Merging Text Transformer Models from Different Initializations

Neha Verma NVERMA7@JHU.EDU
Johns Hopkins University

Maha Elbayad ELBAYADM@META.COM
Meta

Abstract

Recent work on one-shot permutation-based model merging has shown impressive low- or zero-barrier mode connectivity between models from completely different initializations. However, this line of work has not yet extended to Transformers, despite their dominant popularity in the language domain. Therefore, in this work, we investigate the extent to which separate Transformer minima learn similar features, and propose a model merging technique to investigate the relationship between these minima in the loss landscape. The specifics of the architecture, like its residual connections, multi-headed attention, and discrete, sequential input, require specific interventions in order to compute model permutations that remain within the same functional equivalence class. In merging these models with our method, we consistently find lower loss barriers between minima compared to model averaging for several models trained on a masked-language modeling task. Our results show that the minima of these models are less sharp and isolated than previously understood, and provide a basis for future work on further understanding Transformer solutions.

1. Introduction

The geometry of loss landscapes is the subject of extensive prior work attempting to understand the behavior and properties of different solutions [7, 15, 17]. Prior work has found different types of geometric paths of low loss between converged models, demonstrating a degree of connectedness between separately trained minima [10, 11, 26]. Much of this work has involved permuting the weights of these models in order to compare them within a more similar loss space.

While this research has led to important conclusions about loss landscapes between separately trained models, the Transformer architecture has been largely unexplored in terms of understanding its permutation symmetries and loss landscape geometry. Prior work has emphasized the importance of understanding loss landscape geometry, as a better understanding can lead to improvements in optimization and ensembling techniques [1, 11]. There has been some prior work in merging Transformers [13, 14], but these do not provide insights on their loss landscape properties [2].

Therefore, in this work, we explore the extent to which separately trained Transformer models learn similar representations, and then propose a permutation-based merging method to align representations from these separate models. We specifically investigate their connectivity through the lens of permutation-invariant linear interpolation [8].

Our contributions are the following:

1. We introduce a new model merging algorithm based on model permutations that combines Transformers from separate initializations.[1]

1. We release our code at https://github.com/nverma1/merging-text-transformers

2. We demonstrate reduced loss barriers between masked language models trained from completely separate initializations compared to vanilla merging.

2. Proposed Transformer Merging Method

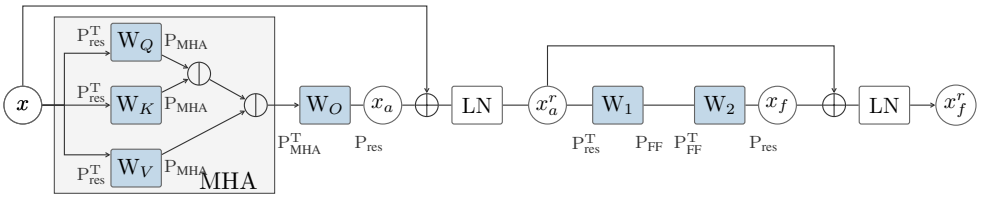

Figure 1: Example of a Transformer layer with parameters in blue, specific hidden states as circles, the flow of operations indicated with arrows, and proposed permute/unpermute matrices indicated.

In this section, we describe the components of our method that address how to permute specific portions of a Transformer model $\boldsymbol{\theta}_B$ in order to bring them into alignment with a separately trained model, $\boldsymbol{\theta}_A$. We diagram a Transformer layer in Figure 1, and the proposed permutation operations that we describe in detail throughout this section. We focus on parameters from Multi-Headed Attention and Feed-Forward layers in this section. We also describe merging parameters involved in Add&Norm or residual computations in Appendix B, as they do not appear in our final method.

2.1. Computing Correlation and Permutation Matrices

Given two separately trained minima $\boldsymbol{\theta}_A$ and $\boldsymbol{\theta}_B$, we compute post-activation features for each sublayer parameter $\boldsymbol{W}_\ell \subset \boldsymbol{\theta}$ in order to find corresponding features between these models [1, 25]. We compute d-dimensional activations across n tokens from both models, and then cross-correlation matrix $\boldsymbol{C} \in \mathbb{R}^{d \times d}$. The optimal permutation π is computed as: $\arg\max_\pi \sum_{i=1}^{d} \boldsymbol{C}_{i,\pi(i)}$. This assignment problem is solved using the Jonker-Volgenant algorithm [1, 3, 26].

After converting π to its corresponding permutation matrix $\boldsymbol{P}$, we can apply $\boldsymbol{P}$ to the original weight matrix $\boldsymbol{W}_\ell^B \subset \boldsymbol{\theta}_B$ so that the order of the layer's features most closely resembles that of $\boldsymbol{W}_\ell^A \subset \boldsymbol{\theta}_A$ before finally interpolating the models: $\boldsymbol{W}_\ell^{B'} \leftarrow \boldsymbol{P}\boldsymbol{W}_\ell^B$. We also apply $\boldsymbol{P}^{\mathrm{T}} = \boldsymbol{P}^{-1}$ to the next layer in order to unpermute the new ordering in model θ_B: $\boldsymbol{W}_{\ell+1}^{B'} \leftarrow \boldsymbol{W}_{\ell+1}^B \boldsymbol{P}^{\mathrm{T}}$.

2.2. Multi-Headed Attention

Since our models are trained separately, the correspondence of their attention heads may differ in addition to the features within each head. To align attention features, we collect hidden states just before the linear layer following dot-product attention. We partition the attention correlation matrix by heads into $\#_{\text{heads}} \times \#_{\text{heads}}$ correlation matrices, for each potential head pair. Let $\boldsymbol{C}^{jk}$ be the block of the correlation matrix corresponding to head pair (j, k). We then compute the optimal permutation for each unique pair, and store its head-internal permutation and cost from the following:

$$\text{cost}(j,k) = \max_\pi \sum_{i=1}^{d_k} \boldsymbol{C}_{i,\pi(i)}^{jk}. \tag{1}$$

We then compute the outer head correspondence permutation with a new assignment problem:

$$\pi_{\text{outer}} = \arg\max_{\pi} \sum_{j=1}^{h} \text{cost}(j, \pi(j)). \tag{2}$$

The outer permutation dictates the subset of previously computed inner permutations used in the final permutation, and the order in which to concatenate them, resulting in our 2-staged MHA permutation. The resulting permutation matrix $\boldsymbol{P}_{\text{MHA}}$ applies as following: $\boldsymbol{P}_{\text{MHA}}^{\text{T}}$ can apply to attention linear layer $\boldsymbol{W}_O$, and we apply $\boldsymbol{P}_{\text{MHA}}$ to each of $\boldsymbol{W}_V$, $\boldsymbol{W}_K$, and $\boldsymbol{W}_Q$.

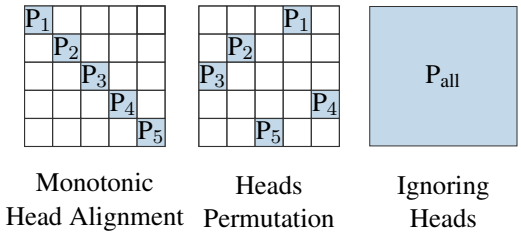

Figure 2: Example permutation matrices resulting from different strategies for attention head alignment. Each $\boldsymbol{P}_i$ reflects permutations for features within attention heads.

We show an example of an output from our proposed algorithm in Figure 2, as well as some baseline approaches. The first permutation matrix shows the resulting block-diagonal structure of assuming that heads are aligned between different minima. The second matrix shows an example from our method, and the third matrix disregards head structure and allows permuting features across heads.[2]

2.3. Feed-Forward Layers

Unlike the residual stream or Multi-Headed attention, Feed-Forward sub-layers require no special attention in order to permute them into a new space. We simply compute correlations from the features after the computation of the first Feed-Forward layer ($\boldsymbol{W}_1$), and compute $\boldsymbol{P}, \boldsymbol{P}^{\text{T}}$ separately for each Transformer layer. We apply these permutations as described in Section 2.1.

3. Experimental Settings

3.1. Models and Datasets

We use 5 different BERT models, seeds 1 through 5, from the MultiBERTs reproductions throughout this work[4, 22]. These models differ by their random initialization and batch ordering. All of our reported experiments include a mean and standard error across the $\binom{5}{2} = 10$ unique pairings.

We use masked language modeling performance as our evaluation. We use the validation set of the Wikitext-103 benchmark as our evaluation data [19][3]. For computing model activations, we extract a random[4] sample of 100k sentences of the Books corpus [29], as it was part of the original BERT pre-training data [4].

2. We note that ignoring heads will not lead to a valid permutation π where $f(\boldsymbol{x}; \boldsymbol{\theta}) = f(\boldsymbol{x}; \pi(\boldsymbol{\theta}))$, but we still include it for experimental comparisons.
3. We obtain the `wikitext-103-raw-v1` version, available from https://huggingface.co/datasets/wikitext
4. We take a diverse sample across different genres among the books available.

3.2. Evaluation

To compute loss barriers, we compute 21 evenly spaced interpolations of $\boldsymbol{\theta}_A$ and $\boldsymbol{\theta}_B$, as $\lambda\boldsymbol{\theta}_A + (1-\lambda)\boldsymbol{\theta}_B$. We use the definition of loss-barrier as Frankle and Carbin [9][5], defined as the maximum difference between the loss of an interpolation and the average loss of the base models:

$$\max_{\lambda} \mathcal{L}(\lambda\boldsymbol{\theta}_A + (1-\lambda)\boldsymbol{\theta}_B) - \frac{1}{2}(\mathcal{L}(\boldsymbol{\theta}_A) + \mathcal{L}(\boldsymbol{\theta}_B)). \tag{3}$$

To compute MLM loss/pseudo-perplexity, we use a masking probability of $p = 0.15$ across block sizes of 128 tokens. For N masked samples in text $\mathbf{W}$, we compute pseudo-perplexity as:

$$\text{Pseudo-PPL}(\mathbf{W};\boldsymbol{\theta}) = 2^{-\frac{1}{N}\sum_{i=1}^{N}\log_2 p_{\boldsymbol{\theta}}(w_i|\mathbf{W}_{-i})} \tag{4}$$

4. Results and Analysis

4.1. By component

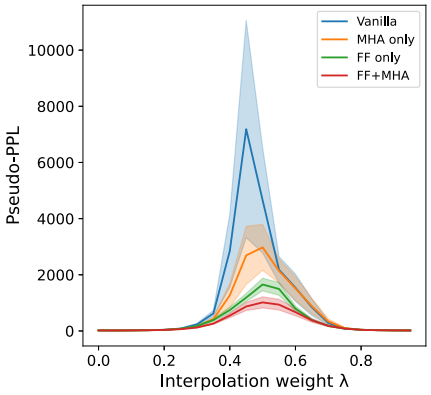

Figure 3: Pseudo-perplexity scores of BERTs, trained on the masked language modeling task, combined using our method. Results across 10 merges are shown with standard error.

We report results on the 10 MultiBERTs merges after merging different sets of Transformer components described in Section 2. We show pseudo-perplexity results across the range of interpolations, displayed Figure 3. We report vanilla averaging, merging all feed-forward sublayers, merging all multi-headed attention sublayers, and merging all feed-forward and multi-headed attention sublayers. Aligning and merging either the feed-forward sublayers or the attention sublayers clearly leads to a perplexity reduction over the baseline, and their combination leads to a stark reduction, of almost $7\times$ the original perplexity at $\lambda = 0.5$. We do not include permuting parameters involved in residual connections (Section B) and the output projection (Section 2.3) here as they do not outperform merging only feed-forward and attention sublayers. We leave further investigation of the difficulty of merging residual-involved weights for future work.

The consistently reduced barrier between minima indicates that these different models are connected with a lower loss path than seen without considering these models within a more similar loss space. We note that we do not observe a linear or convex loss path between these models, as sometimes observed in previous work on MLPs, ResNets, and VGGs [1, 8, 26]. Many possible reasons exist for this discrepancy. For example, in this prior work, the same data is generally used to train models, compute activations for alignment and merging, and compute loss barriers. Due to the extensive pretraining data of these masked language models, and the limited alignment and

5. Referred to as *linear interpolation instability* in this work.

evaluation data we use, we do not test for linear mode connectivity in the same manner. Instead, the stark loss reduction seems to indicate that minima are connected with a barrier *at least as high* as what we report.

4.2. Multi-headed attention

Table 1: Loss Barriers of merged MultiBERTs with feed-forward and attention components merged with different methods. Maintaining head structure in the permutation while allowing different head correspondences between models is the most optimal permutation.

Method	Loss Barrier ↓	Std. Err.
Vanilla Attention Avg.	*4.31*	*0.21*
Monotonic Head Alignment	4.13	0.20
Ignore-Heads	3.97	0.25
Head-Perm	**3.71**	0.23

We report loss barriers for our Head-Permutation MHA approach as compared to some alternatives also described in Section 2.2 in Table 1. These results reflect permuting both attention parameters as well as feed-forward parameters. We see that our proposed Head-Permutation approach for the attention sub-layer outperforms simple attention averaging, as well as approaches ignoring the multi-headed structure of the weight parameters (*Ignore-Heads*), and not allowing for different head correspondences across different models (*Monotonic*). We also show an example correlation matrix between the first multi-headed attention layer from 2 different MultiBERTs models in Figure 4. The correlation matrix shows clear attention head boundaries, as well as a scattered pattern that supports our proposed technique that does not assume any monotonically ordered head correspondence.

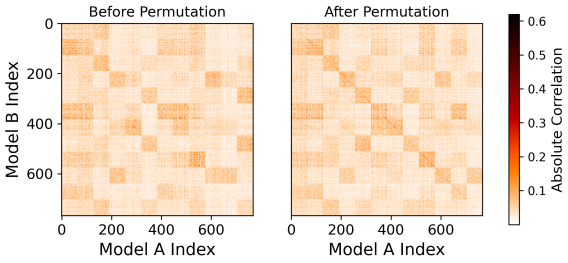

Figure 4: Visualization of correlation matrices between features before and after permuting. These features are from the seventh multi-headed attention layer from 2 different MultiBERTs models.

5. Discussion and Conclusion

By considering the set of functionally equivalent Transformers reachable using permutation mappings, we consistently find linear paths between models with lower loss than vanilla interpolation. This conclusion about the connectedness between these models has implications on our understanding of the "smoothness" of Transformer loss space and the sharpness of their minima. This understanding of the geometric properties of minima can have implications in how we design optimization methods, ensembles of models, and additional merging techniques. For example, it is widely contested whether sharp minima can generalize as well as flat minima across many deep learning models [5, 16]. As we take only a first attempt at connecting separately trained Transformers along a lower loss path, there is much room for future work in understanding Transformer loss landscapes.

References

[1] Samuel Ainsworth, Jonathan Hayase, and Siddhartha Srinivasa. Git re-basin: Merging models modulo permutation symmetries. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=CQsmMYmlP5T.

[2] An Mei Chen, Haw-minn Lu, and Robert Hecht-Nielsen. On the geometry of feedforward neural network error surfaces. *Neural Computation*, 5(6):910–927, 1993. doi: 10.1162/neco.1993.5.6.910.

[3] David F. Crouse. On implementing 2d rectangular assignment algorithms. *IEEE Transactions on Aerospace and Electronic Systems*, 52(4):1679–1696, 2016. doi: 10.1109/TAES.2016.140952.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *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, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423.

[5] Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. Sharp minima can generalize for deep nets. In *International Conference on Machine Learning*, pages 1019–1028. PMLR, 2017.

[6] Felix Draxler, Kambis Veschgini, Manfred Salmhofer, and Fred Hamprecht. Essentially no barriers in neural network energy landscape. In *International conference on machine learning*, pages 1309–1318. PMLR, 2018.

[7] Simon S. Du, Xiyu Zhai, Barnabas Poczos, and Aarti Singh. Gradient descent provably optimizes over-parameterized neural networks. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=S1eK3i09YQ.

[8] Rahim Entezari, Hanie Sedghi, Olga Saukh, and Behnam Neyshabur. The role of permutation invariance in linear mode connectivity of neural networks. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=dNigytemkL.

[9] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2018.

[10] C Daniel Freeman and Joan Bruna. Topology and geometry of half-rectified network optimization. In *5th International Conference on Learning Representations, ICLR 2017*, 2017.

[11] Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry P Vetrov, and Andrew Gordon Wilson. Loss surfaces, mode connectivity, and fast ensembling of dnns. In *Advances in Neural Information Processing Systems*, 2018.

[12] 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*, pages 770–778, 2016.

[13] Moritz Imfeld, Jacopo Graldi, Marco Giordano, Thomas Hofmann, Sotiris Anagnostidis, and Sidak Pal Singh. Transformer fusion with optimal transport, 2023.

[14] Xisen Jin, Xiang Ren, Daniel Preotiuc-Pietro, and Pengxiang Cheng. Dataless knowledge fusion by merging weights of language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=FCnohuR6AnM.

[15] Kenji Kawaguchi. Deep learning without poor local minima. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper_files/paper/2016/file/f2fc990265c712c49d51a18a32b39f0c-Paper.pdf.

[16] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. In *International Conference on Learning Representations*, 2016.

[17] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. *Advances in neural information processing systems*, 31, 2018.

[18] Chaoyue Liu, Libin Zhu, and Mikhail Belkin. Loss landscapes and optimization in over-parameterized non-linear systems and neural networks. *Applied and Computational Harmonic Analysis*, 59:85–116, 2022.

[19] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2016.

[20] Quynh Nguyen, Mahesh Chandra Mukkamala, and Matthias Hein. On the loss landscape of a class of deep neural networks with no bad local valleys. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HJgXsjA5tQ.

[21] Alexandre Rame, Matthieu Kirchmeyer, Thibaud Rahier, Alain Rakotomamonjy, patrick gallinari, and Matthieu Cord. Diverse weight averaging for out-of-distribution generalization. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=tq_J_MqB3UB.

[22] Thibault Sellam, Steve Yadlowsky, Ian Tenney, Jason Wei, Naomi Saphra, Alexander D'Amour, Tal Linzen, Jasmijn Bastings, Iulia Raluca Turc, Jacob Eisenstein, et al. The multiberts: Bert reproductions for robustness analysis. In *International Conference on Learning Representations*, 2021.

[23] Berfin Simsek, François Ged, Arthur Jacot, Francesco Spadaro, Clement Hongler, Wulfram Gerstner, and Johanni Brea. Geometry of the loss landscape in overparameterized neural networks: Symmetries and invariances. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 9722–9732. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/simsek21a.html.

[24] Sidak Pal Singh and Martin Jaggi. Model fusion via optimal transport. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 22045–22055. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/fb2697869f56484404c8ceee2985b01d-Paper.pdf.

[25] George Stoica, Daniel Bolya, Jakob Bjorner, Taylor Hearn, and Judy Hoffman. Zipit! merging models from different tasks without training. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=LEYUkvdUhq.

[26] Norman Tatro, Pin-Yu Chen, Payel Das, Igor Melnyk, Prasanna Sattigeri, and Rongjie Lai. Optimizing mode connectivity via neuron alignment. *Advances in Neural Information Processing Systems*, 33, 2020.

[27] Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International Conference on Machine Learning*, pages 23965–23998. PMLR, 2022.

[28] Shi-hua Zhan, Juan Lin, Ze-jun Zhang, Yi-wen Zhong, et al. List-based simulated annealing algorithm for traveling salesman problem. *Computational intelligence and neuroscience*, 2016, 2016.

[29] Yukun Zhu, Ryan Kiros, Rich 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. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27, 2015.

Appendix A. Related Work

Loss Landscape & Mode Connectivity. Loss landscapes of neural networks trained with an SGD variant have been shown to contain infinitely many global minimizers that are equally reachable via SGD [7, 15]. Overparameterization is one of the reasons behind the abundance of minima leading to different functions that behave similarly on the training data [18, 20, 23]. Permutation and scaling invariances also lead to functionally identical minima that differ in the weight space [8, 26].

Prior work has established that the optima of neural networks are connected by simple curves over which training and test accuracy are nearly constant (no loss barrier) [6, 10, 11]. This phenomenon is referred to as *mode connectivity*. Entezari et al. [8] conjectured that if the permutation invariances of neural networks are taken into account, these optima are linearly mode connected, i.e. the linear path connecting these two models has no loss barrier. This is *linear mode connectivity*.

Interpolating Models Empirically, linear interpolation between neural network weights has become an important tool. In the context of fine-tuning the same large pre-trained model, averaging models enabled state-of the art accuracy on ImageNet [27]. Rame et al. [21], Wortsman et al. [27] established that if fine-tuned models lie in a single low error basin, then weight averaging performs similarly to ensembling.

Prior work on linear interpolation-based model merging has focused on improving the algorithms used to bring the hidden units of two networks into alignment, in order to reduce the barrier to interpolation between them. Such algorithms have included an optimal transport-based method between ResNets [12, 24], and a simulated annealing based method [8, 28] between wide MLPs. Ainsworth et al. [1] develop several permutation-based algorithms for MLPs and ResNets, and demonstrate zero-barrier linear mode connectivity on widened ResNets. Stoica et al. [25] extend this work and allow for feature merges to happen within each model as well, improving model merging outcomes.

Appendix B. Merging Residual Connection Parameters

We diagram the residual connections of a Transformer layer and their relationships to model parameters in Figure 1. The connections can be formulated as the following, where LN refers to LayerNorm:

$$\begin{aligned} \boldsymbol{x}_a^r &= \mathrm{LN}(\boldsymbol{W}_O \mathrm{MHA}(\boldsymbol{x}) + \boldsymbol{x}), \\ \boldsymbol{x}_f^r &= \mathrm{LN}(\boldsymbol{W}_2 \mathrm{ReLU}(\boldsymbol{W}_1 \boldsymbol{x}_a^r) + \boldsymbol{x}_a^r). \end{aligned} \tag{5}$$

As seen in the equations, the input and output of both sublayers are added to create a new output, which implies that if a permutation operation is applied to the output state, the permutation needs to be the same for both addends.

We note that the addends are normalized via the LayerNorm module, and any permutation to the output would need to permute the features of the LayerNorm module as well. We apply the permutation to the weights of LN. Because LN is not a full weight matrix, and maintains the same feature ordering as its input, we must apply the permutation to the addends of the residual connections as well [25].

Now, ignoring the parameters from the LayerNorm module for a moment, we see that applying the permutation to the output of the second residual connection leads to the following:

$$\begin{aligned}\boldsymbol{P}\boldsymbol{x}_f^r &= \boldsymbol{P}\left(\boldsymbol{W}_2\text{ReLU}(\boldsymbol{W}_1\boldsymbol{x}_a^r) + \boldsymbol{x}_a^r\right) \\ &= \boldsymbol{P}\boldsymbol{W}_2\text{ReLU}(\boldsymbol{W}_1\boldsymbol{x}_a^r) + \boldsymbol{P}\boldsymbol{x}_a^r \\ &= \boldsymbol{P}\boldsymbol{W}_2\text{ReLU}(\boldsymbol{W}_1\boldsymbol{x}_a^r) + \boldsymbol{P}(\boldsymbol{W}_O\text{MHA}(\boldsymbol{x}) + \boldsymbol{x}). \end{aligned} \tag{6}$$

We see that due to the residual structure, any permutation applied to second feed-forward weight parameter, $\boldsymbol{W}_2$, must also be applied to MHA, or more specifically the $\boldsymbol{W}_O$ matrix. To unpermute these features, we apply $\boldsymbol{P}^{\mathrm{T}}$ to where the permuted $\boldsymbol{x}_a^r$ and $\boldsymbol{x}_f^r$ states become inputs, which are $\boldsymbol{W}_1$ and $\{\boldsymbol{W}_Q, \boldsymbol{W}_V, \boldsymbol{W}_K\}$, respectively.

Because the input to each layer must be permuted $\boldsymbol{P}\boldsymbol{x}$, and the output of each layer is also permuted $\boldsymbol{P}\boldsymbol{x}_f^r$, we can see that the entire Transformer architecture uses the same $\{\boldsymbol{P}, \boldsymbol{P}^{\mathrm{T}}\}$ matrices for all the weights involved in residual connections. This is unlike our other proposed permutations for multi-headed attention which are specific to each Transformer layer. At the ends of the models, namely the embedding layer(s) and output layer(s), we also apply these transformations, as the input to the first Transformer block and the output of the last Transformer block are permuted. We apply this $\boldsymbol{P}$ to the embedding weights, including positional and any special token embeddings, and at the final layer, we apply $\boldsymbol{P}^{\mathrm{T}}$ to the weight matrix immediately following the last Transformer block LayerNorm. This is usually a pooling or dense layer, depending on the model task.

Because of the multiple potential features, that could contribute to the computation of the residual permutations, from both $\boldsymbol{x}_a^r$ and $\boldsymbol{x}_f^r$ across all layers, we use features from all $\boldsymbol{x}_f^r$, and $\boldsymbol{x}_a^r$ in the forward pass.