On the Expressivity Role of LayerNorm in Transformers' Attention

Shaked Brody[†], **Uri Alon**[♠], **Eran Yahav**[†]
[†] Technion, Israel
[♠] Language Technologies Institute, Carnegie Mellon University, USA
{shakedbr,yahave}@cs.technion.ac.il
ualon@cs.cmu.edu

Abstract

Layer Normalization (LayerNorm) is an inherent component in all Transformer-based models. In this paper, we show that LayerNorm is crucial to the expressivity of the multi-head attention layer that follows it. This is in contrast to the common belief that LayerNorm's only role is to normalize the activations during the forward pass, and their gradients during the backward pass.

We consider a geometric interpretation of LayerNorm and show that it consists of two components: (a) *projection* of the input vectors to a $d - 1$ space that is orthogonal to the $[1, 1, ..., 1]$ vector, and (b) *scaling* of all vectors to the same norm of $\sqrt{d}$. We show that each of these components is *important for the attention layer that follows it in Transformers*: (a) *projection* allows the attention mechanism to create an attention query that attends to all keys equally, offloading the need to learn this operation by the attention; and (b) *scaling* allows each key to potentially receive the highest attention, and prevents keys from being "un-select-able". We show empirically that Transformers do indeed benefit from these properties of LayeNorm in general language modeling and even in computing simple functions such as "majority". Our code is available at https://github.com/tech-srl/layer_norm_expressivity_role.

1 Introduction

LayerNorm (Ba et al., 2016) is the most commonly used normalization technique in modern neural networks such as Transformers (Vaswani et al., 2017).

Originally, Ba et al. (2016) motivated Layer-Norm as an efficient way of normalizing the activations during the forward pass or providing distribution stability as in batch normalization (Ioffe and Szegedy, 2015). Later, Xu et al. (2019) and Xiong et al. (2020) argued that more importantly than normalizing forward activations, LayerNorm stabilizes the gradients during the *backward* pass.

However, in this work, we show that LayerNorm, which was originally proposed for RNNs, has an additional crucial role in the theoretical and practical *expressivity* of the multi-head attention layer that follows it in Transformers. [1] That is, Layer-Norm makes it easier for the Transformer to learn certain functions during training.

LayerNorm can be seen as two independent components: *projection* and *scaling*, that were merged into a single operator. First, LayerNorm *projects* its inputs onto a particular $d - 1$ space that is orthogonal to the "ones" $\vec{\mathbb{1}} = [1, 1, ..., 1]$ vector. This allows the attention layer that follows the Layer-Norm to create queries that are close to $\vec{\mathbb{1}}$, and thus attend to all keys equally, when needed, regardless of the identity of the keys. In Section 3 we show that this projection helps, for example, computing the "majority" among token types in a sequence.

Figure 1a shows how without LayerNorm, the keys and queries in a Transformer's attention have no apparent geometric structure. In contrast, Figure 1b shows that LayerNorm has projected all keys to the hyperplane that is orthogonal to the $\vec{\mathbb{1}}$ vector. Further, the attention mechanism has learned queries that are close to $\vec{\mathbb{1}}$, making them attend equally to any possible key, when trained to compute "majority". We analyze and prove this in Section 4.1.

The second component of LayerNorm is *scaling*: We show that LayerNorm scales the projected input to have an ℓ^2 norm of exactly $\sqrt{d}$. In Section 3, we show that scaling the input vectors prevents the problem of "unselectable" keys (Demeter et al., 2020; Grivas et al., 2022), where some key vectors are contained in the convex hull formed by the other keys, and thus can never get the highest attention score. Figures 1c and 1d show the average fraction

---

[1]Xiong et al. (2020) discuss the differences between placing LayerNorm *before* and *after* a Transformer layer. However, even when placing the LayerNorm *after* the layer, it appears right before the multi-head attention of the *next* layer.
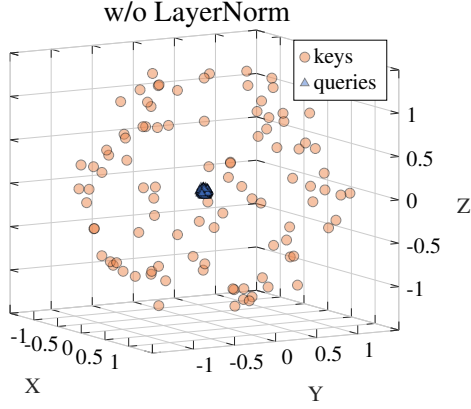
(a) Without LayerNorm, the model has learned key and query vectors without any apparent geometric structure.
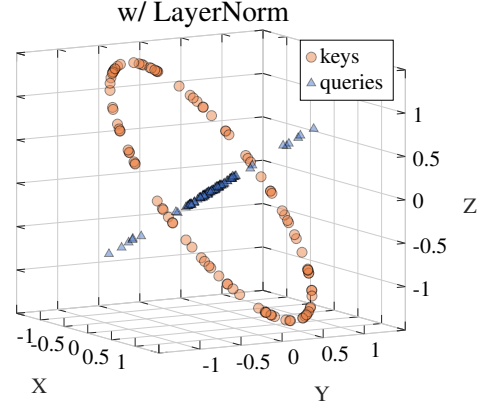
(b) LayerNorm projects the key vectors onto the same hyperplane so that the model can learn to align the queries to be orthogonal to the keys.
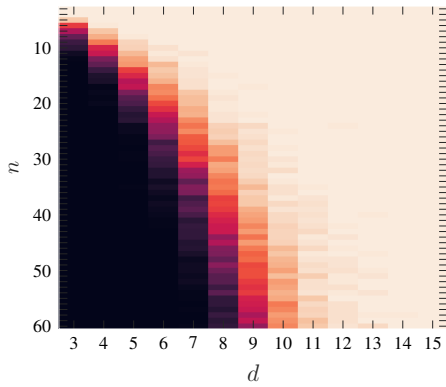
(c) Without LayerNorm, there are "unselectable" key vectors that cannot be selected by getting the maximal attention score (marked in darker colors).
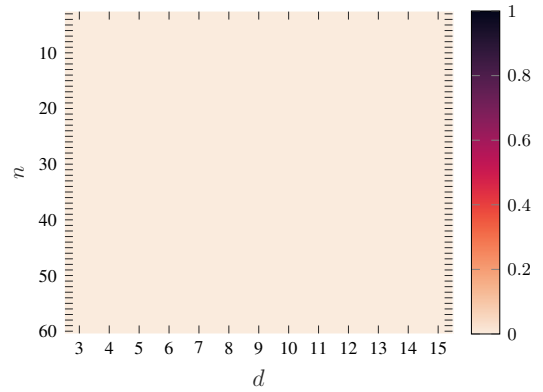
(d) LayerNorm eliminates the problem of "unselectable" key vectors: Applying LayerNorm allows any key to get the highest attention score.

Figure 1: Figures 1a and 1b show the effect of *projection* in LayerNorm, which makes all keys lie on the hyperplane that is orthogonal to the $\vec{\mathbb{1}}$ vector. Figures 1c and 1d show the effect of *scaling*, where n is the number of vectors, d is the dimension, and the color represents the average fraction of "unselectable" key vectors.

(out of 100 runs) of "unselectable" vectors which were randomly drawn from the normal distribution. As shown in Figure 1c, without LayerNorm, the probability of getting "unselectable" keys can be very high in certain settings. Nonetheless, as shown in Figure 1d, with LayerNorm, every key vector is always selectable. We analyze and prove this in Section 4.2.

These results reveal new aspects of the commonly used LayerNorm and show its importance to the attention mechanism in Transformers.

2 Decomposing LayerNorm

Given an input $x \in \mathbb{R}^d$, LayerNorm is defined as the following quotient:[2]

$$y = \frac{x - \mu}{\sigma} \quad (1)$$

where μ is the coordinate-wise average of x and σ is the coordinate-wise standard deviation:

$$\mu = \frac{1}{d} \sum_{i=1}^{d} x_i, \quad \sigma = \sqrt{\frac{1}{d} \sum_{i=1}^{d} (x_i - \mu)^2} \quad (2)$$

$$\boldsymbol{\mu} = [\mu, \mu, ..., \mu] \in \mathbb{R}^d \quad (3)$$

We start with the numerator $x - \mu$ and show that it corresponds to the projection of x onto the hyperplane $\mathcal{H}$ defined by the normal vector $\vec{\mathbb{1}} = [1, 1, ..., 1] \in \mathbb{R}^d$:

$$(x - \mu) \cdot \vec{\mathbb{1}} = x \cdot \vec{\mathbb{1}} - \mu \cdot \vec{\mathbb{1}}$$

$$\sum_{i=1}^{d} x_i - \left(\frac{1}{d} \sum_{i=1}^{d} x_i \right) \cdot d = 0 \quad (4)$$

That is, $x - \mu$ is always orthogonal to the $\vec{\mathbb{1}}$ vector. Next, we show that the denominator scales the

projected vector to have a norm of exactly $\sqrt{d}$:

$$||\boldsymbol{x}-\boldsymbol{\mu}|| = \sqrt{\sum_{i=1}^{d} (x_i - \mu)^2}$$
$$= \sqrt{d}\sqrt{\frac{1}{d}\sum_{i=1}^{d} (x_i - \mu)^2} = \sqrt{d} \cdot \sigma \quad (5)$$

Then, σ in the denominator of LayerNorm scales the projected vector to have a norm of exactly $\sqrt{d}$.

LayerNorm can thus be seen as two independent components: (a) *projection* of the input vectors onto the hyperplane orthogonal to $\vec{\mathbb{1}}$, and (b) *scaling* of the projected vectors to have a norm of $\sqrt{d}$.

3 Expressivity Role in Attention

Each of the components of LayerNorm supports the Transformer's attention in a different way. *Projection* helps creating a query that attends to all keys equally, when needed, while *scaling* helps the model to avoid the problem of "unselectable" keys.

Recall that in Transformers, given vectors $\boldsymbol{q}$ and $\boldsymbol{k}$, the attention scoring function is defined as:

$$s(\boldsymbol{q},\boldsymbol{k}) = \frac{(\boldsymbol{q}\boldsymbol{Q})(\boldsymbol{k}\boldsymbol{K})^{\top}}{\sqrt{d}} = \left(\frac{\boldsymbol{q}\boldsymbol{Q}\boldsymbol{K}^{\top}}{\sqrt{d}}\right)\boldsymbol{k}^{\top} \quad (6)$$

From this point, we refer to $\left(\frac{\boldsymbol{q}\boldsymbol{Q}\boldsymbol{K}^{\top}}{\sqrt{d}}\right)$ as "query", and to $\boldsymbol{k}$ as "key".

3.1 Projection

Projecting all attention keys to the same hyperplane can help attention attending to all keys equally. Since all projected keys are orthogonal to the hyperplane's normal $\vec{\mathbb{1}}$, this fact can be exploited in the training process by learning weights such that the queries will be parallel to $\vec{\mathbb{1}}$. That is, the attention can learn weights such that $\left(\frac{\boldsymbol{q}\boldsymbol{Q}\boldsymbol{K}^{\top}}{\sqrt{d}}\right) \approx c \cdot \vec{\mathbb{1}}$, which will result in $\left(\frac{\boldsymbol{q}\boldsymbol{Q}\boldsymbol{K}^{\top}}{\sqrt{d}}\right) \cdot \boldsymbol{k} \approx 0$ and thus $s(\boldsymbol{q},\boldsymbol{k}) = 0$ for any key $\boldsymbol{k}$.

Giving an equal score to all the keys can help, for example, in computing "majority", where the model needs to find the most frequent token in the input. In Section 4.1, we show that in the "majority" task, a Transformer learns to align the queries to be orthogonal to the keys, allowing a much faster convergence.

3.2 Scaling

Scaling the attention keys to the same size allows a Transformer to avoid the problem of "unselectable"

keys, where there are keys to which the attention cannot focus and give the highest score.

Let $\mathcal{S} = \{\boldsymbol{h}_1, ..., \boldsymbol{h}_{n-1}, \boldsymbol{h}_n\}$ be a set of key vectors, such that $\boldsymbol{h}_n$ lies within the convex hull formed by the other vectors in $\mathcal{S}$. Due to the linearity of the attention scoring function s, the attention mechanism cannot select $\boldsymbol{h}_n$ by giving it the highest attention score. We formulate this in the following theorem:

Theorem 1. *Given a set of vectors $\mathcal{S} = \{\boldsymbol{h}_1, ..., \boldsymbol{h}_{n-1}, \boldsymbol{h}_n\}$ such that $\boldsymbol{h}_n$ is interior to the convex hull of $\mathcal{S}$, then for all $\boldsymbol{v} \in \mathbb{R}^d$ (s.t. $\boldsymbol{v} \neq \vec{\boldsymbol{0}}$):*

$$\max_{i \in [n-1]} \boldsymbol{v}^{\top}\boldsymbol{h}_i > \boldsymbol{v}^{\top}\boldsymbol{h}_n$$

This means that key vectors that are inside the convex hull cannot be selected by getting the highest attention score. Applying LayerNorm to the keys ensures that all keys are *scaled* to the same size, and thus none of them lies inside the convex hull of $\mathcal{S}$. This allows the attention mechanism to potentially focus and select any desired key. The proof of Theorem 1 is provided in Appendix A.

In Section 4.2 we show that this happens in practice when we train a Transformer on language modeling: There are key vectors that lie within the convex hull and therefore cannot be selected by receiving the maximal attention.

4 Experimental Results

In this section, we empirically show the effects of the LayerNorm components – *projection* and *scaling* – on the attention mechanism in Transformers. We first show how a Transformer learns to use the *projection* of LayerNorm to compute "majority"; then, we show that *scaling* allows the model to avoid the problem of "unselectable" keys, allowing the attention mechanism to focus on any key.

4.1 Computing Majority

We demonstrate the ability to compute "majority" using the *projection* property. In this task, the goal is to predict the majority token type in a sequence. Given a sequence of tokens $t_1, t_2, ..., t_n \in \{C_1, C_2, ..., C_k\}$, the goal is to predict the token type C_i that occurs the most among $t_1, t_2, ..., t_n$. For a, a, b, b, b, c, c, for example, the model is trained to predict the output b, b, b, b, b, b, b. Solving this task can be performed simply using exact averaging of the keys.

We trained a single-layer Transformer encoder with dimension $d = 8$ and a single attention head.
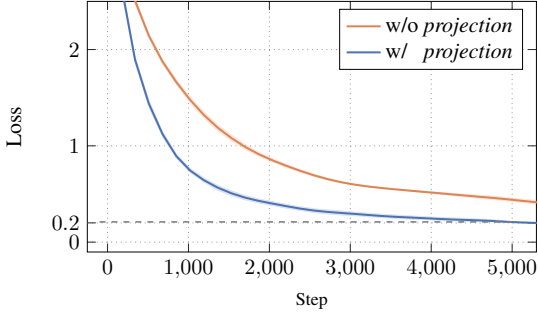
(a) Training loss: *With* projection, the model converges faster compared to the model *without* projection, which required 3x more steps.
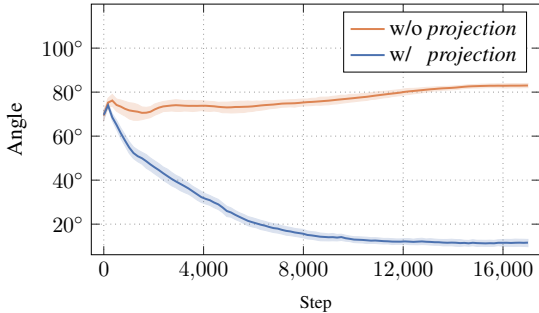
(b) The mean angle of the queries to the $\vec{\mathbb{1}}$ vector. *With* projection, since all keys are orthogonal to $\vec{\mathbb{1}}$, the model has learned to align the queries to be parallel to $\vec{\mathbb{1}}$ and thus give equal attention to all keys.

Figure 2: The training loss and mean angle of the queries to $\vec{\mathbb{1}}$ in the "majority" task across 10 runs with and without *projection*.

We experimented with standard LayerNorm compared to a LayerNorm without *projection* (having a numerator of simply x in Equation (1), similarly to the LayerNorm variant of Zhang and Sennrich (2019)). We trained each model 10 times using different random seeds.

Results Figure 2a shows that *with* projection, the model converges faster compared to *without* projection. We hypothesize that since all key vectors are orthogonal to the $\vec{\mathbb{1}}$ vector, the model can exploit this geometric structure, and learning the task is made easier. Figure 2b shows that indeed, the model *with* projection has learned to align the queries to the $\vec{\mathbb{1}}$ vector, decreasing the angle between the queries and $\vec{\mathbb{1}}$. On the contrary, a model *without* projection has to learn to solve this task "from scratch". This model also converged eventually, but it required 3x more training steps. Figure 3 shows a similar trend in the models' test accuracies.

4.2 Unselectable Keys

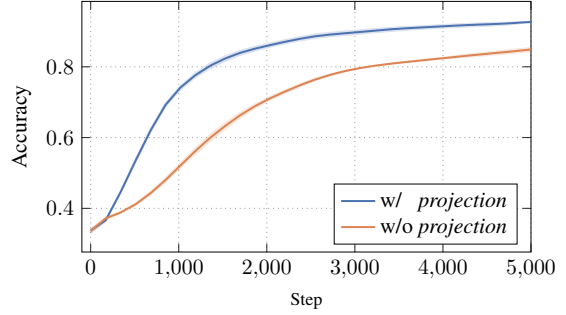We examined the fraction of "unselectable" keys in a Transformer model with and without the *scal-*

Figure 3: The test accuracy in the "majority" task across 10 runs with and without *projection*. *With* projection the model reaches high test accuracy faster compared to the model *without* projection.

Model	L_1	L_2	L_3	L_4
w/o *scaling*	51.0	32.2	34.7	36.8
w/ *scaling*	**0**	**0**	**0**	**0**

Table 1: The fraction of "unselectable" key vectors right before the attention mechanism of each layer of a language model without *scaling*. LayerNorm solves the "unselectable" keys problem using the *scaling* property. Without *scaling*, there are key vectors that cannot be selected by the attention mechanism.

ing component of LayerNorm using the method presented in Grivas et al. (2022). We trained a 4-layer language model (based on GPT2 architecture (Radford et al., 2019)) with $d = 8$ on Wikipedia [3] for 50K steps, and analyzed the inputs to the attention layer in each layer using sequences from the validation set of SQuAD (Rajpurkar et al., 2016).

Results Table 1 shows the following results: *Without scaling*, there are at least 32% "unselectable" keys that cannot receive maximal attention score *in each layer*. In contrast, *scaling* removes this problem and allows the model to potentially focus on any key. In Figures 4a and 4b, we show that this difference in "unselectable" keys is also reflected in higher training and test losses for the model that does not use *scaling*.

5 Conclusion

In this paper, we show that the commonly used LayerNorm component is crucial not only for the optimization process but also for the expressivity of attention in Transformers. We decompose LayerNorm into two geometric operations: *projecting*

---

[3] https://huggingface.co/datasets/wikipedia, 20220301.en split.
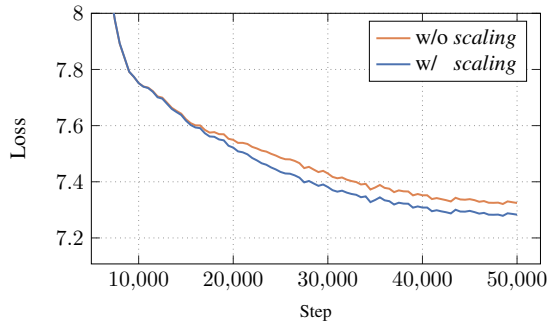
(a) Training loss: *With* scaling, the model converge faster compared to the model *without* scaling.
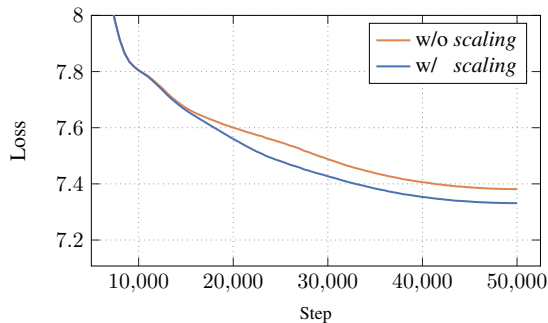
(b) Test loss: *With* scaling, the model achieves lower test loss faster compared to the model *without* scaling.

Figure 4: The training and test loss in the language modeling task.

the input vectors onto a subspace that is orthogonal to the $\vec{\mathbb{1}}$ vector, and *scaling* the projected vectors to have the same norm $\sqrt{d}$.

We show that *projection* helps to compute even simple tasks such as "majority" by performing an exact average of the keys and that *scaling* helps to avoid the problem of "unselectable" keys.

These findings are important for the community's understanding of attention and expressivity in Transformers. Further, these results raise a variety of follow-up questions, such as: why should the keys be orthogonal to the $\vec{\mathbb{1}}$ vector, instead of any, learnable, different vector for every layer? And what would happen if we force each layer's keys to be orthogonal to *multiple* normal vectors? To this end, we make our code publicly available at `https://github.com/tech-srl/layer_norm_expressivity_role`.

6   Limitations

In this work, we found that the implications of the geometric properties of LayerNorm affect mainly small models and are less evident for larger models.

We hypothesize that with a large hidden dimension, a Transformer model can find other solutions for computing "majority" using gradient descent and is, therefore, less dependent on the *projection* component. Further, we believe that the *scaling* component is less useful for high dimensional models, since with higher dimensions, it is less likely to encounter a set of vectors where some of them lie within the convex hull of the others. Therefore, we encourage the community to use LayerNorm before attention layers, especially for small models that operate on long sequences.

Moreover, the *projection* component is clearly a linear operator that can be expressed by a linear layer before the LayerNorm, as we show in Appendix C. Nevertheless, the importance of the projection holds as we discuss in Section 3, and the benefit of using this operator explicitly in Layer-Norm is shown in Section 4.1.

Acknowledgements

We thank Gail Weiss for the helpful discussions.

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *ArXiv preprint*, abs/1607.06450.

David Demeter, Gregory Kimmel, and Doug Downey. 2020. Stolen probability: A structural weakness of neural language models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2191–2197, Online. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Wikimedia Foundation. Wikimedia downloads.

Andreas Grivas, Nikolay Bogoychev, and Adam Lopez. 2022. Low-rank softmax can have unargmaxable classes in theory but rarely in practice. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6738–6758, Dublin, Ireland. Association for Computational Linguistics.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by

reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 448–456. JMLR.org.

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.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. *arXiv e-prints*, page arXiv:1606.05250.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tie-Yan Liu. 2020. On layer normalization in the transformer architecture. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 10524–10533. PMLR.

Jingjing Xu, Xu Sun, Zhiyuan Zhang, Guangxiang Zhao, and Junyang Lin. 2019. Understanding and improving layer normalization. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 4383–4393.

Biao Zhang and Rico Sennrich. 2019. Root mean square layer normalization. *Advances in Neural Information Processing Systems*, 32.

A   Proof of Theorem 1

Theorem 1. *Given a set of vectors* $\mathcal{S} = \{h_1, ..., h_{n-1}, h_n\}$ *such that* h_n *is interior to the convex hull of* $\mathcal{S}$, *then for all* $v \in \mathbb{R}^d$ *(s.t.* $v \neq \vec{0}$*):*

$$\max_{i \in [n-1]} v^\top h_i > v^\top h_n$$

We prove Theorem 1 using Theorem 2, presented and proved in Demeter et al. (2020):

Theorem 2 (Demeter et al. (2020)). *Let* C *be the convex hull of the embeddings* $\{x_i\}$ *of a vocabulary* V. *If an embedding* x_i *for word* $w_i \in V$ *is interior to* C, *then the maximum probability* $P(w_i)$ *assigned to* w_i *using a dot-product softmax is bounded by the probability assigned to at least one word* w_i *whose embedding is on the convex hull.*

Proof of Theorem 1. According to Theorem 2, since h_n is interior to the convex hull of $\mathcal{S}$, the maximum probability assigned to h_n is bounded by the maximum probability assigned to h_i for some $i \in [1-n]$, that is on the convex hull of $\mathcal{S}$. Since probability in Transformers is computed as a dot-product of the final hidden state of the Transformer u and the embedding vector, we can write:

$$u^\top h_i > u^\top h_n \tag{7}$$

for *any* $u \in \mathbb{R}^d$ (the probability is computed as a *softmax* of the dot-product logits, but since softmax is a monotonic function, a higher probability after the softmax necessarily implies a higher logit score).

Therefore, it also implies that for any $v = u \in \mathbb{R}^d$:

$$\max_{i \in [n-1]} v^\top h_i > v^\top h_n \tag{8}$$

$\square$

B   Characteristics of the Normalized Vectors

In this section, we discuss the characteristics of the LayerNorm inputs that are been normalized to the same point. Recall that the *projection* ensures that the normalized output lies on the hyperplane $\mathcal{H}$ defined by the normal vector $\vec{\mathbb{1}} = [1, 1, ..., 1] \in \mathbb{R}^d$.

Let v be a unit vector in $\mathcal{H}$:

$$v \perp \vec{\mathbb{1}} \wedge ||v|| = 1 \tag{9}$$

Therefore

$$\sum_{i=1}^{d} v_i = 0 \tag{10}$$

Let $\mathcal{M}$ be a **2D plane** that is defined using v and $\vec{\mathbb{1}}$. Its parametric representation is:

$$\mathcal{M} : sv + t\vec{\mathbb{1}} \tag{11}$$

Finally, let x be a vector in $\mathcal{M}$. Therefore, there exist $\alpha, \beta \in \mathbb{R}$ such that

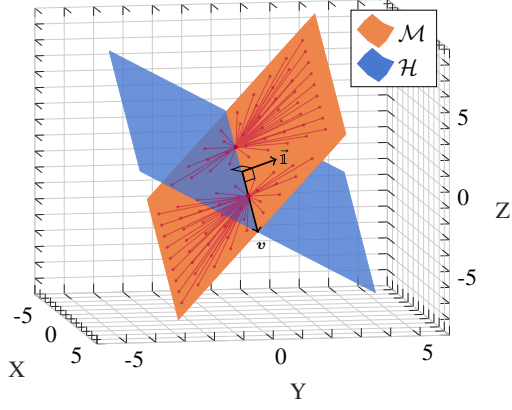$$x = \alpha v + \beta \vec{\mathbb{1}} \tag{12}$$

Figure 5: LayerNorm maps the points of $\mathcal{M}$ to exactly two points in $\mathcal{H}$.

Next, we apply LayerNorm to $\boldsymbol{x}$. First we project $\boldsymbol{x}$ onto $\mathcal{H}$:

$$\boldsymbol{x} - \boldsymbol{\mu} =$$

$$= \alpha\boldsymbol{v} + \beta\vec{\mathbb{1}} - \frac{1}{d}\sum_{i=1}^{d}(\alpha\boldsymbol{v}_i + \beta)\vec{\mathbb{1}}$$

$$= \alpha\boldsymbol{v} + \beta\vec{\mathbb{1}} - \alpha\left(\frac{1}{d}\sum_{i=1}^{d}\boldsymbol{v}_i\right)\vec{\mathbb{1}} - \beta\vec{\mathbb{1}} \tag{13}$$

$$= \alpha\boldsymbol{v} - \alpha\left(\frac{1}{d}\underbrace{\sum_{i=1}^{d}\boldsymbol{v}_i}_{=0}\right)\vec{\mathbb{1}}$$

$$= \alpha\boldsymbol{v}$$

Then, we scale the projected vector to be with a norm of $\sqrt{d}$ and get

$$\text{LayerNorm}\,(\boldsymbol{x}) = \sqrt{d}\frac{\alpha\boldsymbol{v}}{||\alpha\boldsymbol{v}||} \tag{14}$$

We split to cases:[4]

$$\text{LayerNorm}\,(\boldsymbol{x}) = \begin{cases} \sqrt{d}\boldsymbol{v} & \alpha > 0 \\ -\sqrt{d}\boldsymbol{v} & \alpha < 0 \end{cases} \tag{15}$$

To conclude, all vectors belonging to the 2D plane $\mathcal{M}$ are normalized to exactly two points, depending on their α component.

Since the subspaces $\mathcal{H}$ and $\left\{t\vec{\mathbb{1}}|t \in \mathbb{R}\right\}$ are direct sum of the whole space $\mathbb{R}^d$, each vector $\boldsymbol{u} \in \mathbb{R}^d$ has a unique representation as $\boldsymbol{u} = \alpha\boldsymbol{v} + \beta\vec{\mathbb{1}}$. That is, each vector $\boldsymbol{u} \in \mathbb{R}^d$ belongs to some 2D plane $\mathcal{M}$, defined by $\vec{\mathbb{1}}$ and some vector $\boldsymbol{v} \in \mathcal{H}$,

---

[4]As implied from the original formulation of LayerNorm (Ba et al., 2016), LayerNorm is undefined for $\alpha = 0$.

and thus we can characterize the set of points that are being normalized to the same point. Figure 5 illustrates this behavior.

C Constructing the Projection

The *projection* of LayerNorm is a linear transformation, and thus, in this section, we show explicitly the construction of the projection matrix $\boldsymbol{P}$, such that

$$\boldsymbol{P}\boldsymbol{x} = \boldsymbol{x} - \boldsymbol{\mu} \tag{16}$$

Let $V = \mathbb{R}^d$, $W = \left\{\alpha\vec{\mathbb{1}}|\alpha \in \mathbb{R}\right\}$, and $U = \left\{\boldsymbol{x}|\boldsymbol{x}\perp\vec{\mathbb{1}} = [1,1,...,1] \in \mathbb{R}^d\right\}$ be linear subspaces of V.

Let B_U, B_W be the bases of U and W respectively:

$$B_U = \{\boldsymbol{u}_1, \boldsymbol{u}_2, ..., \boldsymbol{u}_{d-1}\} \tag{17}$$

$$B_W = \left\{\vec{\mathbb{1}}\right\} \tag{18}$$

We can define a basis $C = B_U \cup B_W$ of V.

We also denote the standard basis E of V:

$$E = \{\boldsymbol{e}_1, \boldsymbol{e}_2, ..., \boldsymbol{e}_d\} \tag{19}$$

Since $U \cap W = \{0\}$, we have that $U \oplus W = V$ (direct sum). Therefore, each $\boldsymbol{x} \in V$ has a unique representation as $\boldsymbol{x} = \boldsymbol{u} + \boldsymbol{w}$ where $\boldsymbol{u} \in U$ and $\boldsymbol{w} \in W$.

We can also write $\boldsymbol{x}$ using B_U, B_W:

$$\boldsymbol{x} = \alpha\vec{\mathbb{1}} + \sum_{i=1}^{d-1}\beta_i\boldsymbol{u}_i \tag{20}$$

Since we look for the projection of $\boldsymbol{x}$ onto U in the direction of W, we want that

$$\boldsymbol{P}\boldsymbol{x} = \sum_{i}^{d-1}\beta_i\boldsymbol{u}_i \tag{21}$$

To achieve this, we first, change the basis of V from E to C, remove the $\alpha\vec{\mathbb{1}}$ component, and then change back to the standard basis E.

Let M_E^C be the change of basis matrix from basis C to the standard basis E:

$$M_E^C = \begin{bmatrix} | & | & & | & | \\ \boldsymbol{u}_1 & \boldsymbol{u}_2 & \dots & \boldsymbol{u}_{d-1} & \vec{\mathbb{1}} \\ | & | & & | & | \end{bmatrix} \tag{22}$$

Therefore

$$\boldsymbol{P} = M_E^C \boldsymbol{A} \left(M_E^C\right)^{-1} \tag{23}$$

14217

Where

$$A = \begin{bmatrix} | & | & & | & | \\ e_1 & e_2 & \ldots & e_{d-1} & \mathbf{0} \\ | & | & & | & | \end{bmatrix} \quad (24)$$

To get an explicit $P \in \mathbb{R}^{d \times d}$, we instantiate the basis B_U:

$$B_U = \left\{ \begin{bmatrix} 1-d \\ 1 \\ \vdots \\ 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1-d \\ \vdots \\ 1 \\ 1 \end{bmatrix}, ..., \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1-d \\ 1 \end{bmatrix} \right\} \quad (25)$$

Therefore

$$M_E^C = \begin{bmatrix} 1-d & 1 & \cdots & 1 & 1 \\ 1 & 1-d & \cdots & 1 & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & 1 & \cdots & 1-d & 1 \\ 1 & 1 & \cdots & 1 & 1 \end{bmatrix} \quad (26)$$

$$\left(M_E^C\right)^{-1} = \frac{1}{d} \begin{bmatrix} -1 & & & & 1 \\ & -1 & & & 1 \\ & & \ddots & & \vdots \\ & & & -1 & 1 \\ 1 & 1 & \cdots & 1 & 1 \end{bmatrix} \quad (27)$$

And we get

$$P = \frac{1}{d} \begin{bmatrix} d-1 & -1 & \cdots & -1 \\ -1 & d-1 & \cdots & -1 \\ \vdots & \vdots & \ddots & \vdots \\ -1 & -1 & \cdots & d-1 \end{bmatrix} \quad (28)$$

D   Unselectable Keys

Table 2 shows the fraction of "unselectable" keys in a language model with LayerNorm before and after the application of LayerNorm.

	L_1	L_2	L_3	L_4
Before LayerNorm	44.8	28.5	22.3	26.1
After LayerNorm	**0**	**0**	**0**	**0**

Table 2: The fraction of "unselectable" key vectors before and after the LayerNorm followed by the attention mechanism of each layer of a language model. LayerNorm solves the "unselectable" keys problem using the *scaling* property.

To illustrate the impact of "unselectable" tokens, we give some examples from the validation set of Stanford TreeBank (Socher et al., 2013), which is used as a benchmark for the sentiment analysis task. Our results show that important tokens may be "unselectable". We highlighted in bold any token that is "unselectable' in at least one of the layers. Table 3 shows the results of running a language model without LayerNorm (Section 4.2) on the validation set. We also trained a 4-layer Transformer encoder without LayerNorms (based on BERT architecture (Devlin et al., 2019)) on the sentiment analysis task. Table 4 shows the results of running this model on the validation set.

E   Experimental Setup

In this section, we detail the setup of the experiments shown in Section 4.

E.1   Majority

In the experiments, we used a learning rate of 0.001 with a linear scheduler, a hidden size of $d = 8$ (total of 584 learnable parameters), a batch size of 6000, a sequence length of 50, 20 different classes, and the Adam optimizer. We trained the models for 1000 epochs consisting of 17K steps.

The "majority" dataset contains 80K training examples and 20K test examples. Each example is a sequence of length 50 consisting of tokens belonging to one of 20 different classes.

E.2   Language Modeling

We trained a language model with GPT2 architecture (Radford et al., 2019) using the Huggingface library, on the Huggingface processed Wikipedia dataset (20220301.en split, licenses CC BY-SA and CC BY-SA) (Foundation), and tested it on SQuAD (Rajpurkar et al., 2016) (license CC BY-SA 4.0). We used these datasets only to demonstrate the "unselectable" keys problem, and thus we did not violate any of their license conditions. We used the same hyperparameters as Radford et al. (2019), except that we used a hidden size of 8, 4 hidden layers, a learning rate of 5e-5, and a window size of 1024 tokens, resulting in a model with 414K learnable parameters. We train the model on the Wikipedia dataset (6.5M examples) for 50K steps and report our findings on 1000 randomly selected examples from the validation set of SQuAD.

whether you like rap music or loathe it, you can't deny either the tragic loss of two young men in the prime of their **talent** or the **power** of this movie.

it is great summer **fun** to watch arnold and his buddy gerald bounce off a quirky cast of characters.

the lion king was a roaring **success** when it was released eight years ago, but on imax it seems better, not just bigger.

it provides the grand, intelligent **entertainment** of a superior cast playing smart people amid a compelling plot.

some of their jokes work, but most fail miserably and in the end, pumpkin is **far** more **offensive** than it is funny.

Table 3: Examples from the validation set of Stanford TreeBank (Socher et al., 2013). Any token that is "unselectable' in at least one of the layers of the language model (Section 4.2) is marked in **bold**.

---

it's hard to **like** a film about a guy who is utterly unlikeable, and shiner, starring michael caine as an aging british boxing promoter desperate for a taste of fame and fortune, is certainly that.

you'll gasp appalled and **laugh** outraged and possibly, watching the spectacle of a promising young lad treading desperately in a nasty sea, shed an errant tear.

this is **wild** surreal **stuff**, but brilliant and the camera just kind of sits there and lets you look at this and its like you're going from one room to the next and none of them have any relation to the other.

it's a much more **emotional** journey than what shyamalan has given us in his past two movies, and gibson, stepping in for bruce willis, is the perfect actor to take us on the trip.

although german cooking does not come readily to **mind** when considering the world's best cuisine, mostly martha could make deutchland a popular destination for hungry tourists.

Table 4: Examples from the validation set of Stanford TreeBank (Socher et al., 2013). Each **bold** token is "unselectable' in at least one of the layers of a Transformer encoder without LayerNorm, trained on the sentiment analysis task. These examples show that important tokens that may be necessary for the task are "unselectable", which may affect the encoder's ability to learn the task correctly.

E.3 Sentiment Analysis Task

We trained a Transformer encoder with BERT architecture (Devlin et al., 2019) without LayerNorm layers with 50K steps on the Stanford TreeBank dataset (Socher et al., 2013) (Table 4). We used the same hyperparameters as Devlin et al. (2019), except that we used a hidden size of 8, 4 hidden layers, and a learning rate of 5e-5, resulting in a model with 446K learnable parameters.

A For every submission:

☑ A1. Did you describe the limitations of your work?
6

☐ A2. Did you discuss any potential risks of your work?
Not applicable. Our work is about the expressivity of LayerNorm in Transformers, and thus it does not raise any risks that did not exist in the widely used Transformers in general.

☑ A3. Do the abstract and introduction summarize the paper's main claims?
1

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?
4

☑ B1. Did you cite the creators of artifacts you used?
4

☑ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
F

☑ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
F

☑ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
F

☑ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
F

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
F

C ☑ Did you run computational experiments?
4

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
F

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
F

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
4

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
F

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
No response.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
No response.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
No response.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.