear space and time complexity. Another models (Wang et al., 2020) construct the approximate matrix to reduce the complexity by utilizing the low rank property of attention matrix.

Not long ago Google proposed a new model architecture to address the quality and empirical speed issues of existing Transformer variants. This is achieved by combining the Attention layer and FFN into a single unit called GAU while reducing it to just one head (Hua et al., 2022). However, it is flawed in many details such as the scaling factor and the replacement of softmax. In this work, we analyze several questionable details both theoretically and practically and reorganize the model architecture. In addition, we pre-train the new model on a Chinese corpus and compare it with several classical models on the CLUE benchmark (Xu et al., 2020). Results show that the proposed GAU-based model achieves a dev average score of 75.02, 1% higher than RoFormerV1 and being 45% faster. We also compared GAU and RoFormerV2 (Su et al., 2022) which both use the same hyper-parameters and pre-train for the same number of steps. The comparison results show the former is slightly higher, which indicates that GAU is not inferior to RoFormerV2.

To summarize, our contributions include:

- Many details of the original paper such as the scaling factor and the replacement of softmax are analyzed both theoretically and empirically.

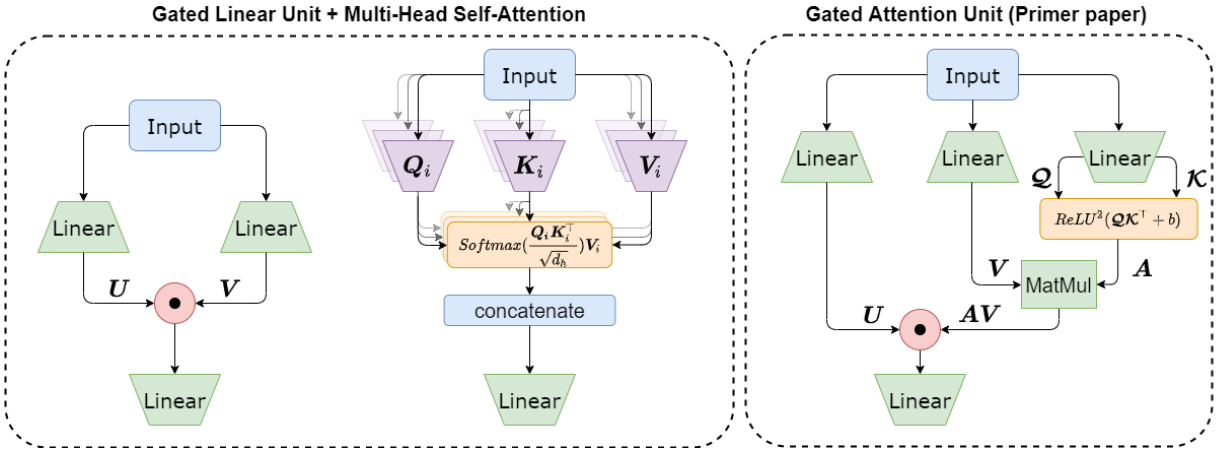

Figure 1: (Hua et al., 2022) (1) The Gated Linear Unit (GLU) and the Multi-Head Self-Attention (MHSA) unit, (2) The Gated Attention Unit (GAU) proposed in the origial paper. Ignore scaling factors and normalization in (1), (2) for brevity.

- We reconstruct the model architecture and pre-train it on a Chinese corpus. In addition, we also compared the fitting ability of GAU and RoFormerV2 on 9 Chinese datasets.

2 Background and related work

2.1 Backbone Network: Transformer

Given a sequence of input tokens $\{s_i\}_{i=1}^n$, the vector representations $\boldsymbol{X} \in \mathbb{R}^{n \times d_h}$ are computed via summing the word or token embedding, position and segment embedding.

Multi-Head Self-Attention. In each layer, Transformer (Vaswani et al., 2017) uses multi-head self-attention to aggregate the output vectors of the previous layer and to encode contextual information for input tokens. The operation for a single head is defined as:

$$\boldsymbol{A}_i = softmax(\frac{\boldsymbol{Q}_i \boldsymbol{K}_i^\top}{\sqrt{d_h}})\boldsymbol{V}_i$$

where $\boldsymbol{Q}_i = \boldsymbol{X}\boldsymbol{W}_q$, $\boldsymbol{K}_i = \boldsymbol{X}\boldsymbol{W}_k$, $\boldsymbol{V}_i = \boldsymbol{X}\boldsymbol{W}_v$ are obtained by applying linear transformations on the temporal dimension of the input sequence. $\boldsymbol{W}_q$, $\boldsymbol{W}_k$, $\boldsymbol{W}_v \in \mathbb{R}^{d_h \times \frac{d_h}{H}}$ are the weight matrices (learnable parameters).

Vanilla FFN. The output for Transformer's FFN can be formulated as follows:

$$\boldsymbol{X}^* = \phi(\boldsymbol{X}_A \boldsymbol{W}_u^\top)\boldsymbol{W}_o$$
$$\boldsymbol{O} = \text{LayerNorm}(\boldsymbol{X}_A + \boldsymbol{X}^*)$$

where $\boldsymbol{W}_u^\top$, $\boldsymbol{W}_o \in \mathbb{R}^{d_{ff} \times d_h}$. Here d_h denotes the hidden size of the model, d_{ff} denotes the intermediate size, and ϕ is an element-wise activation function.

These two layers carry different functions: the Self-Attention layer is responsible for capturing the relationship between tokens, and the FFN layer can enhance the nonlinearity of the model.

2.2 GLU

The Gated Linear Unit (Shazeer and Noam, 2020) is an improved MLP variant augmented with gating. It has been proven effective in many cases (Narang et al., 2021) and is used in many state-of-the-art models. (Du et al., 2021)

$$\boldsymbol{U} = \phi_u(\boldsymbol{X}\boldsymbol{W}_u),\ \boldsymbol{V} = \phi_v(\boldsymbol{X}\boldsymbol{W}_v) \quad \in \mathbb{R}^{n \times d_{ff}}$$
$$\boldsymbol{O} = (\boldsymbol{U} \odot \boldsymbol{V})\boldsymbol{W}_o \quad \in \mathbb{R}^{n \times d_h}$$

where $\phi_u = \phi_v =$ Swish (Elfwing et al., 2017) (Hendrycks and Gimpel, 2016), and $\odot$ stands for the element-wise multiplication (Hadamard product). In GLU, each representation u_i is gated by another linear representation v_i from the same token.

2.3 GAU proposed by the original paper

Since the GLU is more efficient FFN, we use it as the basis for modification. Notice that GLU cannot replace the Self-Attention because it lacks token-to-token interactions. To alleviate this, a natural idea is to multiply the gate value matrix $\boldsymbol{V}$ by attention matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$:

$$\boldsymbol{O} = (\boldsymbol{U} \odot \boldsymbol{A}\boldsymbol{V})\boldsymbol{W}_o$$

Unlike the vanilla Transformer, the attention matrix here is calculated using the following formula:

$$\boldsymbol{Z} = \phi_z(\boldsymbol{X}\boldsymbol{W}_z) \qquad \in \mathbb{R}^{n\times s}$$
$$\boldsymbol{A} = \text{ReLU}^2\left(\frac{\mathcal{Q}(\boldsymbol{Z})\mathcal{K}(\boldsymbol{Z})^\top}{n} + b\right) \qquad \in \mathbb{R}^{n\times n}$$

where $\boldsymbol{Z}$ is a shared linear representation of the input $\boldsymbol{X}$, $s = 128$, $\mathcal{Q}$ and $\mathcal{K}$ are two simple affine transformations that apply per-dim scalars and offsets to $\boldsymbol{Z}$, and b is the relative position bias.

Compared to the standard Scaled-Dot Self-Attention, the attention matrix here is still has the complexity of $O(n^2)$. However, it has been improved in many ways to improve computational efficiency and reduce the number of parameters. The biggest change is to replace the softmax function with squared ReLU which is obtained by NAS method (So et al., 2021). This replacement also shows that the softmax in the attention mechanism is not necessary and can be replaced by a regular activation function with a simple normalization method. Another significant improvement is that the GAU uses only one head, which greatly enhances computing efficiency and reduces VRAM usage.

The Transformer's MHSA comes with $4d_h^2$ parameters, and the FFN layer has $2d_h \times d_{ff}$ parameters. In standard Transformer-based models like BERT, the d_{ff} is set to be $4d_h$. However, the GAU block here only has $3d_h \times d_{ff}$ parameters. ($\boldsymbol{W}_z$, scalars and offsets in Q and K are negligible) By setting $d_{ff} = 2d_h$ for GAU, this compact design allows us to replace each Transformer Encoder block (MHSA + FFN) with two GAU layers while retaining similar training speed and fitting ability.

2.4 RoPE

Rotary Position Embedding (RoPE) (Su et al., 2021) encodes absolute position information with rotation matrix. Compared to sinusoidal position embedding proposed in Transformer, the later is additive, while the former can be considered multiplicative.

We can add absolute position information to $\boldsymbol{q}$ and $\boldsymbol{k}$ which are the linear representations of the input using the following formula:

$$\tilde{\boldsymbol{q}}_m = \boldsymbol{f}(\boldsymbol{q}, m) = \begin{pmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \\ \vdots \\ q_{d-2} \\ q_{d-1} \end{pmatrix} \odot \begin{pmatrix} \cos m\theta_0 \\ \cos m\theta_0 \\ \cos m\theta_1 \\ \cos m\theta_1 \\ \vdots \\ \cos m\theta_{d/2-1} \\ \cos m\theta_{d/2-1} \end{pmatrix} + \begin{pmatrix} -q_1 \\ q_0 \\ -q_3 \\ q_2 \\ \vdots \\ -q_{d-1} \\ q_{d-2} \end{pmatrix} \odot \begin{pmatrix} \sin m\theta_0 \\ \sin m\theta_0 \\ \sin m\theta_1 \\ \sin m\theta_1 \\ \vdots \\ \sin m\theta_{d/2-1} \\ \sin m\theta_{d/2-1} \end{pmatrix}$$

Thus, after the inner product operation, the result attention matrix will carries relative position information.

Scaling Factors	MLM Acc
n^2	20.16%
n	19.34%
$n \cdot s$	**23.11%**
s^2	misconvergence

Table 1: MLM results of various scaling factors.

3 Details

In this section we will discuss several details in the original paper. We first analyse whether to use dropout in section 3.1, we then investigate which scaling factor is best in 3.2. And finally, we compare squared relu function with softmax in terms of the fitting and the generalization ability in section 3.3.

3.1 Dropout

In the original paper, there is no discussion of which part of the model should have dropout added. And furthermore, the dropout rate is set to 0 in appendix section B. Considering that the original FLASH model was pre-trained and tested in processing long sequences, overfitting is not a major constraint on it. For the sake of computational efficiency, dropout was removed. While the maximum sequence length of our model is set to 512, we thus introduce dropout in the linear and attention layers like other prevalent models.

Table 2: MLM accuracy and CLUE dev average score of different activation functions and training strategies.

Model	Training Strategy[4]		MLM Acc	CLUE	Average Pre-training Time
	MLM	Fine-tune			
$\frac{1}{ns}\text{ReLU}^2(\frac{1}{\sqrt{d_h}})$	512	$diff$	25.23%	42.37	07:32:34
	$diff$		24.01%	50.88	
$\frac{1}{c_i \cdot ns}\text{ReLU}^2(\frac{1}{\sqrt{d_h}})$[1]	512	$diff$	31.97%	55.41	08:22:51
	$diff$		30.18%	58.94	
$softmax(\frac{1}{\sqrt{d_h}})$[2]	512	$diff$	40.86%	62.8	**07:11:13**
	$diff$		**41.2%**	65.49	
$softmax(\frac{\log_{512} n}{\sqrt{d_h}})$[3]	512	$diff$	41.07%	64.23	07:31:22
	$diff$		40.62%	**66.52**	

1. The scaled squared ReLU function. 2. The standard softmax in attention mechanism.
3. A softmax variant proposed by (Su, 2021) which improves generalization ability. Hereinafter called $softmax\_plus$.
4. Different training strategies in pre-training and fine-tuning phase. "*diff*" represents that we feed the model with data of different length, and "*512*" means that the input sequence length is fixed at 512. Results are obtained by pre-training for 10k steps.

3.2 Scaling Factor

According to the reference code in the appendix of the original paper, the scaling factor without bias is $\frac{1}{n^2}$ after simplification, which is smaller than the standard Transformer of $\frac{1}{n}$.

We train the model under diffenrent scaling factors for 5k steps and report the result in Table 1, from which we can see that n^2 is not an optimal choice and the model will perform better by replacing it with $n \cdot s$.

3.3 Squared ReLU or Softmax

It's well known that two most important capabilities of a NLP model is the fitting ability and the generalization ability. And we will compare squared ReLU and softmax in these two respects to determine whether using the Squared ReLU function is the best choice.

Generalization ability. We enhance the fitting ability by pre-training on the large-scale unsupervised corpus. However, for prevalent NLP models like BERT and GPT, we do not design specific tasks to improve the generalization ability.

Let's take the input sequence length for a fairly deep dive. In the pre-training phase, the value of n is nearly identical to the max sequence length. Then we directly use the pre-trained model for various tasks as if the model can automatically generalize to different input seqence lengths. However, results in the Table 2 show that when we replace the softmax function with squared ReLU, results of downstream tasks with variable-length inputs turn out badly. This means that compared with softmax, the squared ReLU function is much worse in the generalization ability over the input sequence length.

In the softmax formula, the only thing associated with the length n is the scaling factor c_i:

$$a_{i,j} = \frac{1}{c_i} \exp\left(\frac{\boldsymbol{q}_i \cdot \boldsymbol{k}_j}{\sqrt{d_h}}\right), c_i = \sum_{i=1}^{n} \exp\left(\frac{\boldsymbol{q}_i \cdot \boldsymbol{k}_j}{\sqrt{d_h}}\right)$$

To investigate the effect of scaling factor on generalization ability, we conducted some experiments. Table 2 shows that the scaled squared ReLU:

$$a_{i,j} = \frac{1}{c_i \cdot ns}\text{ReLU}^2\left(\frac{\boldsymbol{q}_i \cdot \boldsymbol{k}_j}{\sqrt{d_h}}\right)$$
$$c_i = \sum_{i=1}^{n} \text{ReLU}^2\left(\frac{\boldsymbol{q}_i \cdot \boldsymbol{k}_j}{\sqrt{d_h}}\right)$$

has a good performance on length generalization.

Besides adding a normalization coefficient to the original function, (Su, 2021) proposed a novel softmax variant in terms of entropy invariance which is expressed as:

$$\boldsymbol{A} = softmax\left(\frac{\log_{512} n}{\sqrt{d_h}}\boldsymbol{Q}\boldsymbol{K}^\top\right)V$$

a_{ij} can be considered as a conditional distribution of j given i. The entropy of it can be calculated as:

$$h_i = -\sum_{j=1}^{n} a_{i,j} \log a_{i,j}$$

Assume that h_i represents the degree of focus of the i[th] token on each token. Specially, if $\boldsymbol{a}_i$ is an uniform distribution, it's entropy is:

$$a_{i,j} = \frac{1}{n}, h_i = -\sum_{i=1}^{n} a_{i,j} \log a_{i,j} = \log n$$

Matrix	Rank/max_len	Sparsity
$\boldsymbol{QK}^\top$	0.25	≈ 0
$softmax(\frac{\boldsymbol{QK}^\top}{\sqrt{d_h}})$	0.9983	10.03%
$\text{ReLU}^2(\frac{\boldsymbol{QK}^\top}{\sqrt{d_h}})$	0.989	12.49%

Table 3: Ratio of rank to max input length (512) of different attention matrices.

And if $h_i \propto n$, then $\log h_i \propto \log n$. To some extent, it indicates that the attention mechanism here degenerates into an uniform distribution, which cannot fully capture the token-to-token interactions. Therefore, h_i should be length-insensitive.

Intuitively speaking, we want the distribution of attention of the i[th] token remains the same rather than being disturbed after the introduction of new tokens. (Su, 2021) analyzed this problem theoretically and proposed a new variant called $softmax\_plus$. And results in Table 2 show that it's beneficial to improve the generalization over the input sequence length by adding a scaling factor inside the softmax function.

From the analysis above, we have two solutions with admirable generalization performance:

- Scaled Squared ReLU function;
- softmax_plus function.

However, It is worth noting that in terms of the computational efficiency, the later runs faster than the former. And thus the softmax_plus function might be a better choice.

Fitting ability. The fitting ability of an NLP model largely depends on its capability to capture the interactions between tokens, which can be mathematically represented by the rank of it's attention matrix.

Since the attention matrix without softmax is obtained by multiplying two low-rank matrices $\boldsymbol{Q}, \boldsymbol{K} \in \mathbb{R}^{n\times s}$, we have $R(\boldsymbol{QK}^\top) \leq s$, where $s = 128$. Due to it's low rank property, we use softmax to obtain a high-rank matrix. As shown in Table 3, the attention matrix after softmax is close to be full rank. However, after applying the relu function, the rank of the attention matrix is not as high as applying softmax, which indicates that it carries less information. This is probably because the matrix is more sparse after applying the relu function. Since the relu function replaces negative values with zero, many relationships between tokens are lost.

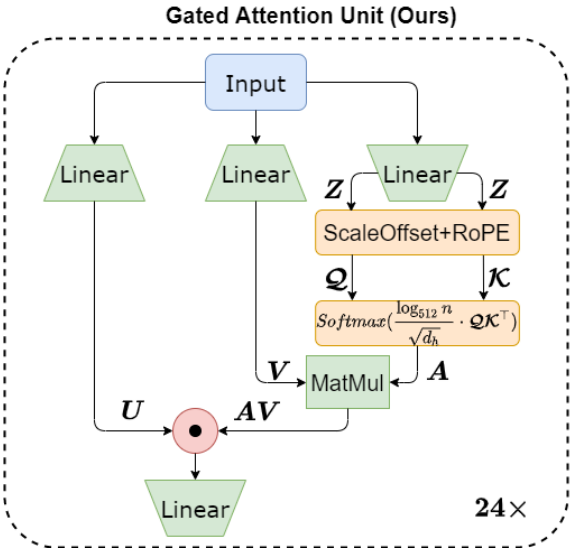

Figure 2: Our proposed GAU-based model.

So in terms of the fitting ability of the model, using the softmax fuction could be a better choice than replacing it with squared relu. And according all factors aforementioned, we use softmax_plus in our reorganized model architecture.

4 Methodology

Combining the analysis in Section 3 with some conclusions from the original paper, we propose a novel GAU-based model. Model architecture and several pre-training details are showcased in section 4.1 and 4.2.

4.1 Model architecture

As shown in Figure 2, our model is built upon 24-layers GAU. Let $\boldsymbol{X} \in \mathbb{R}^{n\times d_h}$ be the representations over n tokens, and $\boldsymbol{Z} \in \mathbb{R}^{n\times s}$ is the shared linear representation. The attention matrix can be calculated as:

$$\boldsymbol{\mathcal{Q}} = w_Q \boldsymbol{Z} + b_Q + RoPE \qquad \in \mathbb{R}^{n\times s}$$
$$\boldsymbol{A} = softmax(\frac{log_{512}n}{\sqrt{d_h}}\boldsymbol{\mathcal{Q}}\boldsymbol{\mathcal{K}}^\top) \quad \in \mathbb{R}^{n\times n}$$

We apply per-dim scaling and offset to $\boldsymbol{Z}$ which is very cheap. Additionally, we add RoPE to $\boldsymbol{\mathcal{Q}}$ to further enhance the generalization ability of the model. (Su et al., 2021)

It is good to note that we apply a *post-norm* (Wang et al., 2022) to GAU layer output:

$$hidden\_states^{l+1} = Norm(hidden\_states^{l} + \boldsymbol{O})$$

Table 4: Comparison of VRAM footprint and speedup between proposed model and baselines in pre-training phase.

Models	Input Sequence Length								Params
	256		512		1024		2048		
	VRAM (M)[1]	Time Cost[2]	VRAM (M)[1]	Time Cost[2]	VRAM (M)[1]	Time Cost[2]	VRAM (M)[1]	Time Cost[2]	
RoFormerV1	6125 (105.95%)	1.487×	10549 (124.6%)	1.45×	25059 (169.1%)	1.582×	OOM[3]	-	124,148k (128.63%)
RoFormerV2	5643 (97.61%)	1.12×	10047 (118.7%)	1.224×	19631 (132.5%)	1.471×	45277 (153.3%)	1.902×	94,777k (98.2%)
GAU (ours)	5781 (100%)	1×	8463 (100%)	1×	14819 (100%)	1×	29533 (100%)	1×	96,519k (100%)

1. batch size = 8. 2. Measured based on time cost of pre-training for 1k steps. Using a single T40 GPU.
3. OOM stands for CUDA out of memory.

Table 5: Best averaged results on the evalulation datasets of CLUE.

Model	AFQMC	CSL	IFLYTEK	TNEWS	WSC	CMNLI	CMRC	CHID	C3	AVG
RoFormerV1	74.21	83.13	60.17	58.07	83.22	81.5	74.31	**86.21**	65.27	74.01
RoFormerV2	**75.96**	**84.81**	**63.24**	**59.39**	**83.93**	81.41	**79.35**	85.63	**74.32**	**76.45**
GAU (ours)	74.51	83.7	62.72	57.93	82.89	**81.97**	78.04	85.49	67.98	75.02
RoFormerV2*	**70.66**	79.13	**59.07**	55.38	63.82	77.26	**70.25**	77.08	53.86	67.39
GAU (ours)*	69.14	**79.6**	58.36	**56.57**	**64.11**	**77.47**	68.86	**78.2**	**56.15**	**67.61**

* Pre-trained for 30k steps using a MLM-only approach.

we replace the layer normalization with $Norm$ function where:

$$Norm(\boldsymbol{X}) = \frac{\boldsymbol{X}}{\sqrt{VAR(\boldsymbol{X}) + \epsilon}}$$

According to the analysis of (Su, 2022), the pre-norm structure is equivalent to increasing the width of the model and decreasing the depth, and thus has underperformed the post-norm.

4.2 Model pre-training

We use the CLUECorpusSmall (14G) from CLUE for WWM pre-training (Cui et al., 2021). See Appendix B for detailed settings.

5 Experimental results

In this section, we will verify the effectiveness (5.3) and efficiency (5.2) of proposed model and several baseline models on various CLUE tasks with detailed explanations.

5.1 Baselines

First of all, the RoFormerV1 [1] model (Su et al., 2021) is included as a standard baseline for calibration. It proposed RoPE which is used in this paper. And to demonstrate the advantages of our proposed GAU layer, we include RoFormerV2 (Su et al., 2022) as a much stronger baseline. Compared to RoFormerV1, RoFormerV2 removes all biases and replaces layer normalization with RMS-Norm, which is consistent with our model.

[1]For this experiment, we adopt code (Apache-2.0 License) from https://github.com/JunnYu/RoFormer_pytorch.

5.2 Computational efficiency

We list comparison results on computational efficiency of 3 models in Table 4, from which we can infer that the GAU-based model has a significant improvement in VRAM usage and training speed compared to baselines.

5.3 Evaluation on CLUE

As shown in Table 5, GAU achieves comparable overall performance to RoFormerV2 on various downstream tasks when fine-tuned. Importantly, it outperforms RoFormerV1 by an absolute improvement of 1%.

Since Roformerv2 was pre-trained on a larger corpus using the multi-task approach, we also compared GAU with RoformerV2 MLM-only version for fair comparison. Both of them are implemented in the same codebase with identical hyper-parameters and pre-trained for 30k steps. We table their comparison results in Table 5. The metrics show the effectiveness of our proposed GAU structure.

6 Conclusion

In this paper, the GAU architecture in the original paper is re-analyzed from several perspectives. The discussion of the scaling factors illustrates that replacing n^2 with $n \cdot s$ helps to improve the model performance. Besides, we compare ReLU^2 and softmax both theoretically and empirically. The results show that softmax has a significant advantage over ReLU^2 in both length generalization ability

and fitting ability. We then present a novel model based on GAU. Experiments on whole word mask language modeling task shows that it is as good as RoFormerV2. Finally, experiment on nine tasks from CLUE demonstrate the superior performance of our model when applied to downstream tasks.

7 Discussion and Future work

The success of GAU depends largely on GLU's efficiency. GLU explicitly introduces control over the flow of information, which is equivalent to having a priori information. However, there is a lack of rigorous theoretical proof for the superiority of GLU over standard FFN. A future work is to investigate this and whether GLU can replace FFN in other scenarios. In addition, we will also pre-train our model on a larger corpus and apply it on a wider range of NLP tasks.

References

Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. 2019. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. 2020. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*.

Yiming Cui, Wanxiang Che, Ting Liu, Bing Qin, and Ziqing Yang. 2021. Pre-training with whole word masking for chinese bert. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:3504–3514.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Nan Du, Yanping Huang, Andrew M Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, et al. 2021. Glam: Efficient scaling of language models with mixture-of-experts. *arXiv preprint arXiv:2112.06905*.

Stefan Elfwing, Eiji Uchibe, and Kenji Doya. 2017. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning.

Dan Hendrycks and Kevin Gimpel. 2016. Gaussian error linear units (gelus).

Weizhe Hua, Zihang Dai, Hanxiao Liu, and Quoc V Le. 2022. Transformer quality in linear time. *arXiv preprint arXiv:2202.10447*.

Jungo Kasai, Hao Peng, Yizhe Zhang, Dani Yogatama, Gabriel Ilharco, Nikolaos Pappas, Yi Mao, Weizhu Chen, and Noah A Smith. 2021. Finetuning pretrained transformers into rnns. *arXiv preprint arXiv:2103.13076*.

Sharan Narang, Hyung Won Chung, Yi Tay, William Fedus, Thibault Fevry, Michael Matena, Karishma Malkan, Noah Fiedel, Noam Shazeer, Zhenzhong Lan, et al. 2021. Do transformer modifications transfer across implementations and applications? *arXiv preprint arXiv:2102.11972*.

Zhen Qin, Weixuan Sun, Hui Deng, Dongxu Li, Yunshen Wei, Baohong Lv, Junjie Yan, Lingpeng Kong, and Yiran Zhong. 2022. cosformer: Rethinking softmax in attention. *arXiv preprint arXiv:2202.08791*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Shazeer and Noam. 2020. Glu variants improve transformer.

David R So, Wojciech Mańke, Hanxiao Liu, Zihang Dai, Noam Shazeer, and Quoc V Le. 2021. Primer: Searching for efficient transformers for language modeling. *arXiv preprint arXiv:2109.08668*.

Jianlin Su. 2021. Reanalyze the scale operation in attention mechanism in terms of entropy invariance.

Jianlin Su. 2022. Why post-norm performs better than pre-norm?

Jianlin Su, Yu Lu, Shengfeng Pan, Bo Wen, and Yunfeng Liu. 2021. Roformer: Enhanced transformer with rotary position embedding. *arXiv preprint arXiv:2104.09864*.

Jianlin Su, Yu Lu, Shengfeng Pan, Bo Wen, and Yunfeng Liu. 2022. Roformerv2: A faster and better roformer - zhuiyiai. Technical report.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Hongyu Wang, Shuming Ma, Li Dong, Shaohan Huang, Dongdong Zhang, and Furu Wei. 2022. Deepnet: Scaling transformers to 1,000 layers.

Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. 2020. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*.

Liang Xu, Hai Hu, Xuanwei Zhang, Lu Li, Chenjie Cao, Yudong Li, Yechen Xu, Kai Sun, Dian Yu, Cong Yu, Yin Tian, Qianqian Dong, Weitang Liu, Bo Shi, Yiming Cui, Junyi Li, Jun Zeng, Rongzhao Wang, Weijian Xie, Yanting Li, Yina Patterson, Zuoyu Tian, Yiwen Zhang, He Zhou, Shaoweihua Liu, Zhe Zhao, Qipeng Zhao, Cong Yue, Xinrui Zhang, Zhengliang Yang, Kyle Richardson, and Zhenzhong Lan. 2020. CLUE: A Chinese language understanding evaluation benchmark. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 4762–4772, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. 2020. Big bird: Transformers for longer sequences. *Advances in Neural Information Processing Systems*, 33:17283–17297.

A CLUE Benchmark

We choose 6 classification datasets (AFQMC, CMNLI, CSL, IFLYTEK, TNews, WSC) and 3 machine reading comprehension datasets (CMRC2018, CHID, C3) which are all from CLUE Benchmark (Xu et al., 2020). Hyperparameters for tasks above are listed in Table 6.

Table 6: Hyperparameters for CLUE datasets.

	AFQMC	**CSL**	**IFLYTEK**	**TNEWS**	**WSC**	**CMNLI**	**CMRC***	**CHID**	**C3**
Sequence length	512								
Batch size	16	16	32	16	16	32	32	32	24
Epochs	3	4	5	3	40	3	3	3	8
Peak learning rate	3e-5	2e-5	3e-5	2e-5	1e-5	3e-5	3e-5	3e-5	2e-5
Warmup proportion	0.1								
Learning rate decay	Linear								
Optimizer	AdamW								
Adam ϵ	1e-8								
Adam (β_1, β_2)	(0.9, 0.999)								
Weight decay	0.01								
Hidden dropout	0.1								
Attention dropout	0.1								
Classifier dropout	0.1								

* We use the average score of F1-score and EM for CMRC dataset and accuracy for the remaining as the evaluation metrics.

B Hyperparameters for MLM pre-training

Hyperparameters for the MLM task on CLUECorpusSmall are listed in Table 7.

Table 7: Hyperparameters for the MLM task on CLUECorpusSmall

	MLM Results
Data	CLUECorpusSmall (14G)
Sequence length	512 or $diff$
Batch size	64
Gradient accumulation steps	4
Number of steps	5k, 10k, 30k, 100k
Warmup proportion	0.1
Peak learning rate	3e-4
Learning rate decay	Linear
Optimizer	AdamW
Adam ϵ	1e-6
Adam (β_1, β_2)	(0.9, 0.999)
Weight decay	0.01
Hidden dropout	0.1
Attention dropout	0.1
Classifier dropout	0.1