GLU Variants Improve Transformer

Noam Shazeer
Google
noam@google.com

February 14, 2020

Abstract

Gated Linear Units [Dauphin et al., 2016] consist of the component-wise product of two linear projections, one of which is first passed through a sigmoid function. Variations on GLU are possible, using different nonlinear (or even linear) functions in place of sigmoid. We test these variants in the feed-forward sublayers of the Transformer [Vaswani et al., 2017] sequence-to-sequence model, and find that some of them yield quality improvements over the typically-used ReLU or GELU activations.

GLU는 Linear + Non-Linear로 구성(Sigmoid)
Non-Linear를 다른 함수로도 바꿔보자

GLU를 튜닝해보니
기존에 사용하던 ReLU난 GELU보다도 성능이 좋았다

1 Introduction

The Transformer [Vaswani et al., 2017] sequence-to-sequence model alternates between multi-head attention, and what it calls "position-wise feed-forward networks" (FFN). The FFN takes a vector x (the hidden representation at a particular position in the sequence) and passes it through two learned linear transformations, (represented by the matrices W_1 and W_2 and bias vectors b_1 and b_2). A rectified-linear (ReLU) [Glorot et al., 2011] activation function applied between the two linear transformations.

기존 Transformer에서는
Attention Block 사이에 FFN이 들어감
이 때, ACTFN은 ReLU임

$$\mathrm{FFN}(x, W_1, W_2, b_1, b_2) = \max(0, xW_1 + b_1)W_2 + b_2 \tag{1}$$

Following the T5 codebase [Raffel et al., 2019] [1], we use a version with no bias:

T5같은 경우 ReLU에서 Bias를 제외시킴

$$\mathrm{FFN}_{\mathrm{ReLU}}(x, W_1, W_2) = \max(xW_1, 0)W_2 \tag{2}$$

Subsequent work has proposed replacing the ReLU with other nonlinear activation functions such as Gaussian Error Linear Units, $\mathrm{GELU}(x) = x\Phi(x)$ [Hendrycks and Gimpel, 2016], and $\mathrm{Swish}_\beta(x) = x\sigma(\beta x)$ [Ramachandran et al., 2017].

기존 GLU의 Non-Linear함수를
Sigmoid에서 GELU나 Swish로 바꿔본다

$$\begin{aligned}\mathrm{FFN}_{\mathrm{GELU}}(x, W_1, W_2) &= \mathrm{GELU}(xW_1)W_2 \\ \mathrm{FFN}_{\mathrm{Swish}}(x, W_1, W_2) &= \mathrm{Swish}_1(xW_1)W_2\end{aligned} \tag{3}$$

2 Gated Linear Units (GLU) and Variants

[Dauphin et al., 2016] introduced Gated Linear Units (GLU), a neural network layer defined as the component-wise product of two linear transformations of the input, one of which is sigmoid-activated. They also suggest omitting the activation, which they call a "bilinear" layer and attribute to [Mnih and Hinton, 2007].

$$\begin{aligned}\mathrm{GLU}(x, W, V, b, c) &= \sigma(xW + b) \otimes (xV + c) \\ \mathrm{Bilinear}(x, W, V, b, c) &= (xW + b) \otimes (xV + c)\end{aligned} \tag{4}$$

We can also define GLU variants using other activation functions:

[1] Also in the interest of ML fairness.

$$\begin{aligned}
\mathrm{ReGLU}(x, W, V, b, c) &= \max(0, xW + b) \otimes (xV + c) \\
\mathrm{GEGLU}(x, W, V, b, c) &= \mathrm{GELU}(xW + b) \otimes (xV + c) \\
\mathrm{SwiGLU}(x, W, V, b, c, \beta) &= \mathrm{Swish}_\beta(xW + b) \otimes (xV + c)
\end{aligned} \tag{5}$$

In this paper, we propose additional variations on the Transformer FFN layer which use GLU or one of its variants in place of the first linear transformation and the activation function. Again, we omit the bias terms.

$$\begin{aligned}
\mathrm{FFN}_{\mathrm{GLU}}(x, W, V, W_2) &= (\sigma(xW) \otimes xV)W_2 \quad \text{Sigmoid} \\
\mathrm{FFN}_{\mathrm{Bilinear}}(x, W, V, W_2) &= (xW \otimes xV)W_2 \quad \text{Bi-Linear} \\
\mathrm{FFN}_{\mathrm{ReGLU}}(x, W, V, W_2) &= (\max(0, xW) \otimes xV)W_2 \quad \text{ReLU} \\
\mathrm{FFN}_{\mathrm{GEGLU}}(x, W, V, W_2) &= (\mathrm{GELU}(xW) \otimes xV)W_2 \quad \text{GELU} \\
\mathrm{FFN}_{\mathrm{SwiGLU}}(x, W, V, W_2) &= (\mathrm{Swish}_1(xW) \otimes xV)W_2 \quad \text{Swish}
\end{aligned} \tag{6}$$

All of these layers have three weight matrices, as opposed to two for the original FFN. To keep the number of parameters and the amount of computation constant, we reduce the number of hidden units d_{ff} (the second dimension of W and V and the first dimension of W_2) by a factor of $\frac{2}{3}$ when comparing these layers to the original two-matrix version.

3 Experiments on Text-to-Text Transfer Transformer (T5)

We test the FFN variants we have described on the transfer-learning setup from [Raffel et al., 2019]. An encoder-decoder transformer model [Vaswani et al., 2017] is trained on a denoising objective of predicting missing text segments, and subsequently fine-tuned on various language understanding tasks.

1) Encoder-Decoder 트랜스포머 모델에서 Masking을 예측하는 Pre-train
2) Task에 맞는 Finetuning
두 학습과정에서 실험해볼 것임

3.1 Model Architecture

Baseline 모델은 T5

We use the same code base, model architecture, and training task as the base model from [Raffel et al., 2019]. The encoder and decoder each consist of 12 layers, with $d_{model} = 768$. For the attention layers, $h = 12$ and $d_k = d_v = 64$. The FFN layers have hidden size $d_{ff} = 3072$. As we describe above, for the GLU-variant-based FFN layers, which have thee weight matrices instead of two, we reduce the hidden layer to $d_{ff} = 2048$, so as to maintain the same parameter and operation counts as the base model.

Table 1: Heldout-set log-perplexity for Transformer models on the segment-filling task from [Raffel et al., 2019]. All models are matched for parameters and computation.

Log-Perplexity로 측정

Training Steps	65,536	524,288
$\mathrm{FFN}_{\mathrm{ReLU}}(baseline)$	1.997 (0.005)	1.677
$\mathrm{FFN}_{\mathrm{GELU}}$	1.983 (0.005)	1.679
$\mathrm{FFN}_{\mathrm{Swish}}$	1.994 (0.003)	1.683
$\mathrm{FFN}_{\mathrm{GLU}}$	1.982 (0.006)	1.663
$\mathrm{FFN}_{\mathrm{Bilinear}}$	1.960 (0.005)	1.648
$\mathrm{FFN}_{\mathrm{GEGLU}}$	**1.942** (0.004)	**1.633**
$\mathrm{FFN}_{\mathrm{SwiGLU}}$	**1.944** (0.010)	**1.636**
$\mathrm{FFN}_{\mathrm{ReGLU}}$	1.953 (0.003)	1.645

3.2 Pre-Training and Perplexity Results

Identically to [Raffel et al., 2019], we pre-train for 524,288 steps on the span-filling objective on the C4 dataset. Each training batch consists of 128 examples, each of which has an input of 512 tokens and an output of 114 tokens, the output containing multiple spans of tokens which were deleted from the input[2]. Similarly to [Raffel et al., 2019], we use the Adafactor optimizer [Shazeer and Stern, 2018] and an inverse-square-root learning-rate schedule. We also decay the learning rate linearly for the final 10 percent of the training steps. Our main departure from [Raffel et al., 2019] is that we use no dropout during pre-training. We find this to produce superior results. We compute the log-perplexity on the training objective on a heldout shard of C4, which we believe to be a good indicator of model quality. For each model architecture, we also trained four models for a shorter period (65,536 steps) to measure inter-run variability. The results are listed in table 1. The GEGLU and SwiGLU variants produce the best perplexities.

3.3 Fine-Tuning

We then fine-tune each fully-trained model once on an examples-proportional mixture of the Stanford Question-Answering Dataset (SQuAD) [Rajpurkar et al., 2016] and all the language understanding tasks in the GLUE [Wang et al., 2018] and SuperGlue [Wang et al., 2019] benchmarks.[3] Fine-tuning consists of 131072 steps with a learning rate of 10^{-3}. As in training, the input sequences for each step have a combined length of approximately 65,536 tokens. Following [Raffel et al., 2019], we use a dropout rate of 0.1 on the layer outputs, feed-forward hidden-layers and attention weights. The embedding matrices are fixed during fine-tuning.

Tables 2, 3 and 4 show results on the development sets. For each task, we report the best score of any of the checkpoints recorded during fine-tuning. While the results are noisy, the new GLU-variants perform best on most of the tasks. For comparison, at the bottom of each of the tables we list the reuslts from [Raffel et al., 2019]. The model is identical to our FFN_{ReLU} model. Their results are notably worse, which we believe was caused by their use of dropout during pre-training. Also listed are the inter-run standard deviations measured by [Raffel et al., 2019].

Table 2: GLUE Language-Understanding Benchmark [Wang et al., 2018] (dev).

	Score Average	CoLA MCC	SST-2 Acc	MRPC F1	MRPC Acc	STSB PCC	STSB SCC	QQP F1	QQP Acc	MNLIm Acc	MNLImm Acc	QNLI Acc	RTE Acc
FFN_{ReLU}	83.80	51.32	94.04	**93.08**	**90.20**	89.64	89.42	89.01	91.75	85.83	86.42	92.81	80.14
FFN_{GELU}	83.86	53.48	94.04	92.81	**90.20**	89.69	89.49	88.63	91.62	85.89	86.13	92.39	80.51
$\text{FFN}_{\text{Swish}}$	83.60	49.79	93.69	92.31	89.46	89.20	88.98	88.84	91.67	85.22	85.02	92.33	81.23
FFN_{GLU}	84.20	49.16	94.27	92.39	89.46	89.46	89.35	88.79	91.62	86.36	86.18	92.92	**84.12**
$\text{FFN}_{\text{GEGLU}}$	84.12	53.65	93.92	92.68	89.71	90.26	90.13	89.11	91.85	86.15	86.17	92.81	79.42
$\text{FFN}_{\text{Bilinear}}$	83.79	51.02	**94.38**	92.28	89.46	90.06	89.84	88.95	91.69	**86.90**	**87.08**	92.92	81.95
$\text{FFN}_{\text{SwiGLU}}$	84.36	51.59	93.92	92.23	88.97	**90.32**	**90.13**	**89.14**	**91.87**	86.45	86.47	**92.93**	83.39
$\text{FFN}_{\text{ReGLU}}$	**84.67**	**56.16**	**94.38**	92.06	89.22	89.97	89.85	88.86	91.72	86.20	86.40	92.68	81.59
[Raffel et al., 2019]	83.28	53.84	92.68	92.07	88.92	88.02	87.94	88.67	91.56	84.24	84.57	90.48	76.28
ibid. stddev.	0.235	1.111	0.569	0.729	1.019	0.374	0.418	0.108	0.070	0.291	0.231	0.361	1.393

4 Conclusions

We have extended the GLU family of layers and proposed their use in Transformer. In a transfer-learning setup, the new variants seem to produce better perplexities for the de-noising objective used in pre-training, as well as better results on many downstream language-understanding tasks. These architectures are simple to implement, and have no apparent computational drawbacks. We offer no explanation as to why these architectures seem to work; we attribute their success, as all else, to divine benevolence.

[2]Each training step took approximately 0.15 seconds on a 32-core TPUv2 cluster.

[3]This departs from [Raffel et al., 2019], who fine-tuned separately on the different tasks. We chose one fine-tuning run for simplicity.

Table 3: SuperGLUE Language-Understanding Benchmark [Wang et al., 2019] (dev).

	Score Average	BoolQ Acc	CB F1	CB Acc	CoPA Acc	MultiRC F1	MultiRC EM	ReCoRD F1	ReCoRD EM	RTE Acc	WiC Acc	WSC Acc
FFN_{ReLU}	72.76	80.15	83.37	89.29	70.00	76.93	39.14	73.73	72.91	83.39	67.71	77.88
FFN_{GELU}	72.98	80.64	86.24	**91.07**	74.00	75.93	38.61	72.96	72.03	81.59	68.34	75.96
$\text{FFN}_{\text{Swish}}$	72.40	80.43	77.75	83.93	67.00	76.34	39.14	73.34	72.36	81.95	68.18	81.73
FFN_{GLU}	73.95	80.95	77.26	83.93	73.00	76.07	39.03	74.22	73.50	84.12	67.71	**87.50**
$\text{FFN}_{\text{GEGLU}}$	73.96	81.19	82.09	87.50	72.00	**77.43**	**41.03**	75.28	**74.60**	83.39	67.08	83.65
$\text{FFN}_{\text{Bilinear}}$	73.81	**81.53**	82.49	89.29	**76.00**	76.04	40.92	74.97	74.10	82.67	**69.28**	78.85
$\text{FFN}_{\text{SwiGLU}}$	**74.56**	81.19	82.39	89.29	73.00	75.56	38.72	**75.35**	74.55	**85.20**	67.24	86.54
$\text{FFN}_{\text{ReGLU}}$	73.66	80.89	**86.37**	**91.07**	67.00	75.32	40.50	75.07	74.18	84.48	67.40	79.81
[Raffel et al., 2019]	71.36	76.62	91.22	91.96	66.20	66.13	25.78	69.05	68.16	75.34	68.04	78.56
ibid. stddev.	0.416	0.365	3.237	2.560	2.741	0.716	1.011	0.370	0.379	1.228	0.850	2.029

Table 4: SQuAD [Rajpurkar et al., 2016] v1.1 (dev).

	EM	F1
FFN_{ReLU}	83.18	90.87
FFN_{GELU}	83.09	90.79
$\text{FFN}_{\text{Swish}}$	83.25	90.76
FFN_{GLU}	82.88	90.69
$\text{FFN}_{\text{GEGLU}}$	83.55	91.12
$\text{FFN}_{\text{Bilinear}}$	**83.82**	91.06
$\text{FFN}_{\text{SwiGLU}}$	83.42	91.03
$\text{FFN}_{\text{ReGLU}}$	83.53	**91.18**
[Raffel et al., 2019]	80.88	88.81
ibid. Standard Deviation	0.343	0.226

References

Yann N. Dauphin, Angela Fan, Michael Auli, and David Grangier. Language modeling with gated convolutional networks. *CoRR*, abs/1612.08083, 2016. URL http://arxiv.org/abs/1612.08083.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323, 2011.

Dan Hendrycks and Kevin Gimpel. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. *CoRR*, abs/1606.08415, 2016. URL http://arxiv.org/abs/1606.08415.

Andriy Mnih and Geoffrey Hinton. Three new graphical models for statistical language modelling. In *Proceedings of the 24th international conference on Machine learning*, pages 641–648, 2007.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv e-prints*, 2019.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.

Noam Shazeer and Mitchell Stern. Adafactor: Adaptive learning rates with sublinear memory cost. *arXiv preprint arXiv:1804.04235*, 2018.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, 2017.

Alex Wang, Amapreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. Superglue: A stickier benchmark for general-purpose language understanding systems. *arXiv preprint arXiv:1905.00537*, 2019.