Towards Structured Sparsity in Transformers for Efficient Inference

Harry Dong [1] **Beidi Chen** [1] **Yuejie Chi** [1]

Abstract

Transformer models have been critical in accelerating progress in numerous fields, yet scaling these models come at high computational costs. In this paper, we explore sparsity properties in transformers and manipulate existing sparsity in transformers to be more structured for efficient training and inference. In particular, we create sparse structures that have inter-layer similarity and are block sparse which have the potential to bypass a significant amount of model loading and computation. We present preliminary results and ideas using a small transformer which we hope to extend to more complex models.

1. Introduction

Though initially designed in the context of natural language processing, transformers (Vaswani et al., 2017) have proven to be essential assets to numerous domains, such as computer vision (Dosovitskiy et al., 2020; Khan et al., 2022) and bioinformatics (Zhang et al., 2023). The rise of transformer-based large language models (LLMs) such as GPT-4 (OpenAI, 2023), LLaMA (Touvron et al., 2023), and LaMDA (Thoppilan et al., 2022), has constantly pushed the state-of-the-art performance in various tasks, but they come with heavy computational and memory burdens. In this paper, we induce different sparse structures in transformers such that they can be leveraged to make transformers more efficient, both in terms of FLOPs and wall clock time. Focusing on transformer feedforward (FF) layers, we present transformers that have shared sparsity patterns across layers and block sparse activations with negligible performance degradation.

Model sparsity has been frequently studied in the context of model pruning where unimportant neurons are removed. However, many of such methods require repeatedly pruning and retraining the model to minimize performance degradation (Blalock et al., 2020), which is impractical for LLMs. Pruning methods exist for transformers (Frantar & Alistarh, 2023; Chen et al., 2021), but they produce unstructured sparsity or are domain specific.

Intriguingly, trained transformer-based models naturally contain sparse structures in their FF activations without explicit regularizations or constraints (Zhang et al., 2021; Li et al., 2022). There has been work to obtain a firmer grasp of the cause (Geva et al., 2020), yet it is still lacking. Nonetheless, understanding this phenomenon is crucial to exploiting it for transformer efficiency. Some initiatives to do so involve turning FF layers into a mixture of experts (Zhang et al., 2021; Fedus et al., 2022) and setting a top-k threshold in the FF activation functions (Li et al., 2022). Getting inspiration from these ideas, we hope to create more structured and hardware-aware sparse activations which could lead to even larger leaps in transformer efficiency.

Notation. In this paper, we define scalars, row vectors, and matrices using capital letters (e.g. A), lowercase bold letters (e.g. $\mathbf{a}$), and uppercase bold letters (e.g. $\mathbf{A}$), respectively.

Brief Background on FF layers. Vanilla transformers consist of alternating attention layers and FF layers which all sit in between an embedding layer and output layer. The FF layers are typically neural networks with a single hidden layer of dimension scaled by F where in many models, $F = 4$. More formally, for $\mathbf{x}, \mathbf{b}_2 \in \mathbb{R}^D$, $\mathbf{b}_1 \in \mathbb{R}^{F \cdot D}$, $\mathbf{W}_1 \in \mathbb{R}^{D \times F \cdot D}$, $\mathbf{W}_2 \in \mathbb{R}^{F \cdot D \times D}$, and activation function $\sigma(\cdot)$, each FF layer produces $\mathbf{z} \in \mathbb{R}^D$ by evaluating

$$\mathbf{a} = \sigma(\mathbf{x}\mathbf{W}_1 + \mathbf{b}_1), \tag{1}$$

$$\mathbf{z} = \mathbf{a}\mathbf{W}_2 + \mathbf{b}_2. \tag{2}$$

We call the first and second linear operations FF1 and FF2, respectively. Sparsity is expressed as the fraction of nonzero elements in a data structure, so a lower value is preferable in our setting. When we discuss activation sparsity, we are referring to the sparsity in $\mathbf{a} \in \mathbb{R}^{F \cdot D}$.

2. Inherent Sparsity in Transformers

Before discussing ways to create structured sparsity, we first seek to better understand the circumstances in which sparsity naturally manifests in transformers and if it affects performance. By adjusting the activation function in FF

[1]Department of Electrical and Computer Engineering, Carnegie Mellon University, USA. Correspondence to: Harry Dong <harryd@andrew.cmu.edu>.

Work presented at the ES-FoMo Workshop at ICML 2023, Honolulu, Hawaii, USA.

layers, we demonstrate that sparsity is not unique to any activation function, broadening our methods' applicability to a wide class of functions.

The following experiments use the same training procedure and variations of the same model, a 6 layer GPT-2 transformer (Radford et al., 2019) trained on WikiText-103 (Merity et al., 2016) for causal language modeling evaluated using validation perplexity (PPL). We use a hidden dimension of 768, $F = 4$, 12 attention heads, and a dropout rate of 0.1. All experiments are run for 15 epochs via Adam with a cosine scheduler that warms up to a learning rate of 1e-4 for the first 5% of gradient steps using a batch size of 32, each sample having length 1024. Weight tying (WT) is applied to the embedding layer and linear head.

2.1. Exploration on Activation Functions

While many models (including the original GPT-2) default to using Gaussian error linear unit (GELU) activation functions in FF layers, it makes studying sparsity difficult since the GELU function only equals 0 at a single point, rather than a region like rectified linear units (ReLUs). We note that in practice, GELU activations can approach 0 asymptotically as the pre-activation value becomes more negative which has been observed to occur at great intensities for larger transformers (Dettmers et al., 2022). This can be seen as a form of approximate sparsity where most elements in $\mathbf{a}$ have very small magnitudes. Hence, while we focus on ReLUs for this paper, we are optimistic that these principles can transfer to the approximate sparsity observed in GELUs and, broadly, other 0-saturating functions. In this section, we begin by studying the effect of different activations functions on the performance and activation sparsity. For $\mathbf{x} \in \mathbb{R}^D$, define the following element-wise activations:

$$\text{FlipReLU}^k(\mathbf{x}) = \begin{cases} \text{ReLU}(\mathbf{x}) & \text{layer } k \text{ is even} \\ -\text{ReLU}(-\mathbf{x}) & \text{layer } k \text{ is odd} \end{cases},$$

$$\text{SplitReLU}(\mathbf{x}) = \left[\text{ReLU}(\mathbf{x}_{1:\frac{D}{2}}), \quad -\text{ReLU}(-\mathbf{x}_{\frac{D}{2}+1:D})\right].$$

FlipReLU alternates between ReLU and the negative version of it from layer to layer; SplitReLU applies ReLU on half of a vector and the negative version to the remaining half. While these provide equivalent representational power as ReLUs in a model, we seek to observe the robustness of sparsity emergence. Results are shown in Table 1.

The choice of activation appears to matter very little regarding performance since the validation PPL of all models with the FF layers are approximately the same. However, activation functions that have a region that saturates at 0 have the benefit of sparsity. Despite using different activation functions besides GELU, the final activation sparsity levels are all approximately the same. This opens two avenues to explore: (i) the emergence of activation sparsity does not depend on the zero-saturating activation function deployed and (ii) the study of activation sparsity is justified as there is negligible performance loss when using a zero-saturating activation function. The first suggests, counterintuitively, directing our focus away from the activation function when understanding the cause of activation sparsity; the second opens the opportunity to manipulate this sparsity into a more desirable form, the focus for the remainder of this paper.

Table 1. Validation PPL and average activation sparsity across all validation inputs of models trained on different activation functions.

ACTIVATION	VAL. PPL	SPARSITY
GELU	22.60	1.000
RELU	23.14	0.109
SPLITRELU	23.11	0.112
FLIPRELU	23.03	0.112

2.2. Baseline Sparsity

The baseline we use is the 6-layer GPT-2 model with ReLU activation functions described previously. Similar to previous observations in LLMs (Zhang et al., 2021; Li et al., 2022), our small baseline also exhibits sparse activations that are very unevenly distributed. Figure 1 reveals that nearly all neurons in all layers are rarely activated, aside from a handful of neurons which are frequently activated. For the activation frequencies for all 6 layers, see Appendix A. The goal is to morph this natural sparsity into more structured patterns that can be exploited for efficiency.

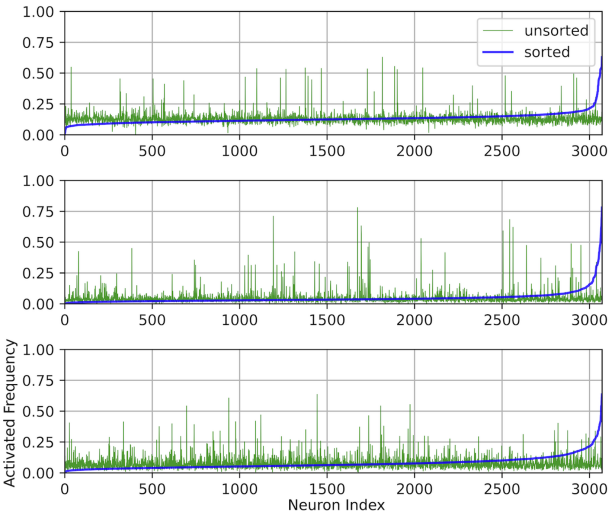

Figure 1. Sorted and unsorted neuron activation frequencies of the baseline in layers 1 to 3, from top to bottom, across all validation inputs.

3. Creating Structured Sparsity

Since we have shown activation sparsity is essentially free, we now try to manipulate this sparsity into more useful

forms to improve transformer efficiency.

3.1. Inter-layer Sparsity Patterns

Typically, WT is applied by tying the weights of the embedding matrix and linear head of language models (Inan et al., 2016; Press & Wolf, 2016), as we have done to our baseline. Extending this idea, we also tie the weights of each FF layer across all layers. In other words, all FF1s have the same weights and similarly for the FF2s. Bias terms are not tied. This is to encourage shared representations between layers.

WT greatly reduces the number of learnable parameters, which allows some flexibility to add more. The baseline sets $F = 4$ and has no WT between FF layers. We test $F = 4, 24$ with WT, in which the $F = 24$ model has roughly the same number of learnable parameters as the baseline with the exception of some bias terms. Parameter counts, validation PPLs, and sparsities of the models are displayed in Table 2.

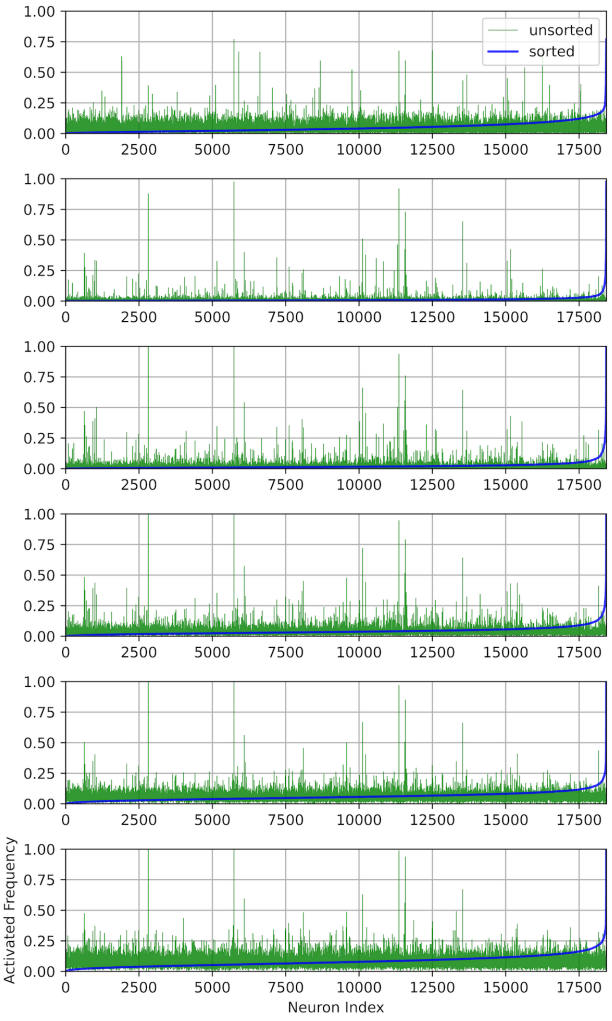

Figure 2. Sorted and unsorted neuron activation frequencies of the model with WT and $F = 24$ in layers 1 to 6, from top to bottom, across all validation inputs.

Table 2. Validation PPL, parameter count, and mean validation sparsity of each model. The first row is the baseline.

FF WT	F	VAL. PPL	PARAMS (M)	SPARSITY
×	4	23.14	81.9	0.109
✓	4	27.19	58.3	0.106
✓	24	22.23	82.0	0.043

The model with WT and expansion factor of 24 produced a slightly better validation PPL compared to the baseline, implying that transformer depth can be traded for width. Interestingly, this model can represent the baseline by just activating $\frac{1}{6}$ of the FF layer at each layer, but that is not what happens. Interestingly, unsorted frequencies in Figure 2 show that similar sparsity behaviors exist even when FF weights are tied, and some heavy-hitting neurons are shared across layers. For instance, all layers shared the same most frequently activated neuron. This type of inter-layer activation sparsity structure arises naturally without cost in PPL or number of parameters. This relationship is also reflected when we look at the similarity between activated neurons between layers in Figure 3.

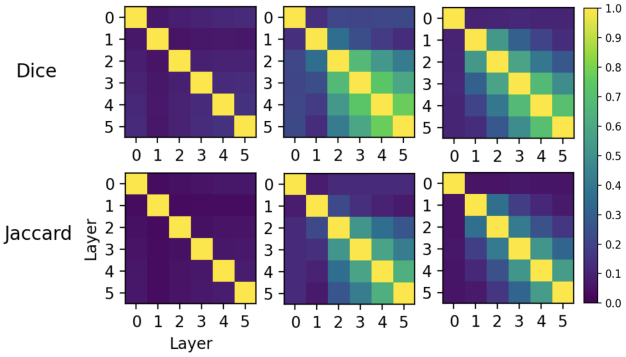

Figure 3. Activation similarity between layers as measured by the Dice coefficient and Jaccard index. From left to right, the columns are results from the baseline, FF WT with $F = 4$, and FF WT with $F = 24$.

3.2. Block Sparsity

Now, we shift our focus to block sparsity, which is of great interest in terms of efficiency from a systems perspective because it has the potential to sidestep a large amount of memory loading. We define a block as a consecutive group of neurons in a FF layer, and a block is activated if at least one of its neurons is activated. When a FF layer is completely partitioned into blocks, we can define the block sparsity as the sum of the fraction of the current layer's neurons contained in each activated block. In general, even if an FF layer is sparse, it becomes less likely that it is block sparse as the block size increases. For simplicity, we restrict each block to be the same size.

Table 3. Block penalization ablation study on WikiText-103. Sparsity and block sparsity levels are averaged across all validation inputs into all FF layers.

λ	EXEMPT	VAL. PPL	SPARSITY	B. SPARSITY
0	N/A	23.14	0.109	0.971
5E-4	0	23.20	0.012	0.362
5E-4	128	23.33	0.016	0.352
5E-4	256	23.34	0.020	0.362
1E-3	0	24.48	0.005	0.204
1E-3	128	24.50	0.011	0.203
1E-3	256	24.26	0.016	0.219

One method of encouraging block sparsity is regularization. For a length S input, define $\mathbf{a}_k \in \mathbb{R}^{S\times F\cdot D}$ to be the FF activation output at layer k and $\mathbf{a}_{1:K}$ to be the activations for all K layers. For an arbitrary $\mathbf{M} \in \mathbb{R}^{M\times N}$, denote $\mathcal{B}(\mathbf{M}, B) \in \mathbb{R}^{M\times \frac{N}{B}\times B}$ to be the operator that reshapes $\mathbf{M}$ with a valid B. Next, let $\mathbf{y}$ and $\hat{\mathbf{y}}$ be the target and model output, respectively, which are fed into an unregularized loss function $\mathcal{L}(\hat{\mathbf{y}}, \mathbf{y})$. Then, the new loss function $\mathcal{L}_b$ is

$$\mathcal{L}_b(\hat{\mathbf{y}}, \mathbf{y}, \mathbf{a}_{1:K}) \coloneqq \mathcal{L}(\hat{\mathbf{y}}, \mathbf{y}) + \frac{\lambda B}{FD} \sum_{i,j,k} \|\mathcal{B}(\mathbf{a}_k, B)_{i,j}\|_2 \tag{3}$$

for tunable parameters λ and B. The regularization term is an ℓ_1 penalization on the magnitude of each block. The $\frac{B}{FD}$ term averages the summation over j, but this can be absorbed into λ.

We compare performance and block sparsity levels as we vary λ keeping $B = 64$. This value of B is large enough to obtain nontrivial blocks but also small enough for adequate sparsity to emerge. Because some neurons will be activated very frequently, we can select a group of neurons that are exempt from regularization. This group can be treated as its own block that will be always activated. The motivation is that the frequently activated neurons will be encouraged to be concentrated among the exempt neurons, and the regularized blocks will mostly contain infrequently activated neurons. Comparisons between setting the first 0, 128, or 256 neurons to be exempt are included in Table 3. Example block activations can be found in Figure 4 and Appendix B.

Our regularized loss function, Equation (3), is successful in creating block sparsity while also making the activations even more sparse. In fact, imposing this structure has little effect on the performance, though as expected, increasing λ creates greater sparsity at the cost of performance. The number of exempt neurons had no noticeable effect on the block sparsity and performance, but based on Figure 5, this did concentrate the heavy hitting neurons. See Appendix A.1 for more examples.

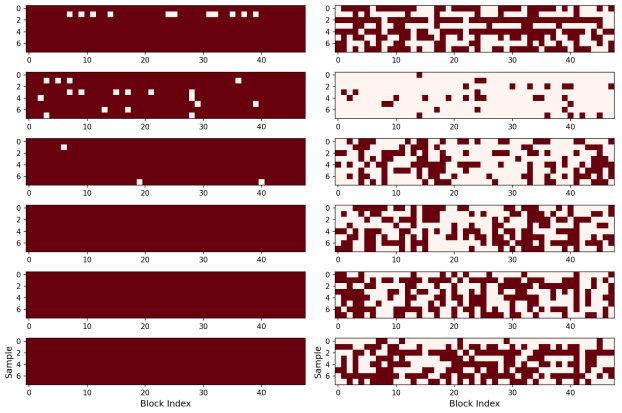

Figure 4. Activated blocks of the same 8 random validation inputs passed into layers 1 to 6 (top to bottom) of the baseline (left) and $\lambda = 5\text{e-}4$ with 0 exempt neurons (right) models. Dark red squares are activated blocks, and the x-axis and y-axis are the block indices and sample indices, respectively.

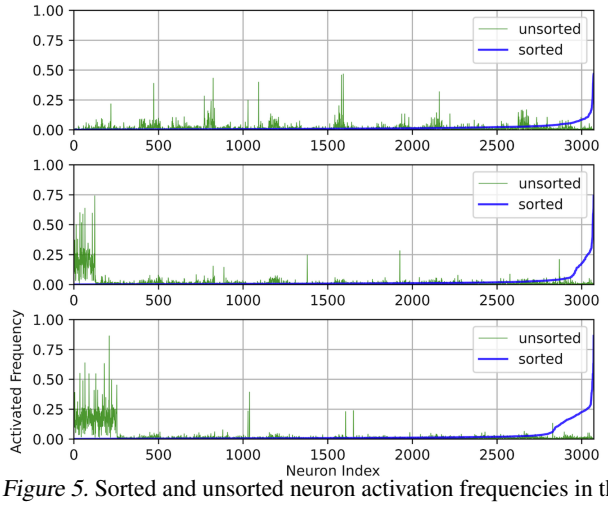

Figure 5. Sorted and unsorted neuron activation frequencies in the first layer of the $\lambda = 5\text{e-}4$ models with 0 (top), 128 (center), and 256 (bottom) exempt neurons across all validation inputs.

4. Future Work

Transformer activation sparsity is a natural phenomenon that is more malleable than it may seem. We have shown that we can induce two types of structured sparsity in FF layers: sparse inter-layer similar activations and block sparse activations. One future direction is to use these structures to compress and accelerate transformers during training and inference. For example, block sparsity could be used bypass loading significant parts of a model when dealing with small batches or skip computations in the FF layer. Additionally, more exploration could be done such as creating other structures to further understand how these sparse structures arise. It would also be interesting to investigate if this naturally occurring sparsity could be destroyed while preserving performance and the properties that are gained and lost by doing so. Furthermore, we hope to also scale these directions and results to much larger transformer models.

References

Blalock, D., Gonzalez Ortiz, J. J., Frankle, J., and Guttag, J. What is the state of neural network pruning? *Proceedings of machine learning and systems*, 2:129–146, 2020.

Chen, T., Cheng, Y., Gan, Z., Yuan, L., Zhang, L., and Wang, Z. Chasing sparsity in vision transformers: An end-to-end exploration. *Advances in Neural Information Processing Systems*, 34:19974–19988, 2021.

Dettmers, T., Lewis, M., Belkada, Y., and Zettlemoyer, L. Llm. int8 (): 8-bit matrix multiplication for transformers at scale. *arXiv preprint arXiv:2208.07339*, 2022.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Fedus, W., Zoph, B., and Shazeer, N. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *The Journal of Machine Learning Research*, 23(1):5232–5270, 2022.

Frantar, E. and Alistarh, D. Massive language models can be accurately pruned in one-shot. *arXiv preprint arXiv:2301.00774*, 2023.

Geva, M., Schuster, R., Berant, J., and Levy, O. Transformer feed-forward layers are key-value memories. *arXiv preprint arXiv:2012.14913*, 2020.

Inan, H., Khosravi, K., and Socher, R. Tying word vectors and word classifiers: A loss framework for language modeling. *arXiv preprint arXiv:1611.01462*, 2016.

Khan, S., Naseer, M., Hayat, M., Zamir, S. W., Khan, F. S., and Shah, M. Transformers in vision: A survey. *ACM computing surveys (CSUR)*, 54(10s):1–41, 2022.

Li, Z., You, C., Bhojanapalli, S., Li, D., Rawat, A. S., Reddi, S. J., Ye, K., Chern, F., Yu, F., Guo, R., et al. Large models are parsimonious learners: Activation sparsity in trained transformers. *arXiv preprint arXiv:2210.06313*, 2022.

Merity, S., Xiong, C., Bradbury, J., and Socher, R. Pointer sentinel mixture models, 2016.

OpenAI. Gpt-4 technical report, 2023.

Press, O. and Wolf, L. Using the output embedding to improve language models. *arXiv preprint arXiv:1608.05859*, 2016.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Thoppilan, R., Freitas, D. D., Hall, J., Shazeer, N., Kulshreshtha, A., Cheng, H.-T., Jin, A., Bos, T., Baker, L., Du, Y., Li, Y., Lee, H., Zheng, H. S., Ghafouri, A., Menegali, M., Huang, Y., Krikun, M., Lepikhin, D., Qin, J., Chen, D., Xu, Y., Chen, Z., Roberts, A., Bosma, M., Zhao, V., Zhou, Y., Chang, C.-C., Krivokon, I., Rusch, W., Pickett, M., Srinivasan, P., Man, L., Meier-Hellstern, K., Morris, M. R., Doshi, T., Santos, R. D., Duke, T., Soraker, J., Zevenbergen, B., Prabhakaran, V., Diaz, M., Hutchinson, B., Olson, K., Molina, A., Hoffman-John, E., Lee, J., Aroyo, L., Rajakumar, R., Butryna, A., Lamm, M., Kuzmina, V., Fenton, J., Cohen, A., Bernstein, R., Kurzweil, R., Aguera-Arcas, B., Cui, C., Croak, M., Chi, E., and Le, Q. Lamda: Language models for dialog applications, 2022.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., and Lample, G. Llama: Open and efficient foundation language models, 2023.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Zhang, S., Fan, R., Liu, Y., Chen, S., Liu, Q., and Zeng, W. Applications of transformer-based language models in bioinformatics: A survey. *Bioinformatics Advances*, 2023.

Zhang, Z., Lin, Y., Liu, Z., Li, P., Sun, M., and Zhou, J. Moefication: Transformer feed-forward layers are mixtures of experts. *arXiv preprint arXiv:2110.01786*, 2021.

A. Neuron Activation Frequencies

Neuron activation frequency plots of the model with WT and $F = 24$ are in the main text as Figure 2. The plots for the baseline and model with WT and $F = 4$ are shown in Figure 6 and Figure 7, repsectively.

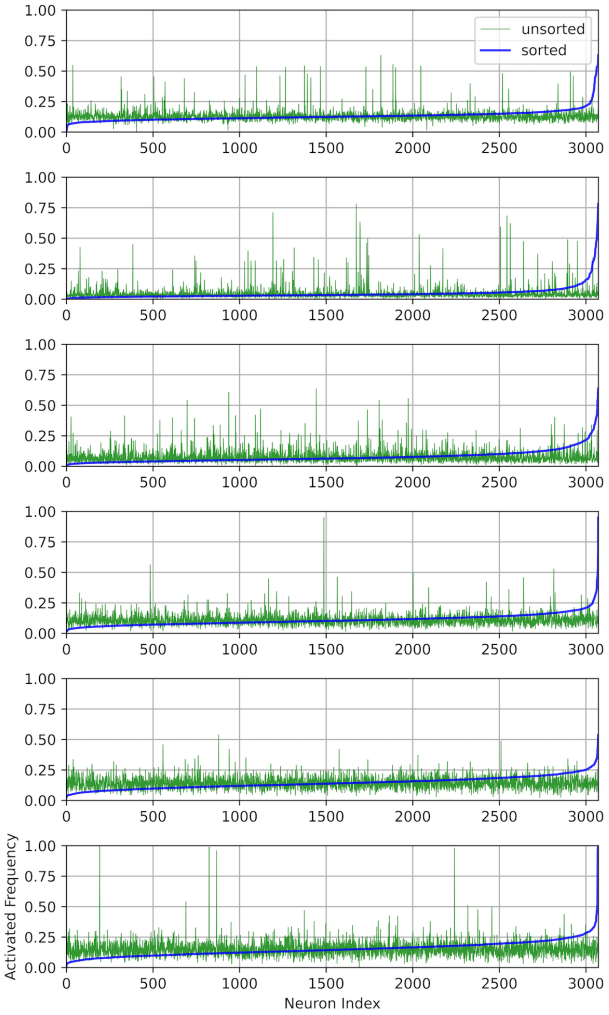

Figure 6. Sorted and unsorted neuron activation frequencies of the baseline in layers 1 to 6, from top to bottom, across all validation inputs.

A.1. Neuron Activation Frequencies for Block Sparse Layers

Figure 8, Figure 9, and Figure 10 show how the block sparsity regularizer in Equation (3) and number of exempt neurons affect the activation frequency distribution.

Regularizing for block sparsity dramatically reduces the frequency of activations for all neurons, compared to Figure 6. However, the same uneven distribution of activations is present. By increasing the number of exempt neurons, we are able to shove most of the frequently activated neurons there, reducing the activation frequency of all other neurons.

B. Block Activations

In Figure 11, Figure 12, and Figure 13, we show example block sparsity maps using the same samples.

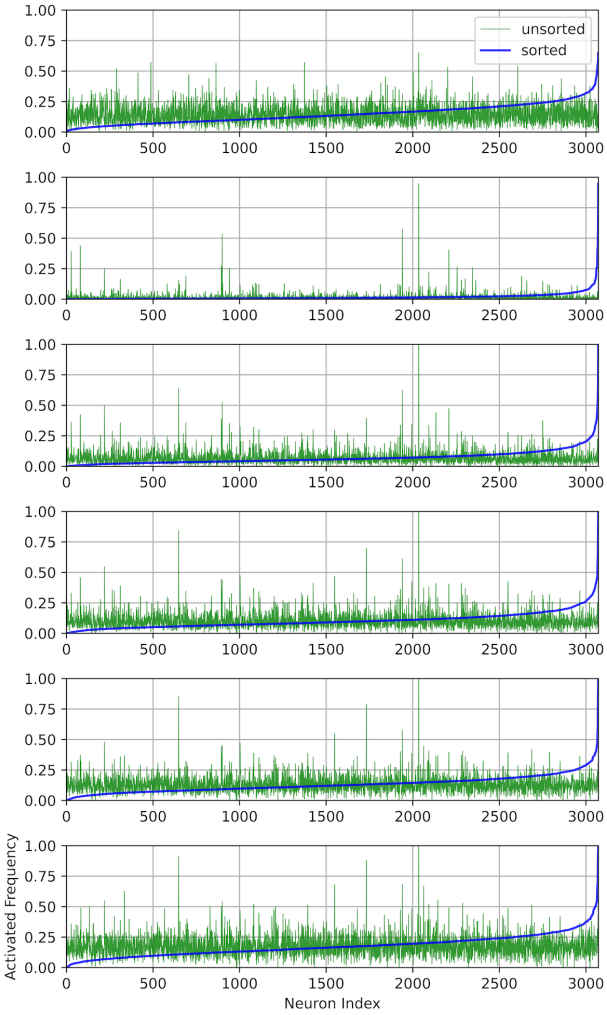

Figure 7. Sorted and unsorted neuron activation frequencies of the model with WT and $F = 4$ in layers 1 to 6, from top to bottom, across all validation inputs.

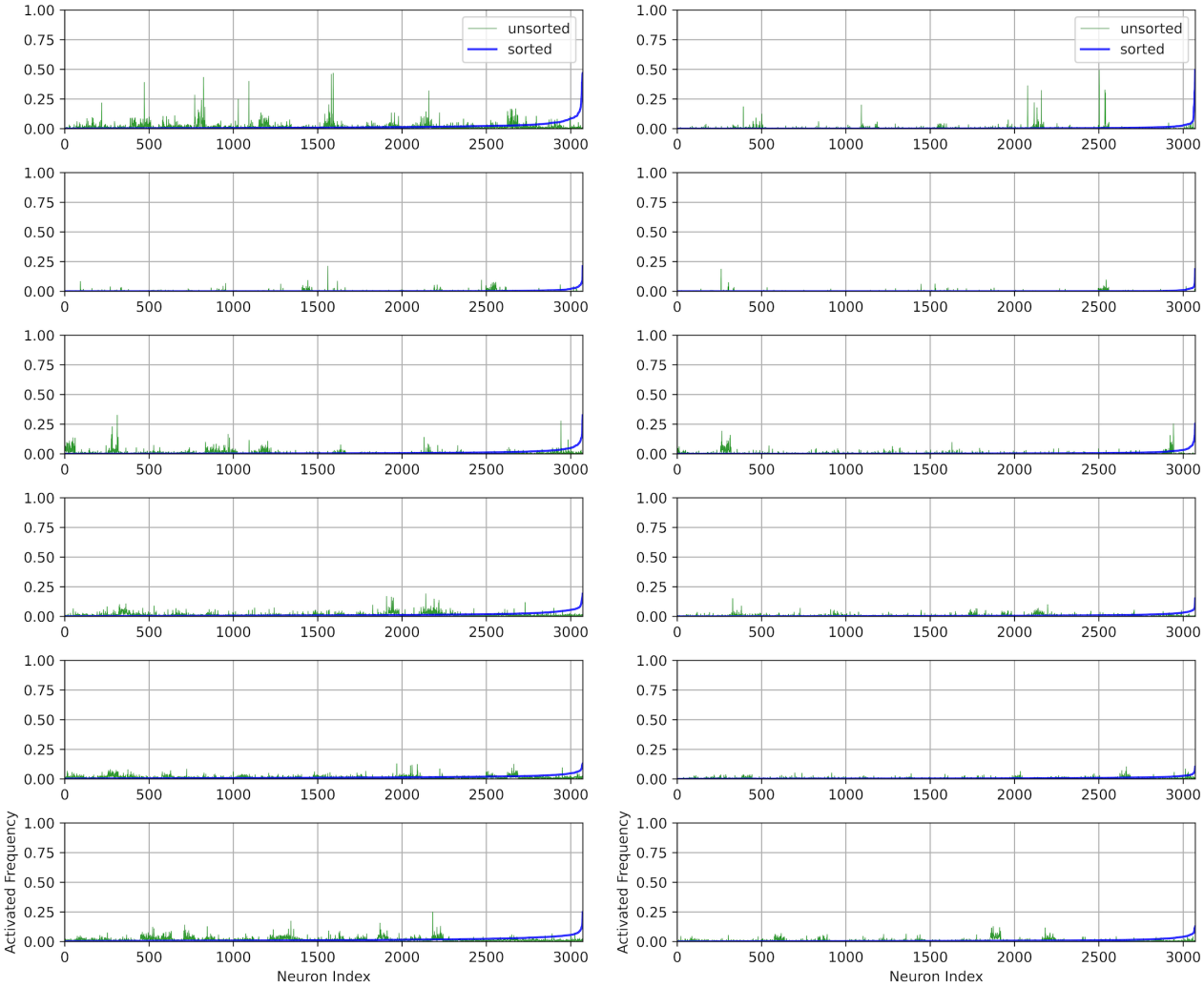

Figure 8. Sorted and unsorted neuron activation frequencies of the models with $\lambda =$ 5e-4 (left) and $\lambda =$ 1e-3 (right) in layers 1 to 6, from top to bottom, across all validation inputs. Both models have no exempt neurons.

Figure 9. Sorted and unsorted neuron activation frequencies of the models with $\lambda =$ 5e-4 (left) and $\lambda =$ 1e-3 (right) in layers 1 to 6, from top to bottom, across all validation inputs. Both models have 128 exempt neurons.

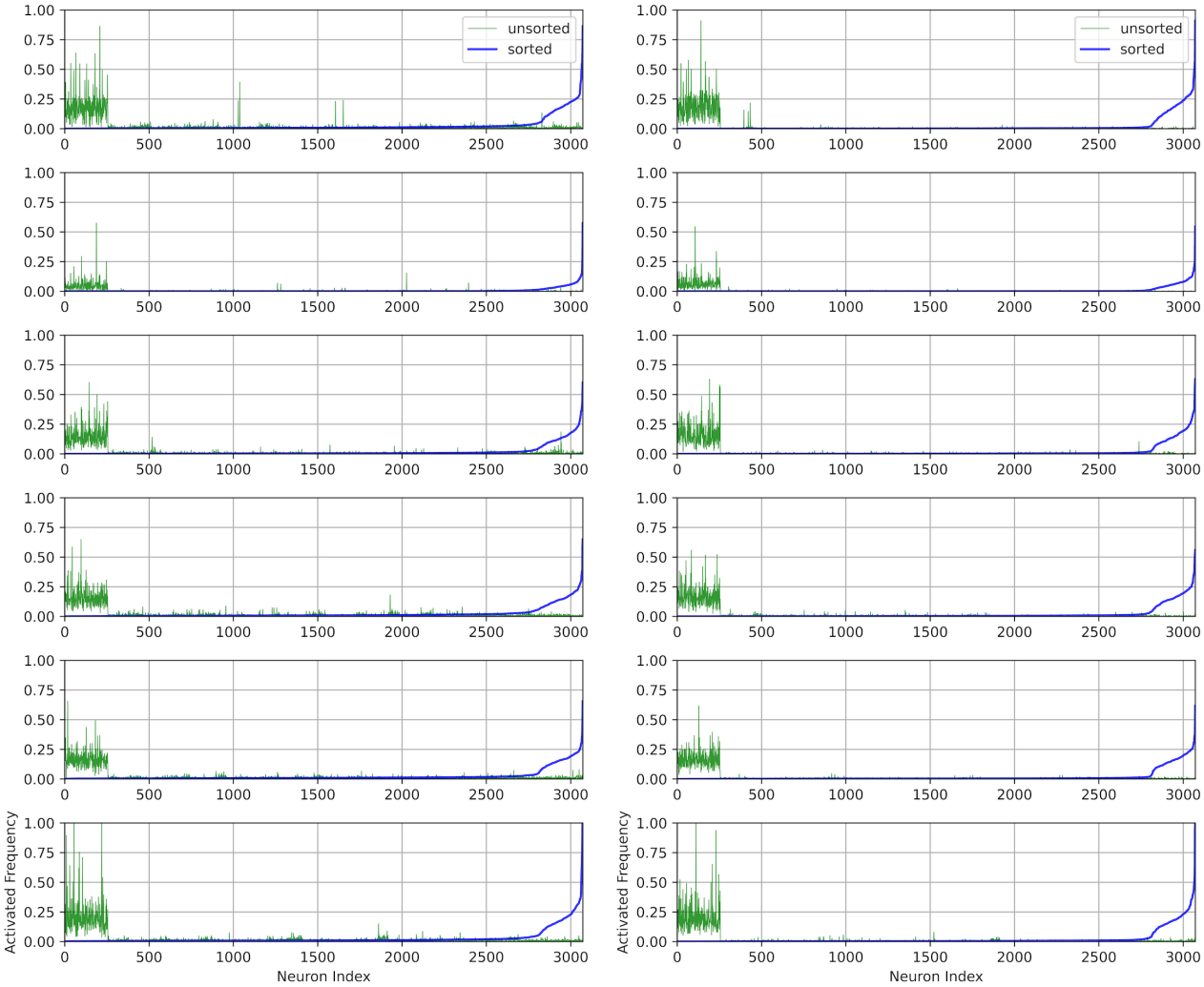

Figure 10. Sorted and unsorted neuron activation frequencies of the models with $\lambda =$ 5e-4 (left) and $\lambda =$ 1e-3 (right) in layers 1 to 6, from top to bottom, across all validation inputs. Both models have 256 exempt neurons.

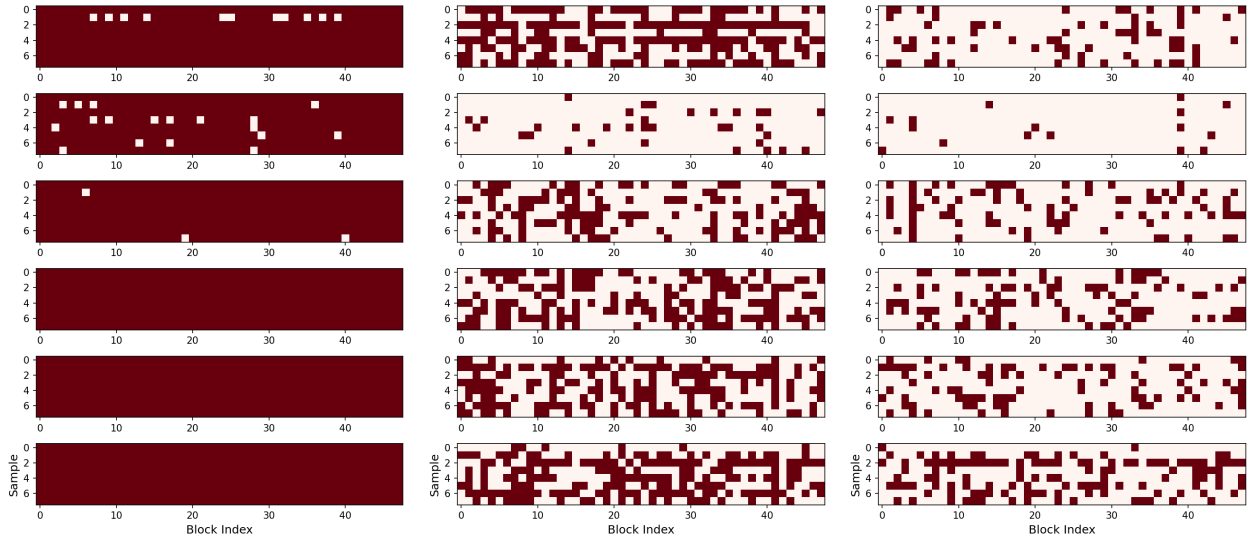

Figure 11. Activated blocks of the same 8 random validation inputs passed into layers 1 to 6 (top to bottom) of the baseline (left), $\lambda =$ 5e-4 (center), and $\lambda =$ 1e-3 (right) models. Dark red squares are activated blocks.

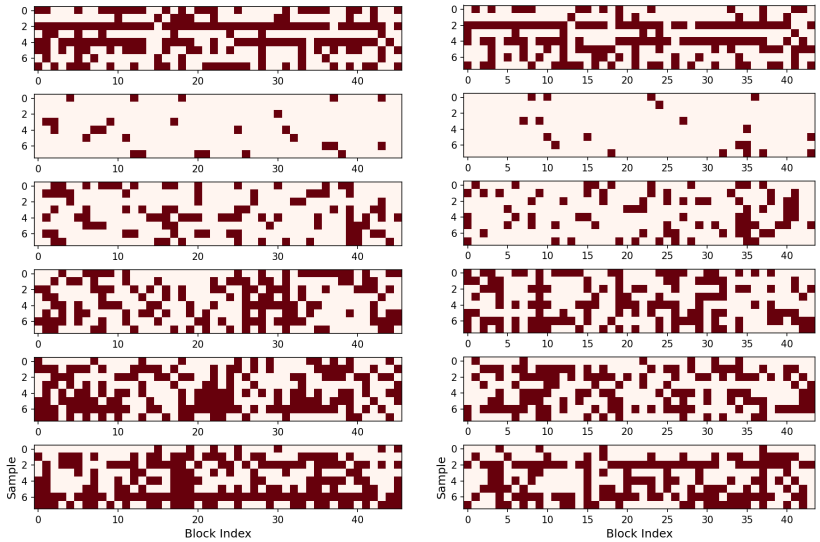

Figure 12. Activated blocks of the same 8 random validation inputs passed into layers 1 to 6 (top to bottom) of the models with 128 exempt neurons (left) and 256 exempt neurons (right), fixing $\lambda =$ 5e-4. Dark red squares are activated blocks.

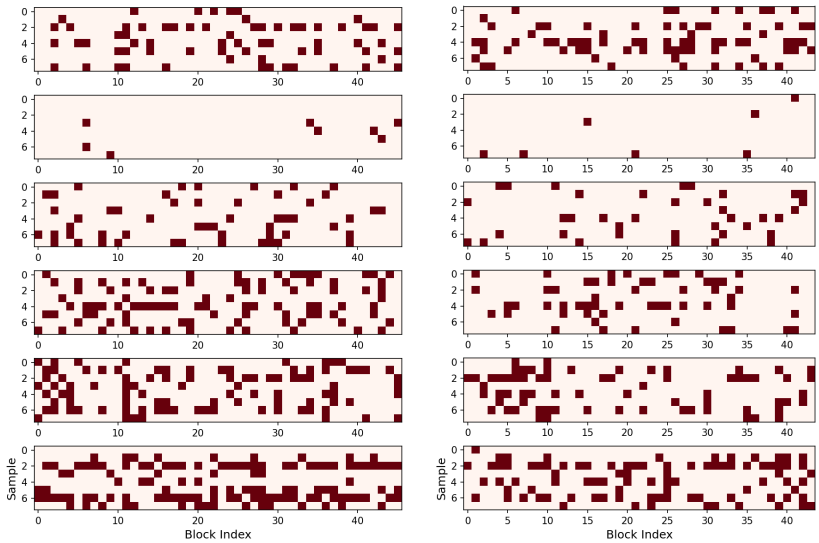

Figure 13. Activated blocks of the same 8 random validation inputs passed into layers 1 to 6 (top to bottom) of the models with 128 exempt neurons (left) and 256 exempt neurons (right), fixing $\lambda =$ 1e-3. Dark red squares are activated blocks.