NORMFORMER: IMPROVED TRANSFORMER PRETRAINING WITH EXTRA NORMALIZATION

ABSTRACT

During pretraining, the Pre-LayerNorm transformer suffers from a gradient magnitude mismatch: gradients at early layers are much larger than at later layers, while the optimal weighting of residuals is larger at earlier than at later layers. These issues can be alleviated by the addition of two normalization and two new scaling operations inside each layer. The extra operations incur negligible compute cost (+0.5% parameter increase), but improve pretraining perplexity and downstream task performance for both causal and masked language models of multiple sizes. Adding NormFormer on top of the GPT3-Medium architecture can reach the SOTA perplexity 22% faster, or converge 0.33 perplexity better in the same compute budget. This results in significantly stronger zero shot performance. For masked language modeling, NormFormer improves fine-tuned GLUE performance by 1.9% on average.

1 INTRODUCTION

The original transformer architecture (Vaswani et al., 2017) applies Layer Normalization (Ba et al., 2016) after each sublayer's residual connection ("Post-LN") in order to reduce the variance of the inputs to the following sublayer, i.e.:

$$\text{PostLN}(x) = \text{LayerNorm}(x + \text{Sublayer}(x)),$$

with

$$\text{LayerNorm}(x) = \frac{x - E[x]}{\sqrt{Var[x] + \epsilon}} \cdot \alpha + \beta,$$

where α and β are trainable parameters, and ϵ is a small constant. Recent work has observed that Post-LN transformers tend to have larger magnitude gradients in later layers compared to earlier layers (Xiong et al., 2020) and has advocated moving the LayerNorm operation to the beginning of each sublayer ("Pre-LN"; see Figure 1, left), i.e.:

$$\text{PreLN}(x) = x + \text{Sublayer}(\text{LayerNorm}(x)).$$

In practice Pre-LN transformers can be trained with larger learning rates, shorter learning rate warmup and often yield improved performance compared to Post-LN transformers (Xiong et al., 2020). Indeed, most recent large pretrained language models use Pre-LN transformers (Baevski & Auli, 2019; Radford et al., 2019; Raffel et al., 2020; Brown et al., 2020; Lieber et al., 2021).

In this work we show that, while Pre-LN improves stability over Post-LN, it also has the opposite side effect: gradients at earlier layers tend to be larger than gradients at later layers. We propose `NormFormer`, which alleviates the gradient magnitude mismatch by adding 2 extra normalization operations to each layer (see Figure 1, middle), and 2 extra sets of scaling parameters (Figure 1, right). These operations reduce gradients to early layers and increase gradients to later layers, bringing their magnitudes closer together, and overall improve the scaling of intermediate hidden states.

Compared to well-tuned Pre-LN baselines, `NormFormer` models reach target pretraining perplexities faster and achieve better pretraining perplexities and downstream task performance when controlling for compute.

The rest of this paper is organized as follows: Section 2 describes the proposed modifications, Section 3 shows pretraining and downstream task performance for fully trained `NormFormer` models

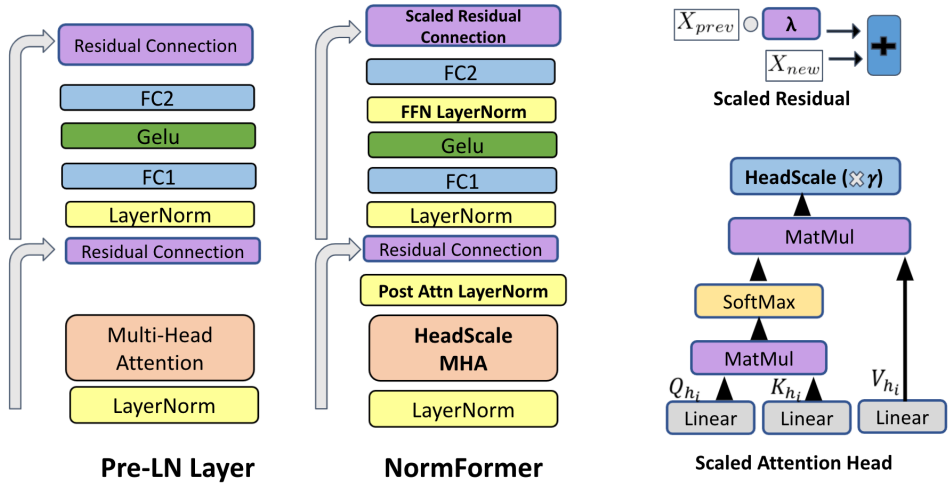

Figure 1: Left: a baseline Pre-LayerNorm transformer layer. Center: `NormFormer`, with the four proposed additions in bold. Right Top: The scaled residual connection `ResScale`, which consists of an elementwise multiplication of the previous features with trainable λ parameters. Right Bottom: a single attention head with our proposed `HeadScale` operation applied prior to the output projection with trainable parameters γ_i.

against well-tuned baselines. Section 4 analyses our proposed modifications, and shows the gradient mismatch introduced by Pre-LN and how `NormFormer` alleviates it. Section 5 shows that `NormFormer` improves over the baseline at a wide range of hyperparameter configurations, and that removing any of the added operations degrades performance.

2 Approach

2.1 NormFormer

We propose four modifications to the Pre-LN transformer and call the resulting architecture `NormFormer`.

First, we apply head-wise scaling inside the attention module (Sec. 2.1.1). Second, we add another set of scaling weights to learn an improved mixing of the inputs and residuals that are fed into the LayerNorms (Sec. 2.1.2). Finally, we add two additional LayerNorm operations: one after the attention module and a second after the first fully connected layer (Sec. 2.1.3). The four modifications introduce a small number of additional learnable parameters, which provide a cost-effective way for each layer to change the magnitude of its features, and therefore the magnitude of the gradients to subsequent components. The changes are visualized in Figure 1 and described below.

2.1.1 Scaling Attention Heads

The standard multi-head attention operation is defined as:

$$\begin{aligned}
&\text{MultiHeadAttention}(Q, K, V) = \text{Concat}(\text{h}_1, \ldots, \text{h}_n) W^O \\
&\text{h}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \\
&\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V,
\end{aligned}$$

where n is the number of heads, i is the attention head index, d_k is the dimensionality of the keys and W^O, W_i^Q, W_i^K, W_i^V are learned projection matrices for the output, query, key and value, respectively.

We propose scaling the output of each attention head via learned scalar coefficients γ_i:

$$\text{HeadScaleMHA}(Q, K, V) = \text{Concat}(\gamma_1 \text{h}_1, \ldots, \gamma_n \text{h}_n) W^O$$

where γ is initialized as a vector of 1s.

2.1.2 SCALING RESIDUALS

Standard Post-LN transformers simply sum the previous output (residual) with the new output. As we will see, the optimal mixture of these two inputs varies from layer to layer, with earlier layers typically requiring larger residual weightings. Moreover, they also vary per dimension of the hidden representation. We thus propose scaling the residual in each embedding dimension via learned scalar coefficients $(\lambda_{resid})_i$:

$$\text{ResScale(x)} = \lambda_{resid} \circ x + \text{Sublayer}(\text{LayerNorm}(x))$$

where $\circ$ is elementwise multiplication, and λ_{resid} are initialized as a vector of 1s.

While this can be applied at any normalization layer, we find it it particularly effective for normalizing the feedforward network (FFN) submodule. This can also be interpreted to a degree as alleviating the need for the FFN to produce a pertinent linear map which might be difficult depending on the non-linear activation used, and concentrating instead on nonlinear operations with its two-layer network, where the linear map is learnt using λ_{resid} instead.

2.1.3 ADDITIONAL LAYER NORMALIZATION AND PUTTING IT ALL TOGETHER

In the Pre-LN transformer each layer l modifies an input x_l as follows:

$$x_{l+1}^{\texttt{PreLN}} = \text{FFN}(\text{MHA}(x_l))$$

$$\begin{aligned} \text{where} \qquad \text{MHA}(x) &= x + \text{MultiHeadAttention}(\text{LN}(x), \text{LN}(x), \text{LN}(x)) \\ \text{FFN}(x) &= x + \sigma(\text{LN}(x) W_1 + b_1) W_2 + b_2 \\ \text{LN}(x) &= \text{LayerNorm}(x) \end{aligned}$$

In this work σ is the GELU non-linear activation introduced in Hendrycks & Gimpel (2016).

Our overall method, `NormFormer`, instead modifies each input x_l as:

$$x_{l+1}^{\texttt{NormFormer}} = \text{NormFFN}(\text{NormScaledMHA}(x_l))$$

$$\begin{aligned} \text{where} \qquad \text{NormScaledMHA}(x) &= x + \mathbf{LN}(\mathbf{HeadScaleMHA}(\text{LN}(x), \text{LN}(x), \text{LN}(x))) \\ \text{NormFFN}(x) &= \boldsymbol{\lambda_{resid}} \circ \boldsymbol{x} + \mathbf{LN}(\sigma(\text{LN}(x) W_1 + b_1)) W_2 + b_2 \end{aligned}$$

where bolded operations are newly introduced. This incorporates both additional scaling parameters introduced in the previous two subsections, and the two new normalization layers, which we subsequently refer to as **Post Attn LN** and **FFN LN** to differentiate them.

2.2 EXPERIMENTS

Causal Language Models We pretrain causal LMs (CLM) that roughly match the "Small" (125M parameter) and "Medium" (355M parameter) model sizes introduced in Brown et al. (2020). We show that the conclusions hold for larger models in Section 5.

Our model architecture differs from Brown et al. (2020) in two ways: (1) we use only dense attention, while they alternate between dense and locally banded sparse attention; (2) we train our models with sinusoidal positional embeddings, following Shortformer (Press et al., 2020b), since early experiments found this to produce comparable results with fewer learned parameters.

We train the baseline models for 300 billion tokens. We train `NormFormer` models for an equivalent number of GPU hours, which typically results in 6% fewer steps and tokens due to the additional overhead of the normalization operations.

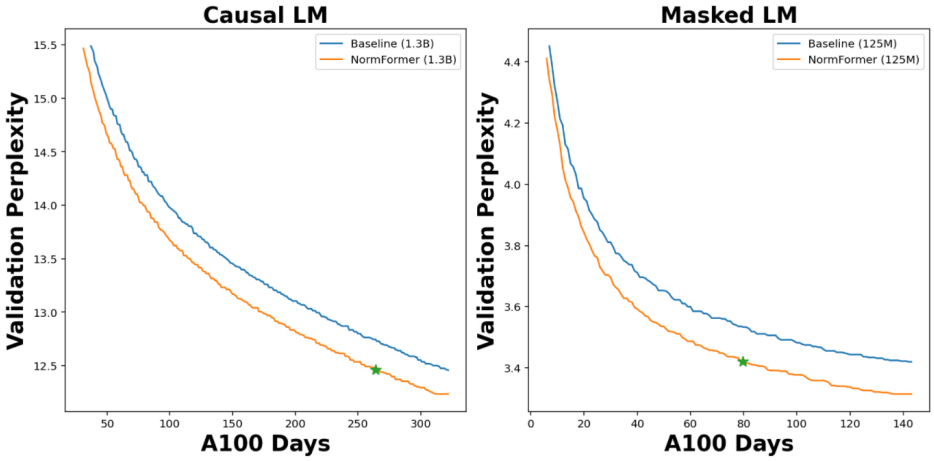

Figure 2: Pretraining perplexity on held-out validation data for Causal and Masked Language Models as a function of training compute (GPU days). The green star shows the point where NormFormer outperforms the baseline's lowest perplexity, reflecting 22% and 43% compute savings for CLM and MLM models, respectively.

Model Size	GPT-3 Paper	Baseline	NormFormer
125M	6e-4	1e-3	3e-3
355M	3e-4	1e-3	1e-3
1.3B	2e-4	6e-4	6e-4

Table 1: Searching for learning rates on our dataset results in higher values than reported in Brown et al. (2020).

We find that, on our dataset, the learning rates proposed in GPT-3 are suboptimally low.[1] For both baseline and NormFormer at each size, we selected the best learning rate from 6 50,000 step runs at $1e{-}4, 6e{-}4, 3e{-}4, 6e{-}4, 1e{-}3, 3e{-}3$. The learning rates we obtained from this process, shown in Table 1, are 3-5 times larger than those used in the GPT-3 paper. This may also explain our baseline improving over GPT-3 zero shot results in Table 2. Additionally, we have verified that the baseline and NormFormer both perform worse at the full training budget with the GPT-3 learning rates than with the higher learning rates. Other hyperparameters do not differ from GPT-3 [2]

Zero Shot Evaluation In addition to validation perplexity, we evaluate CLMs on a subset of the tasks that GPT3 evaluated on in a zero-shot setting (Brown et al., 2020), with the same prompts. We select WinoGrande (Sakaguchi et al., 2020), StoryCloze (Mostafazadeh et al., 2016), OpenBookQA (Mihaylov et al., 2018), HellaSwag (Zellers et al., 2019) and PIQA (Bisk et al., 2020) because GPT3 showed strong performance on these tasks at small scale, as well as consistently improving performance with scale.

Masked Language Models (MLM) We adopt the RoBERTa-base, Pre-LN architecture and hyperparameters used in Liu et al. (2019). For the baseline, we pretrain for 2 million batches of 1 million tokens, about $\frac{1}{4}$ of the training budget of the original `roberta-base`. NormFormer runs through 192 batches in the same amount of time.

Fine-Tuning We fine-tune both the baseline MLM and NormFormer with learning rates $1e{-}5, 1e{-}4, 3e{-}4, 1e{-}3, 3e{-}3, 6e{-}3$ and report the best performance on the validation set for

[1]The difference in optimal learning rates may be partly due to architectural differences between our baseline and GPT-3 (e.g., not using locally banded sparse attention).

[2]See Table 2.1 in Brown et al. (2020).

	Model Size	PPL	HS	PI	WG	SC	OB	Avg
Random Baseline	-	-	25.0	50.0	50.0	50.0	25.0	40.0
GPT3-125M (paper)	124.38	-	33.7	64.6	52.0	63.3	35.6	49.8
GPT3-125M (replicated)	124.38	21.11	33.7	66.5	52.2	66.1	35.4	50.8
GPT3-125M (High LR)	124.38	21.09	35.3	67.5	50.5	66.3	35.0	50.9
NormFormer-125M	124.49	**20.24**	34.9	65.9	53.4	67.5	40.0	**52.3**
GPT3-355M (paper)	354.74	-	43.6	70.2	52.1	68.5	43.2	55.5
GPT3-355M (replicated)	354.74	15.41	46.1	70.8	54.6	71.1	41.2	56.8
GPT3-355M (High LR)	354.74	14.85	48.4	71.7	53.8	73.3	43.4	58.1
NormFormer-355M	355.01	**14.52**	49.7	72.0	56.7	73.2	43.8	**59.1**
GPT3-1.3B (paper)	1313.46	-	54.7	75.1	58.0	73.4	46.8	61.6
GPT3-1.3B (replicated)	1313.46	12.56	58.5	74.6	57.9	76.8	49.4	63.4
GPT3-1.3B (High LR)	1313.46	12.52	57.5	74.3	59.3	76.3	50.8	63.6
NormFormer-1.3B	1314.05	**12.29**	60.5	74.5	60.1	77.5	50.8	**64.7**

Table 2: Zero-Shot Accuracy for Causal LMs. *GPT-3 (paper)* results taken from Brown et al. (2020). `HS`: HellaSwag, `PI`: PIQA, `WG`: WinoGrande, `SC`: StoryCloze, `OB`: OpenBookQA. PPL is validation perplexity during pretraining.

	Model Size	PPL	CoLA	MNLI	MRPC	QNLI	QQP	RTE	SST-2	Avg
Baseline	125.42M	3.42	74.3	85.9	84.6	91.6	90.7	66.4	92.9	83.8
NormFormer	125.50M	**3.32**	**82.6**	**86.3**	**86.0**	**91.9**	**91.3**	**67.9**	**93.8**	**85.7**

Table 3: Masked LM: Pretraining validation perplexity (PPL) and fine-tuned performance on GLUE tasks for Pre-LN and NormFormer models. Note that models are trained for an equal amount of compute, which is less than the publicly-released `roberta-base` models.

each GLUE task (Wang et al., 2019), following Liu et al. (2019). Other fine-tuning hyperparameters match those used for `roberta-base` in Liu et al. (2019).

Pretraining data We pretrain all models on a collection of English language text introduced in Liu et al. (2019), consisting of the BookCorpus (Zhu et al., 2019), Wikipedia and Common Crawl, as well as the English portion of the CC100 corpus (Conneau et al., 2020). We encode our data with the byte-level Byte Pair Encoding (BPE) vocabulary from Liu et al. (2019), originally introduced in Radford et al. (2019). The combined dataset contains around 450GB of uncompressed text and 110B BPE tokens. We hold out 40M BPE tokens from this data as a validation set on which we report pretraining perplexities.

Implementation details We train our causal and masked language models in `fairseq` (Ott et al., 2019). Although NormFormer introduces fewer than 0.07% additional parameters, it slows individual training updates and increases memory usage by 2-6%, primarily due to the FFN LNs. Accordingly, we compare NormFormer to baseline models trained for an equal amount of GPU time, i.e., controlling for compute rather than the number of training updates.

3 RESULTS

We report pretraining perplexities for CLMs and MLMs as a function of training wall-time (GPU days) in Figure 2. We observe that NormFormer trains significantly faster and achieves better validation perplexities for a given training compute budget. The green stars mark the first validation step where NormFormer matches the baseline's lowest perplexity and shows that NormFormer matches Pre-LN models while needing only 60% and 57% as much compute for CLM and MLM models, respectively. This is particularly impressive since NormFormer models take 2-6% longer for each training step (see Section 2.2) and thus see less data than Pre-LN models in this comparison.

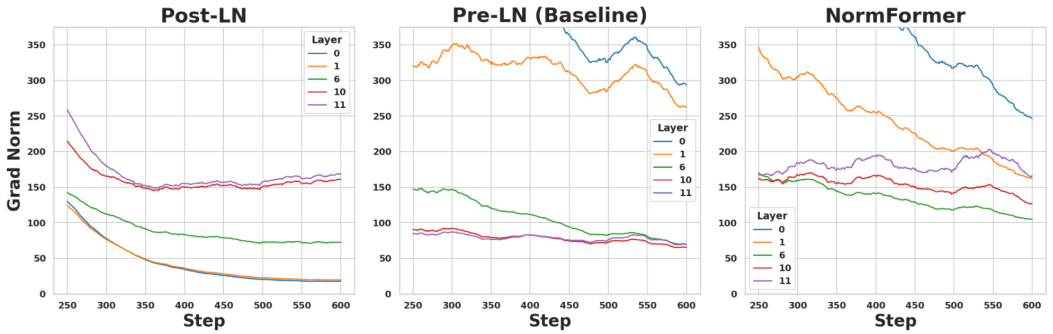

Figure 3: Average L1 norm of gradients to the second fully connected weight for layers 0,1,6,10 and 11, early in training.

We observe a similar trend on downstream tasks. In Table 2 we report zero shot accuracy for causal LMs using the tasks and prompts from Brown et al. (2020). NormFormer outperforms GPT-3 at both the 125M parameter and 355M parameter sizes. The gains from `Normformer` extra parameters operations outpace the gains from normal scaling laws. Changing the hidden dimension of a 125M parameter model from 768 to 780, for example, results in a 127 million parameter model that is only .08 perplexity better than the baseline whereas `NormFormer-125M` adds only 100,000 parameters and is 0.83 perplexity better than the baseline.

For MLM models, we report fine-tuned accuracy on GLUE in Table 3. We again find that NormFormer MLM models outperform their Pre-LN counterparts.

4 Analysis

4.1 Analysis of gradient norms by layer

We begin by examining the magnitude of the gradients at different layers for Post-LN, Pre-LN and NormFormer models, since large magnitude differences in gradients across layers can destabilize training, particularly when training in mixed precision (Micikevicius et al., 2018). Figure 3 shows the average L1 norm of the gradients to the second fully connected weight in various layers for a 12 layer, 125M parameter CLM model. As reported in past work (Xiong et al., 2020), we observe that the gradients to later layers in Post-LN models are much larger than for earlier layers, and that the gradients to early layers quickly vanish in the early stages of training. Pre-LN models have the opposite behavior, with early layers instead receiving significantly larger gradients than later layers. Finally, introducing an additional LN between fully connected layers brings the the average gradient norms closer together for different layers in the network.

4.2 Analysis of learned scaling parameters

In Figure 4 we present the distribution of scaling parameters learned by NormFormer models. For the FFN LN, the α parameters are smaller for earlier layers, reducing the magnitude of the inputs to early fully connected parameters, thereby decreasing the magnitude of their gradients. The pattern for the post attention LN, in the middle of Figure 4, is only visible at layer 0, but all layers have coefficients below 1, indicating downscaling.[3] The `HeadScale` parameters γ, shown in the right-most plot in Figure 4 vary more than the others, and have no relationship with depth in the network. We interpret this as evidence that the `HeadScale` parameters dynamically increase the importance of well initialized attention heads, as suggested in Chen et al. (2021).

One result of reducing the gradient mismatch, besides better perplexities and downstream task performance, is the ability to train stably with larger learning rates. To measure the stability of an

[3]The downscaling is more apparent in Figure 5 in the Appendix, which plots the change in grad norm for each operation at each layer. It shows that adding extra normalization reduces the gradient norm for all attention parameters at every layer and that only FFN parameters at later layers, have increased gradient norms.

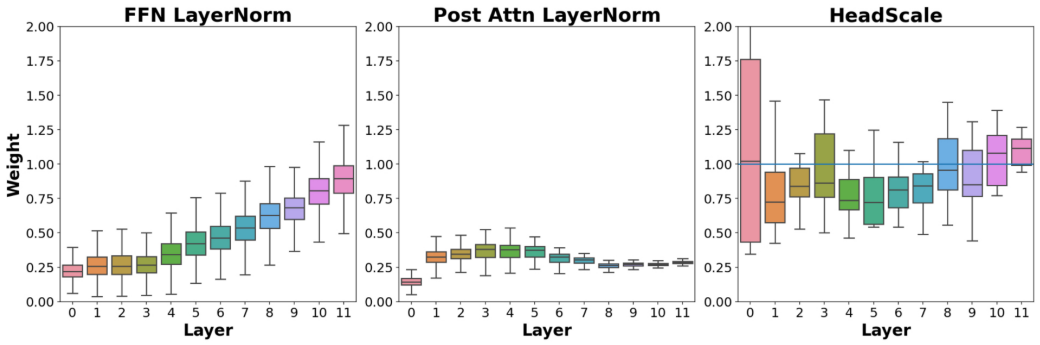

Figure 4: Distribution of learned scaling parameters in three of the added operations. For FFN LN, earlier layers receive downscaled inputs, keeping their gradients in the same range as the gradients of later layers. This plot is discussed in detail in Section 4.

	Baseline	NormFormer
Peak LR	0.02	**0.0275**
Last Step	400	**550**
Failed Operation	$q@k$	Output Projection
Failed Layer	0	12

Table 4: LR Stability Test. The learning rate starts from 0 and is linearly increased by `5e-5` at each training step until training destabilizes. NormFormer is able to reach a higher learning rate before destabilizing.

architecture, we train it on a learning rate schedule with a very large peak learning rate and very long warmup, so that the learning rate increases a little each step until the loss explodes. Table 4 shows that NormFormer models are stable at a higher peak learning rate than the baseline. For the baseline, loss explodes because the activations produced by multiplying the query and key features at layer 0 are too large to be represented in 16 bits. The down scaling of the attention outputs allows NormFormer to avoid this issue and remain stable with larger learning rates.

After training with the `ResScale` approach, one can also view the vectors that are learnt. Taking the minimum value over λ_{resid} for each layer we see the values are the smallest for the last layer (around 0.8) while being close to 1 for the first layer. According to an experiment at 125M parameters, all the benefit seems to result from the scaling attached to the final residual connection of each layer for the FFN. Adding scaling parameters for the MHA residual actually increased PPL slightly by .03. We thus only apply this approach to the FFN.

5 Ablations

This section provides evidence that removing any of our additions to the transformer block degrades performance on language modeling tasks, and that our additions improve language modeling performance across a wide range of hyperparameter settings and model sizes. Experiments use CLMs, and are run with the default hyperparameters given in Table 8 in the appendix for 470 V100 Hours (100,000 updates for the baseline) unless otherwise mentioned.

Removing any of the added operations hurts performance Table 5 shows that none of the four introduced operations can be removed without degrading performance. Rows 2-5 remove each operation one at a time. In all cases perplexity increases, with the removal of `HeadScale` being the most damaging. In Row 6 (`+ 3 More LN`) we try to introduce more normalization inside self attention, applying LN to the query, key and value features in addition to our 3 other operations, for a total of 6 new operations. In this setting, every non LN operation inside the transformer layer is followed by an LN. We find that this does not change perplexities at a fixed number of updates, but

Architecture	Valid PPL
NormFormer	**15.88**
- Post-Attn LN	15.92
- FFN LN	16.14
- Head Scale	16.22
- Res Scale	16.2
+ 3 More LN	15.88
Baseline	16.37

Table 5: Language Modeling Validation perplexities after 470 V100 Hours of pretraining. Removing any of our proposed additions degrades performance (Rows 2-5). Adding more normalization inside the Multi Headed Attention (Row 6) does not impact perplexity at a fixed number of updates, but reduces throughput such that the model can only complete 87,500 updates vs. 92,500 for Rows 1-5 and 100,000 for Row 7.

reduces training speed by another 5%. This result suggests that there is not much upside to adding even more normalization after our additions.

We see similar improvement from the addition of extra normalization as we scale model size.

Other Hyperparameter Settings Table 6 shows language modeling perplexities for 7 different hyperparameter configurations, separated by horizontal lines. `NormFormer` outperforms the baseline in all settings.

Other Experiments Replacing the FFN LN with the FFNGeGlu proposed in Shazeer (2020), which includes scaling but no normalization, degraded performance in our setting. We also find that the LN variant proposed in Raffel et al. (2020), which removes the bias and the mean substraction from the normalization, performs equally well to our LN and has fewer trainable parameters, but is about 2x slower than the `FusedLayerNorm` implementation we use. We therefore do not adopt it.

	Learning Rate	Setting Changes	Valid PPL
Baseline	0.001	-	16.80
NormFormer	0.001	-	**16.33**
Baseline	0.003	-	16.37
NormFormer	0.003	-	**15.88**
Baseline	0.006	-	16.58
NormFormer	0.006	-	**16.22**
Baseline	0.003	Longer Warmup	16.50
NormFormer	0.003	Longer Warmup	**16.06**
Baseline	0.003	GPT3	16.29
NormFormer	0.003	GPT3	**15.88**
Baseline	0.003	Clip Grad Norms at 0.1	16.46
NormFormer	0.003	Clip Grad Norms at 0.1	**16.14**

Table 6: Longer Warmup: increase LR Warmup to 6,000 steps (from 500). GPT3: increase sequence length to 2048, increase dropout to 0.1, increase training budget to 1,000 V100 hours. Grad Clip: clip gradient norms at 0.1. NormFormer outperforms the baseline in all settings.

6 RELATED WORK

Layer normalization (Ba et al., 2016) is an important component of the transformer architecture. Xiong et al. (2020) shows that for Post-LN: gradients are too big for later layers and solves this problem with Pre-LN. We build on the Pre-LN architecture to make it even more stable and efficient.

Press et al. (2020a) proposes an architecture where instead of interleaving attention and feed forward sublayers, the attention all happens first. This increases the number of late FFN parameters, rather than increasing their importance and gradient norm, as our FFN LN does, and does not impact stability. We note that there are other solutions from other domains for enforcing stability, such as in the field of reinforcement learning (Parisotto et al., 2020).

Our `HeadScale` operation is related to that used in Chen et al. (2021), but used differently. Whereas that work prunes attention heads with low γ parameters, we use the γ parameters to improve pretraining performance.

Our `ResScale` operation is related to Highway Networks (Srivastava et al., 2015), which advocate nonlinear mappings of the residual, while the original ResNet paper, with experiments on convolutional nets, also suggests a full linear map is possible (He et al., 2016). We found linear maps were unstable in our transformers, while elementwise scaling worked well.

These approaches are also related to techniques for initializing neural networks: GradInit (Zhu et al., 2021) introduces a set of scalars and biases for initialization based on a variance heuristic, and Admin (Liu et al., 2020) applies a similar heuristic in profiling and initialization stages. Similarly, some other approaches targeted initialization as well, in particular ReZero (Bachlechner et al., 2020), FixUp (Huang et al., 2020) and LookLinear (Balduzzi et al., 2017). We note that DALL-E (Ramesh et al., 2021) also added a per residual scaling factor (only during backprop). Our approach, in contrast, only has new learnable parameters without variance heuristics, and has no extra stages or changes in initialization.

7 CONCLUSION

We identify a mismatch in the gradients of Pre-LN transformer weights: earlier layers receive much larger gradients than later layers, while the optimal scaling of residuals is larger at earlier than at early layers. We propose `NormFormer`, which alleviates these issues by adding 4 extra operations to each transformer layer. These modifications help the gradient mismatch for fully connected parameters and improve validation perplexity and downstream task performance for both causal and masked language models. None can be removed without degrading performance back towards the baseline, and adding more normalization – at least of the types we have tried – improve performance. Since NormFormer primarily addresses the gradient mismatch by increasing the gradients to the last FFN layers while decreasing the gradient magnitudes in other parts of the network, future work could examine whether all 4 operations need to be added to every layer. Additionally, the small computational increase associated with NormFormer could be alleviated by fusing the FFN LN with the preceding fully connected layer, with or without the mean centering and bias, which do not appear to improve pretraining perplexity. In general, we have shown that adding small numbers of learnable parameters in the right places in our architectures can alleviate certain issues in current state of the art networks. Future work should ascertain if there are additional similarly efficient modifications that can bring gains, whilst helping us understand current deficiencies further.

8 APPENDIX

Wikitext103 Table 7 shows that NormFormer can also provide gains on top of a well tuned language model in settings with much less data. We simply add our four operations to the architecture and hyperparameters of Baevski & Auli (2019). Convergence perplexity improves, and we reach the baseline perplexity in 70% as many steps. In this setting, `NormFormer` does not improve in the last 30% of training, which suggests that with more tuning the perplexity gap could be widened.

	Steps to Final PPL	PPL
Baseline	100%	18.70
NormFormer	70%	18.65

Table 7: Wikitext 103 results following Baevski & Auli (2019). `Steps to Final PPL`: at what percentage of the 280K steps did the model reach 18.70 perplexity. `PPL`: Best Perplexity

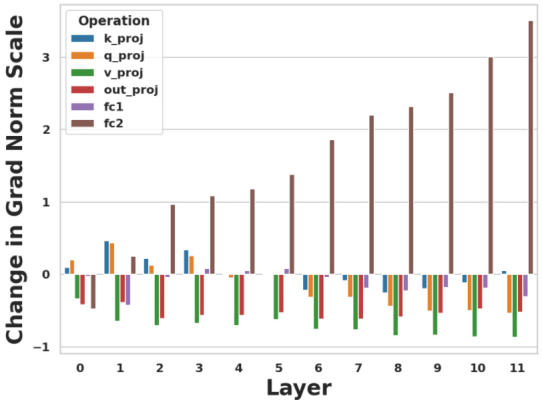

Figure 5: Change in grad scale norm for each operation of NormFormer compared to the baseline. Norms are the average between step 950 and 1000, normalized to control for different losses. 2.0 on the Y axis means the gradient to a parameter is twice as large, on average. The NormFormer increases the norm to fully connected parameters in later layers, while reducing the gradient norm to attention parameters at all layers. The results are discussed in detail in Section 4.

Learning Rate	0.003
Batch Size	524K Tokens
Parameters	124M+
Layers	12
Layer Dimension	768
Dropout	0
LR Warmup Updates	500
LR Scheduler	Linear Decay
Sequence Length	1024
Train Budget	470 V100 Hours

Table 8: Hyperparameters for ablation runs. This train budget allows the baseline model to run for 100,000 updates.

REFERENCES

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016.

Thomas Bachlechner, Bodhisattwa Prasad Majumder, Huanru Henry Mao, Garrison W Cottrell, and Julian McAuley. Rezero is all you need: Fast convergence at large depth. *arXiv preprint arXiv:2003.04887*, 2020.

Alexei Baevski and Michael Auli. Adaptive input representations for neural language modeling. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=ByxZX20qFQ.

David Balduzzi, Marcus Frean, Lennox Leary, JP Lewis, Kurt Wan-Duo Ma, and Brian McWilliams. The shattered gradients problem: If resnets are the answer, then what is the question? In *International Conference on Machine Learning*, pp. 342–350. PMLR, 2017.

Yonatan Bisk, Rowan Zellers, Ronan Le bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):7432–7439, Apr. 2020. doi: 10.1609/aaai.v34i05.6239. URL https://ojs.aaai.org/index.php/AAAI/article/view/6239.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

Xiaohan Chen, Yu Cheng, Shuohang Wang, Zhe Gan, Zhangyang Wang, and Jingjing Liu. Earlybert: Efficient bert training via early-bird lottery tickets, 2021.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. Unsupervised cross-lingual representation learning at scale. 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

Xiao Shi Huang, Felipe Perez, Jimmy Ba, and Maksims Volkovs. Improving transformer optimization through better initialization. In *International Conference on Machine Learning*, pp. 4475–4483. PMLR, 2020.

Opher Lieber, Or Sharir, Barak Lenz, and Yoav Shoham. Jurassic-1: Technical details and evaluation. Technical report, AI21 Labs, August 2021.

Liyuan Liu, Xiaodong Liu, Jianfeng Gao, Weizhu Chen, and Jiawei Han. Understanding the difficulty of training transformers. *arXiv preprint arXiv:2004.08249*, 2020.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. Mixed precision training. In *International Conference on Learning Representations*, 2018.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2381–2391, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1260. URL https://www.aclweb.org/anthology/D18-1260.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. A corpus and cloze evaluation for deeper understanding of commonsense stories. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 839–849, San Diego, California, June 2016. Association for Computational Linguistics. doi: 10.18653/v1/N16-1098. URL https://www.aclweb.org/anthology/N16-1098.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. FAIRSEQ: A fast, extensible toolkit for sequence modeling. 2019.

Emilio Parisotto, Francis Song, Jack Rae, Razvan Pascanu, Caglar Gulcehre, Siddhant Jayakumar, Max Jaderberg, Raphael Lopez Kaufman, Aidan Clark, Seb Noury, et al. Stabilizing transformers for reinforcement learning. In *International Conference on Machine Learning*, pp. 7487–7498. PMLR, 2020.

Ofir Press, Noah A. Smith, and Omer Levy. Improving transformer models by reordering their sublayers, 2020a.

Ofir Press, Noah A Smith, and Mike Lewis. Shortformer: Better language modeling using shorter inputs. *arXiv preprint arXiv:2012.15832*, 2020b.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. Technical report, OpenAI, 2019.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. 21:1–67, 2020.

Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. *arXiv preprint arXiv:2102.12092*, 2021.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):8732–8740, Apr. 2020. doi: 10.1609/aaai.v34i05.6399. URL https://ojs.aaai.org/index.php/AAAI/article/view/6399.

Noam Shazeer. Glu variants improve transformer, 2020.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015.

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

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. 2019.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tie-Yan Liu. On layer normalization in the transformer architecture, 2020.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4791–4800, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1472. URL https://www.aclweb.org/anthology/P19-1472.

Chen Zhu, Renkun Ni, Zheng Xu, Kezhi Kong, W Ronny Huang, and Tom Goldstein. Gradinit: Learning to initialize neural networks for stable and efficient training. *arXiv preprint arXiv:2102.08098*, 2021.

Yukun Zhu, Ryan Kiros, Richard Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. *arXiv preprint arXiv:1506.06724*, 2019.