Prompt-prompted Adaptive Structured Pruning for Efficient LLM Generation

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

Abstract

Large language models (LLMs) have remarkable utility, but this comes at a considerable computational cost at deployment. Fortunately, some methods such as pruning or mixture of experts exploit sparsity in transformer feedforward (FF) blocks to gain boosts in speed and reduce memory, yet these techniques can be costly and inflexible in practice, as they often require training or are restricted to specific types of architectures. To address this, we introduce GRIFFIN, a novel *training-free* method that selects unique FF experts at the sequence level for efficient generation across *a plethora of LLMs with different non-ReLU activation functions*. This is possible due to a critical observation that many trained LLMs naturally produce highly structured FF activation patterns within a sequence, which we call *flocking*. GRIFFIN maintains the original model's performance with little to no degradation on a variety of tasks, all while improving latency (e.g. 1.29× and 1.25× speed-ups in Gemma 7B and Llama 2 13B, respectively, on an NVIDIA L40). Code can be found at `https://github.com/hdong920/GRIFFIN`.

1. Introduction

Transformers (Vaswani et al., 2017) have demonstrated incredible capabilities across a plethora of domains (Lin et al., 2022; Khan et al., 2022; Nerella et al., 2023). Their large language model (LLM) successors (Touvron et al., 2023; Team et al., 2023; Jiang et al., 2023; 2024; Team et al., 2024; Anthropic, 2024) have pushed the bar higher, but these behemoths have become performative at the price of enormous amounts of computation and memory demands. One significant contributor is the model size itself. Not only is storage an issue, model layers tend to be wide and plenty, slowing down inference. Moreover, given the existence of sparse structures in LLMs, especially in feedforward (FF) blocks (Geva et al., 2020; Dettmers et al., 2022; Li et al., 2022; Liu et al., 2023), these models waste computation on intermediate features with little to no impact on the final result. For instance, it has been observed that in OPT-175B (Zhang et al., 2022), fewer than 5% of neurons in FF blocks have nonzero values per token (Liu et al., 2023), meaning 95% of the compute in each FF block is wasted. Usually consisting of around two-thirds of the parameters in an LLM, FF blocks can be serious memory and compute bottlenecks. These inefficiencies are problematic in latency-sensitive scenarios like in chatbots and autonomous vehicles.

There have been many methods to exploit sparsity in LLMs for efficiency gains, such as pruning (LeCun et al., 1989; Frankle & Carbin, 2018; Blalock et al., 2020; Liang et al., 2021; Liu et al., 2024; Frantar & Alistarh, 2023; Sun et al., 2023; Dery et al., 2024) and mixtures of experts (MoEs) (Jacobs et al., 1991; Zhang et al., 2021; Liu et al., 2023; Piórczyński et al., 2023; Csordás et al., 2023; Yerram et al., 2024; Zheng et al., 2024). Pruning removes low-impact pre-trained weights to reduce storage, yet this often does not translate to real speed-ups in practice, unless the pruning is done in a hardware-friendly manner (Xia et al., 2022; Santacroce et al., 2023; Ma et al., 2023; Li et al., 2023; Xia et al., 2023) which typically causes greater performance deterioration. MoEs better preserve the original performance by adaptively selecting subsets of the model to use per input but also come with drawbacks. Unless the model has been trained in this fashion (Fedus et al., 2022; Jiang et al., 2024), it will need to learn a cheap yet effective gating function (expert selection mechanism) and sometimes even require full fine tuning. Perhaps an even bigger weakness of many of these methods is the limited ability to effectively carry over to pre-trained LLMs with non-ReLU activations. In summary, there are a few challenges: **1)** structured sparsity is difficult and costly to enforce; **2)** adaptive selection of model subsets need to be cheap; **3)** the method should work on various activation functions.

Daunting at first, these challenges become surmountable because of a simple observation of a phenomenon we call *flocking*, highly consistent sparse activations that persist throughout a sequence observed in many LLMs. *Flocking emerges in FF activations (inputs into the FF down pro-*

[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 2024, Vienna, Austria. Copyright 2024 by the author(s).

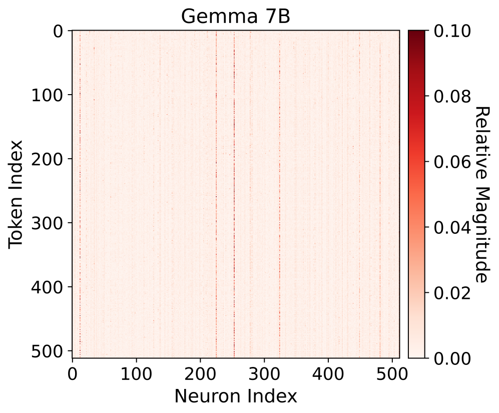

Figure 1. Relative FF activation magnitudes of the first 512 features and tokens across a sequence from PG-19 (Rae et al., 2019; Gao et al., 2020) in layer 10 of Gemma 7B. This shows flocking, where relative activation magnitudes are shared within a sequence.

jection) when we focus on a sequence's relative activation magnitudes instead of the absolute values. An example from Gemma 7B is shown in Figure 1, with more in Appendix E. *The key takeaway is that neurons which produce high relative magnitudes are naturally shared across tokens within a sequence*, as seen with the distinct vertical streaks. Increasingly bizarre, we observe this across different architectures and different non-ReLU activation functions.

Unlike existing pruning or MoE methods, we exploit flocking in our design of GRIFFIN (**G**ating by **R**epetition **I**n **F**eed**f**orward **I**ntermediate **N**eurons), *a highly performative and efficient training-free method to adaptively activate FF neurons*. GRIFFIN does this by using a sequence's prompt to determine the experts to activate during generation, allowing it to overcome all of the aforementioned challenges:

1. **No Preparation:** Our no-cost method is completely training-free and requires no preparation. Moreover, the simple implementation of GRIFFIN means it can be dropped into any FF block and instantly deployed.

2. **Simple Expert Selection:** Flocking in the prompt reveals the most relevant FF neurons for generation with little to no performance loss. The selection process is parameter-free and adds negligible overhead.

3. **Model & Activation Function Diversity:** Thorough experimentation demonstrates the efficacy of GRIFFIN on numerous models, including Llama 2 (Touvron et al., 2023), Gemma (Team et al., 2024), Mistral (Jiang et al., 2023), OPT (Zhang et al., 2022), and ReluLlama (Team, 2023). Together, the tested activation functions include ReLU, SwiGLU, GEGLU, and ReGLU.

In this paper, we show GRIFFIN is a simple and strong adaptive structured pruning method because of flocking. In more detail, we formalize the neuron selection problem and its motivation in Section 2. Then, we present our novel approach in Section 3.2, which requires an examination of the surprising phenomenon of flocking shared by many LLMs in Section 3.1. Our rigorous experiments demonstrate GRIFFIN preserves performance on classification and generation even after removing 50% of FF neurons (Section 4.1), all while having lower latency (Section 4.2). For instance, GRIFFIN reduces the number of active parameters in Llama 2 13B from 13B to 8.8B during generation to improve latency by 1.25× with almost no loss in performance.

2. Problem Formulation

This section sets up the problem we are tackling. Since FF blocks operate identically and independently for each token unlike attention, we begin with defining the FF block with a single column vector input $\boldsymbol{x} \in \mathbb{R}^D$:

$$\mathrm{FF}(\boldsymbol{x}) = \mathrm{FF}_2(\underbrace{\mathrm{FF}_1(\boldsymbol{x})}_{\boldsymbol{z}}) \tag{1}$$

where $\mathrm{FF}_2(\boldsymbol{z}) = \boldsymbol{W}_2\boldsymbol{z} + \boldsymbol{b}_2$ and FF_1 is nonlinear. This describes a wide range of FF architectures and arbitrary activation functions σ. For instance, in OPT,

$$\mathrm{FF}_1(\boldsymbol{x}) = \sigma(\boldsymbol{W}_1\boldsymbol{x} + \boldsymbol{b}_1). \tag{2}$$

For FF blocks with GLU variants such as in Llama 2,

$$\mathrm{FF}_1(\boldsymbol{x}) = \sigma(\boldsymbol{W}_g\boldsymbol{x} + \boldsymbol{b}_g) \odot (\boldsymbol{W}_1\boldsymbol{x} + \boldsymbol{b}_1) \tag{3}$$

where $\odot$ signifies element-wise multiplication. For all examples, $\boldsymbol{W}_1, \boldsymbol{W}_g \in \mathbb{R}^{D_{\mathrm{FF}} \times D}$ and $\boldsymbol{W}_2 \in \mathbb{R}^{D \times D_{\mathrm{FF}}}$ where typically, $D_{\mathrm{FF}} \gg D$. We refer to $\boldsymbol{z} = \mathrm{FF}_1(\boldsymbol{x})$ as the FF activations. The goal is to find $\widehat{\boldsymbol{W}}_1 \in \mathbb{R}^{k \times D}$, $\widehat{\boldsymbol{b}}_1 \in \mathbb{R}^k$, and $\widehat{\boldsymbol{W}}_2 \in \mathbb{R}^{D \times k}$ (additionally $\widehat{\boldsymbol{W}}_g \in \mathbb{R}^{k \times D}$ and $\widehat{\boldsymbol{b}}_g \in \mathbb{R}^k$ if needed) where $k < D_{\mathrm{FF}}$ such that when the FF block is reparameterized with these matrices, the output value is preserved. In other words, for

$$\widehat{\boldsymbol{z}} = \widehat{\mathrm{FF}}_1(\boldsymbol{x}) = \sigma(\widehat{\boldsymbol{W}}_g\boldsymbol{x} + \widehat{\boldsymbol{b}}_g) \odot (\widehat{\boldsymbol{W}}_1\boldsymbol{x} + \widehat{\boldsymbol{b}}_1), \tag{4}$$

$$\widehat{\mathrm{FF}}_2(\widehat{\boldsymbol{z}}) = \widehat{\boldsymbol{W}}_2\widehat{\boldsymbol{z}} + \boldsymbol{b}_2, \tag{5}$$

$\mathrm{FF}(\boldsymbol{x}) \approx \widehat{\mathrm{FF}}_2(\widehat{\mathrm{FF}}_1(\boldsymbol{x}))$, and similarly for FF blocks with non-GLU functions. Crucially, this selection leads to multiplication with smaller matrices which are naturally efficient on GPUs and TPUs (Fatahalian et al., 2004; Wang et al., 2019). For all equations defined in this section, they operate independently on each row of a length S sequence input $\boldsymbol{X} \in \mathbb{R}^{S \times D}$ (e.g. the activations for a sequence are $\boldsymbol{Z} = \mathrm{FF}_1(\boldsymbol{X}) \in \mathbb{R}^{S \times D_{\mathrm{FF}}}$).

3. From Flocking to GRIFFIN

Here, we take deeper dive into the phenomenon of flocking and describe the intuitive algorithm of GRIFFIN which is directly inspired by it.

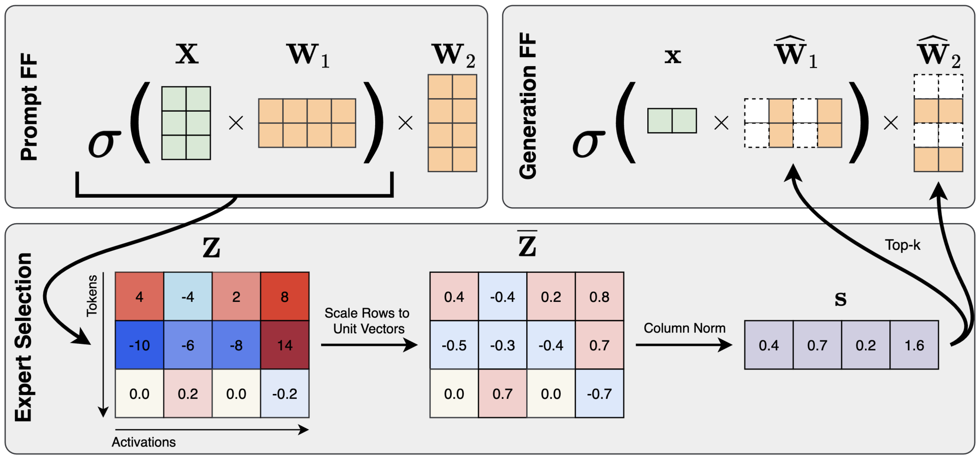

Figure 2. GRIFFIN overview. Relative activations from the prompt determine expert neurons to use for generation.

3.1. Observing Flocking

Observations of sparse activations in transformer FF blocks are not new (Geva et al., 2020; Dettmers et al., 2022; Li et al., 2022; Dong et al., 2023; Liu et al., 2023). In ReLU-based LLMs like OPT (Zhang et al., 2022), the activations can be exceptionally sparse and become more apparent for larger models (Liu et al., 2023). As more models use non-sparse activation functions like GLU variants (Shazeer, 2020), it is difficult for neurons to have no contribution to the output since these functions do not have an interval that maps to zero. Without exact sparsity, the efficacy of many sparsity-exploiting methods becomes limited. As such, this has ushered a wave of models that are either adapted from available models (Zhang et al., 2021; Liu et al., 2021; Mirzadeh et al., 2023; Zheng et al., 2024; Jiang et al., 2024) or trained from scratch (Fedus et al., 2022) which can produce activations that are exactly zero with little to no performance loss. Even so, these methods require considerable amounts of computational resources. *In contrast, we demonstrate many pretrained LLMs, including those with GLU networks, have naturally occuring structured sparsity in the form of flocking which we exploit.*

Flocking arises when we look at the relative impact of each neuron per token within a sequence. To see this, we normalize rows of $\boldsymbol{Z}$ to be unit vectors to construct $\overline{\boldsymbol{Z}} \in \mathbb{R}^{S \times D_{\text{FF}}}$ (i.e. $[\overline{\boldsymbol{Z}}]_i = [\boldsymbol{Z}]_i / \|[\boldsymbol{Z}]_i\|_2$), the *relative activations*. We show example relative activation magnitudes for a sequence in Llama 2 7B and Gemma 7B in Figure 1. Since there are distinct vertical streaks, this intriguingly implies that activations that have relatively greater weight are common across all tokens in a sequence. Notably, Llama 2 7B and Gemma 7B use SwiGLU and GEGLU activations, respectively, along with other major architecture differences. We call this phenomenon flocking, and we observe this in virtually all FF layers (see Appendix E). While relative activation magnitudes are shared within a sequence, they are not generally shared between sequences (Appendix A). *This lack of consistency implies pruning FF neurons without retraining would be less effective than an adaptive method.*

3.2. GRIFFIN Algorithm

Using our insight on flocking, we introduce GRIFFIN as a simple general purpose and training-free adaptive structured pruning method for efficient generation, captured in Figure 2. In a nutshell, we select experts during the prompt phase of each sample which are then used for the entire duration of the generation phase. This effective approach is based on a key observation on flocking: *since tokens within a sequence share activation patterns, the prompt and generated tokens will also share activation patterns.*

Prompt Phase Expert Selection. Our experts or neurons are chosen at the sequence level, so we need to consider the dynamics of the entire input sequence. To select expert neurons, we use a statistic $\boldsymbol{s} \in \mathbb{R}^{D_{\text{FF}}}$ to inform us of the importance of each neuron. At the prompt phase, we take the ℓ_2-norm of $\overline{\boldsymbol{Z}}$ along the token axis:

$$\boldsymbol{s} = \left[\|[\overline{\boldsymbol{Z}}]_{\cdot,1}\|_2 \quad \cdots \quad \|[\overline{\boldsymbol{Z}}]_{\cdot,D}\|_2 \right]^\top . \tag{6}$$

Taking the indices of the top-k across $\boldsymbol{s}$ gives us the neurons we will use for this sample's generation phase which make up the set $\mathcal{E}$. Using the experts in $\mathcal{E}$, we can find $\widehat{\boldsymbol{W}}_1, \widehat{\boldsymbol{b}}_1, \widehat{\boldsymbol{W}}_g, \widehat{\boldsymbol{b}}_g$, and $\widehat{\boldsymbol{W}}_2$ by selecting corresponding rows and columns in $\boldsymbol{W}_1, \boldsymbol{b}_1, \boldsymbol{W}_g, \boldsymbol{b}_g$, and $\boldsymbol{W}_2$, respectively. This is done for every FF block during the prompt phase. Elaborated in Appendix B, $\boldsymbol{s}$ highlights neurons consistently activated at relatively high intensities.

Generation with Experts. When generating tokens, we directly use the pruned layers which contain the expert neurons, $\widehat{\text{FF}}_1$ and $\widehat{\text{FF}}_2$, to estimate $\text{FF}(\boldsymbol{X}) \approx \widehat{\text{FF}}_2(\widehat{\text{FF}}_1(\boldsymbol{X}))$ for all future tokens. In Llama 2 13B and Gemma 7B, this reduces the active number of parameters from 13B to 8.8B and from 8.5B to 5.4B, respectively, during generation.

Table 1. XSum (1-shot Rouge-1), CNN/DailyMail (1-shot Rouge-1), CoQA (0-shot F1), SCROLLS QASPER (0-shot F1) at 50% FF sparsity.

Model	XSum	CNN	CoQA	QASPER
Llama 2 7B	27.15	10.08	77.35	26.31
Magnitude	9.71	9.66	56.59	12.93
GRIFFIN	24.75	10.97	77.18	25.76
Llama 2 13B	26.90	2.51	79.18	28.32
Magnitude	5.72	0.02	65.69	15.55
GRIFFIN	25.69	3.31	79.22	27.91
Gemma 7B	26.86	17.45	79.04	30.78
Magnitude	1.49	0.00	2.92	7.02
GRIFFIN	25.86	18.26	78.52	27.37
Mistral 7B	28.67	0.28	80.70	24.56
Magnitude	3.58	0.26	61.99	17.18
GRIFFIN	26.59	1.26	80.15	23.92
OPT 6.7B	23.60	13.85	68.70	18.53
Magnitude	1.63	1.20	31.53	7.28
GRIFFIN	21.17	13.01	68.99	17.40

Table 2. Generation phase latency (s). We denote "$P + G$" as the task of generating G tokens from a length P prompt. When relevant, times are in the format 50% FF sparsity.

Setup	Full	Magnitude	GRIFFIN
Llama 2 13B			
2048+128	6.8	5.4	5.4
2048+2048	119.1	95.0	94.9
Gemma 7B			
2048+128	4.5	4.1	4.2
2048+2048	78.5	67.7	67.4

4. Experiments

We showcase the superb performance of GRIFFIN on numerous tasks and models (Section 4.1) while achieving lower latency (Section 4.2). For more experiments covering some properties and ablations of GRIFFIN, see Appendix D. All experiments are run on NVIDIA L40 GPUs.

4.1. Performance

We evaluate various models on several generation and classification tasks. For generation tasks, we use XSum (Narayan et al., 2018), CNN/DailyMail (Nallapati et al., 2016), COQA (Reddy et al., 2019), and SCROLLS QASPER (Dasigi et al., 2021; Shaham et al., 2022). For classification, we evaluate on HellaSwag (Zellers et al., 2019), PIQA (Bisk et al., 2020), COPA (Roemmele et al., 2011), ARC-Easy/Challenge (Clark et al., 2018), and BoolQ (Clark et al., 2019). With the exception of XSum and CNN/DailyMail, we use LM Evaluation Harness for our experiments (Gao et al., 2023). Aside from comparing with the original LLM, we also compare GRIFFIN with a static sequence-level selection based on neuron magnitudes. Similar to neuron magnitude pruning, this baseline selects experts based on neuron magnitudes in $\boldsymbol{W}_1$ for the generation phase but uses the entire FF blocks for prompting like GRIFFIN. In the case of GLU variants, the neuron-wise norms of $\boldsymbol{W}_1$ and $\boldsymbol{W}_g$ are elementwise multiplied to produce the pruning metric.

Fixing FF sparsity to be 50%, Table 1 shows the generation performance of GRIFFIN. We see that magnitude neuron pruning completely annihilates the original performance in most generation settings. In contrast, GRIFFIN *achieves not only better performance than the baseline in most scenarios, but also preserves most of or matches the original performance on all tasks*. The baseline does much better for classification but still trails behind GRIFFIN (Appendix C). Moreover, GRIFFIN maintains performance on larger batch sizes (Appendix C), extending its deployment settings.

4.2. Efficiency

We now present efficiency metrics of GRIFFIN. We collect synthetic datasets with samples having identical lengths and average results across samples. Like many other MoE methods, GRIFFIN is ideal for single sample inputs, such as in the case of personal devices, so we use batch size 1 for these experiments. Using Hugging Face implementations of Llama 2 13B and Gemma 7B at FP16 precision, we measure the latency in different scenarios on an NVIDIA L40 GPU.

Recalling that our magnitude selection baseline is just neuron pruning at generation, this has the best possible speed-up since there is no selection overhead per sample. From Table 2, GRIFFIN matches the best case, producing up to a $1.29\times$ and $1.25\times$ improvement in latency for long generation at 50% FF sparsity in Gemma 7B and Llama 2 13B, respectively. This illustrates that our method is as fast as a static neuron pruned LLM during generation while being adaptive to preserve the accuracy of the full model. In offloading settings with large models, our method has the potential to further accelerate inference. For a prompt, GRIFFIN essentially performs structured pruning on the massive network, and if this pruned model can fit on a single device, it will avoid offloading for the entirety of generation.

5. Conclusion

We have shown a special form of FF activation sparsity and a simple way to exploit it. Flocking is a curious phenomenon present in many LLMs where tokens within a sequence activate neurons at similar intensities. This motivated the design of GRIFFIN, a learning-free adaptive structured pruning mechanism to remove FF neurons during inference at the sequence level which preserves the full model's performance on a large collection of tasks at 50% FF sparsity while achieving lower latency. With its straightforward algorithm and no-cost deployment, GRIFFIN expands the accessibility of numerous LLMs for generative inference.

Acknowledgements

We thank Zixin Wen for insightful discussions. The work of Y. Chi and H. Dong is supported in part by the grants NSF CCF-1901199, DMS-2134080 and ONR N00014-19-1-2404. The work of H. Dong is also supported in part by the Wei Shen and Xuehong Zhang Presidential Fellowship at Carnegie Mellon University.

References

Anthropic. The claude 3 model family: Opus, sonnet, haiku, 2024.

Bisk, Y., Zellers, R., Bras, R. L., Gao, J., and Choi, Y. Piqa: Reasoning about physical commonsense in natural language. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*, 2020.

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.

Clark, C., Lee, K., Chang, M.-W., Kwiatkowski, T., Collins, M., and Toutanova, K. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*, 2019.

Clark, P., Cowhey, I., Etzioni, O., Khot, T., Sabharwal, A., Schoenick, C., and Tafjord, O. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv:1803.05457v1*, 2018.

Csordás, R., Irie, K., and Schmidhuber, J. Approximating two-layer feedforward networks for efficient transformers. *arXiv preprint arXiv:2310.10837*, 2023.

Dasigi, P., Lo, K., Beltagy, I., Cohan, A., Smith, N. A., and Gardner, M. A dataset of information-seeking questions and answers anchored in research papers. *arXiv preprint arXiv:2105.03011*, 2021.

Dery, L., Kolawole, S., Kagey, J.-F., Smith, V., Neubig, G., and Talwalkar, A. Everybody prune now: Structured pruning of llms with only forward passes. *arXiv preprint arXiv:2402.05406*, 2024.

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.

Dong, H., Chen, B., and Chi, Y. Towards structured sparsity in transformers for efficient inference. In *Workshop on Efficient Systems for Foundation Models@ ICML2023*, 2023.

Fatahalian, K., Sugerman, J., and Hanrahan, P. Understanding the efficiency of gpu algorithms for matrix-matrix multiplication. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, pp. 133–137, 2004.

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.

Frankle, J. and Carbin, M. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

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

Gao, L., Biderman, S., Black, S., Golding, L., Hoppe, T., Foster, C., Phang, J., He, H., Thite, A., Nabeshima, N., et al. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

Gao, L., Tow, J., Abbasi, B., Biderman, S., Black, S., DiPofi, A., Foster, C., Golding, L., Hsu, J., Le Noac'h, A., Li, H., McDonell, K., Muennighoff, N., Ociepa, C., Phang, J., Reynolds, L., Schoelkopf, H., Skowron, A., Sutawika, L., Tang, E., Thite, A., Wang, B., Wang, K., and Zou, A. A framework for few-shot language model evaluation, 12 2023. URL https://zenodo.org/records/10256836.

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

Jacobs, R. A., Jordan, M. I., Nowlan, S. J., and Hinton, G. E. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87, 1991.

Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., Casas, D. d. l., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

Jiang, A. Q., Sablayrolles, A., Roux, A., Mensch, A., Savary, B., Bamford, C., Chaplot, D. S., de las Casas, D., Hanna, E. B., Bressand, F., Lengyel, G., Bour, G., Lample, G., Lavaud, L. R., Saulnier, L., Lachaux, M.-A., Stock, P., Subramanian, S., Yang, S., Antoniak, S., Scao, T. L., Gervet, T., Lavril, T., Wang, T., Lacroix, T., and Sayed, W. E. Mixtral of experts, 2024.

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.

LeCun, Y., Denker, J., and Solla, S. Optimal brain damage. *Advances in neural information processing systems*, 2, 1989.

Li, Y., Yu, Y., Zhang, Q., Liang, C., He, P., Chen, W., and Zhao, T. Losparse: Structured compression of large language models based on low-rank and sparse approximation. *arXiv preprint arXiv:2306.11222*, 2023.

Li, Z., You, C., Bhojanapalli, S., Li, D., Rawat, A. S., Reddi, S. J., Ye, K., Chern, F., Yu, F., Guo, R., et al. The lazy neuron phenomenon: On emergence of activation sparsity in transformers. In *The Eleventh International Conference on Learning Representations*, 2022.

Liang, T., Glossner, J., Wang, L., Shi, S., and Zhang, X. Pruning and quantization for deep neural network acceleration: A survey. *Neurocomputing*, 461:370–403, 2021.

Lin, T., Wang, Y., Liu, X., and Qiu, X. A survey of transformers. *AI Open*, 2022.

Liu, B., Zhang, Z., He, P., Wang, Z., Xiao, Y., Ye, R., Zhou, Y., Ku, W.-S., and Hui, B. A survey of lottery ticket hypothesis. *arXiv preprint arXiv:2403.04861*, 2024.

Liu, Z., Li, F., Li, G., and Cheng, J. Ebert: Efficient bert inference with dynamic structured pruning. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pp. 4814–4823, 2021.

Liu, Z., Wang, J., Dao, T., Zhou, T., Yuan, B., Song, Z., Shrivastava, A., Zhang, C., Tian, Y., Re, C., et al. Deja vu: Contextual sparsity for efficient llms at inference time. In *International Conference on Machine Learning*, pp. 22137–22176. PMLR, 2023.

Ma, X., Fang, G., and Wang, X. Llm-pruner: On the structural pruning of large language models. *Advances in neural information processing systems*, 36:21702–21720, 2023.

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

Mirzadeh, I., Alizadeh, K., Mehta, S., Del Mundo, C. C., Tuzel, O., Samei, G., Rastegari, M., and Farajtabar, M. Relu strikes back: Exploiting activation sparsity in large language models. *arXiv preprint arXiv:2310.04564*, 2023.

Nallapati, R., Zhou, B., Gulcehre, C., Xiang, B., et al. Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023*, 2016.

Narayan, S., Cohen, S. B., and Lapata, M. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. *arXiv preprint arXiv:1808.08745*, 2018.

Nerella, S., Bandyopadhyay, S., Zhang, J., Contreras, M., Siegel, S., Bumin, A., Silva, B., Sena, J., Shickel, B., Bihorac, A., et al. Transformers in healthcare: A survey. *arXiv preprint arXiv:2307.00067*, 2023.

Piórczyński, M., Szatkowski, F., Bałazy, K., and Wójcik, B. Exploiting transformer activation sparsity with dynamic inference. *arXiv preprint arXiv:2310.04361*, 2023.

Rae, J. W., Potapenko, A., Jayakumar, S. M., Hillier, C., and Lillicrap, T. P. Compressive transformers for long-range sequence modelling. *arXiv preprint*, 2019. URL https://arxiv.org/abs/1911.05507.

Reddy, S., Chen, D., and Manning, C. D. Coqa: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics*, 7:249–266, 2019.

Roemmele, M., Bejan, C. A., and Gordon, A. S. Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In *2011 AAAI Spring Symposium Series*, 2011.

Santacroce, M., Wen, Z., Shen, Y., and Li, Y. What matters in the structured pruning of generative language models? *arXiv preprint arXiv:2302.03773*, 2023.

Shaham, U., Segal, E., Ivgi, M., Efrat, A., Yoran, O., Haviv, A., Gupta, A., Xiong, W., Geva, M., Berant, J., et al. Scrolls: Standardized comparison over long language sequences. *arXiv preprint arXiv:2201.03533*, 2022.

Shazeer, N. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

Sun, M., Liu, Z., Bair, A., and Kolter, J. Z. A simple and effective pruning approach for large language models. *arXiv preprint arXiv:2306.11695*, 2023.

Team, G., Anil, R., Borgeaud, S., Wu, Y., Alayrac, J.-B., Yu, J., Soricut, R., Schalkwyk, J., Dai, A. M., Hauth, A., et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

Team, G., Mesnard, T., Hardin, C., Dadashi, R., Bhupatiraju, S., Pathak, S., Sifre, L., Rivière, M., Kale, M. S., Love, J., et al. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024.

Team, S. Sparse large language models with relu activation, 2023.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 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.

Wang, Y. E., Wei, G.-Y., and Brooks, D. Benchmarking tpu, gpu, and cpu platforms for deep learning. *arXiv preprint arXiv:1907.10701*, 2019.

Xia, M., Zhong, Z., and Chen, D. Structured pruning learns compact and accurate models. *arXiv preprint arXiv:2204.00408*, 2022.

Xia, M., Gao, T., Zeng, Z., and Chen, D. Sheared llama: Accelerating language model pre-training via structured pruning. *arXiv preprint arXiv:2310.06694*, 2023.

Yerram, V., You, C., Bhojanapalli, S., Kumar, S., Jain, P., Netrapalli, P., et al. Hire: High recall approximate top-k estimation for efficient llm inference. *arXiv preprint arXiv:2402.09360*, 2024.

Yin, L., Jaiswal, A., Liu, S., Kundu, S., and Wang, Z. Pruning small pre-trained weights irreversibly and monotonically impairs ”difficult” downstream tasks in llms, 2024.

Zellers, R., Holtzman, A., Bisk, Y., Farhadi, A., and Choi, Y. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*, 2019.

Zhang, S., Roller, S., Goyal, N., Artetxe, M., Chen, M., Chen, S., Dewan, C., Diab, M., Li, X., Lin, X. V., Mihaylov, T., Ott, M., Shleifer, S., Shuster, K., Simig, D., Koura, P. S., Sridhar, A., Wang, T., and Zettlemoyer, L. Opt: Open pre-trained transformer language models, 2022.

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.

Zheng, H., Bai, X., Chen, B., Lai, F., and Prakash, A. Learn to be efficient: Build structured sparsity in large language models. *arXiv preprint arXiv:2402.06126*, 2024.

A. Feedforward Activations Between Sequences

Flocking patterns are unique to each sequence. We show this by taking the ℓ_2-norm of $\overline{\boldsymbol{Z}}$ along the token axis to obtain a length D_{FF} vector for each sample or sequence, roughly capturing the contribution of each FF neuron throughout a sequence. Taking the top-k of this for each sample at each layer, we find the Jaccard similarity between two sequences based on the indices selected for different k. In other words, we compute the intersection over union of every unique pair of top-k sets. Higher values indicate more similar top-k sets. From Figure 3 where we aggregate Jaccard similarities across WikiText (Merity et al., 2016) samples, we observe a lack of inter-sample activation similarities for the vast majority of layers in Llama 2 7B and Gemma 7B, unless the sets of selected neurons are large.

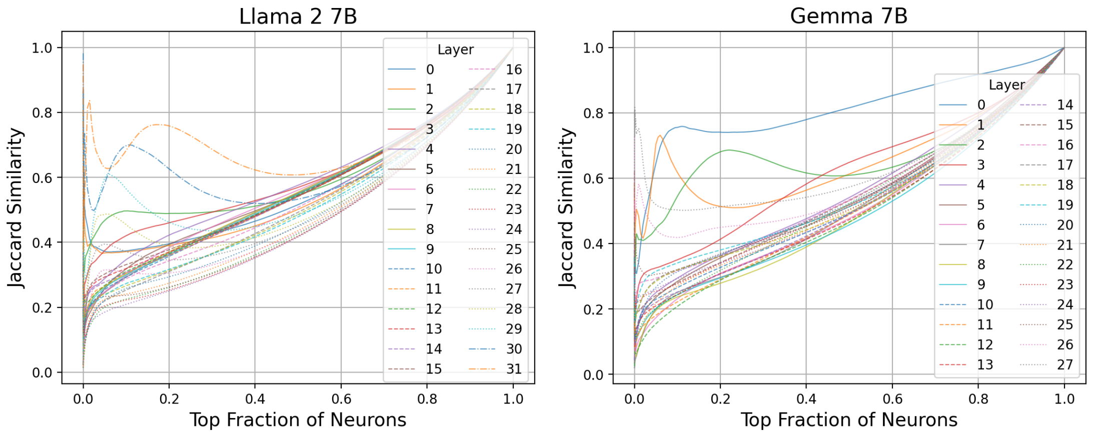

Figure 3. Average Jaccard similarity between WikiText samples' top FF neuron activations in Llama 2 7B (left) and Gemma 7B (right). Higher values indicate greater similarity.

B. Selection Metric

Here we present visualizations of our gating statistic $\boldsymbol{s}$ from (6). For a single sample, we find $\boldsymbol{s}$ and sort the entries normalized between 0 and 1 in Figure 4. In both models, values in $\boldsymbol{s}$ are heavily concentrated in a handful of features. Since $\boldsymbol{s}$ aggregates the relative activation magnitudes across tokens, this implies $\boldsymbol{s}$ can capture heavily and frequently activated neurons.

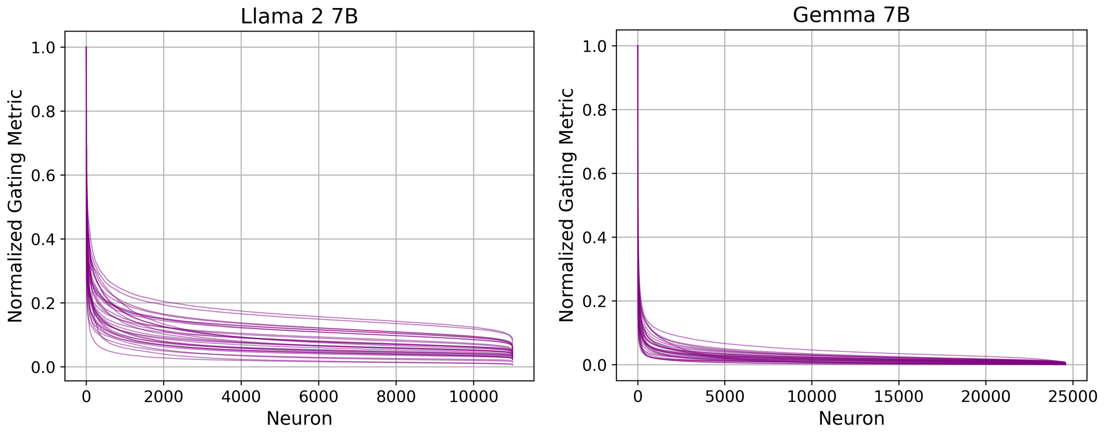

Figure 4. Sorted entries of $\boldsymbol{s}$ for each layer of Llama 2 7B (left) and Gemma 7B (right) with a sequence from PG-19 as the input. Each line is a layer.

Table 3. Generation tasks XSum (1-shot), CNN/DailyMail (1-shot), CoQA (0-shot), SCROLLS QASPER (0-shot) at 50% FF sparsity. Magnitude neuron pruning fails in almost every case while GRIFFIN effectively preserves performance.

Model	XSum (Rouge-1/2/L)	CNN/DailyMail (Rouge-1/2/L)	CoQA (F1/EM)	QASPER (F1)
Llama 2 7B	27.15/9.06/22.62	10.08/0.13/9.55	77.35/63.88	26.31
Magnitude	9.71/1.31/8.59	9.66/0.63/9.32	56.59/39.93	12.93
GRIFFIN	24.75/7.41/20.55	10.97/0.66/10.37	77.18/63.58	25.76
Llama 2 13B	26.90/9.45/22.09	2.51/0.22/2.34	79.18/66.37	28.32
Magnitude	5.72/0.78/5.06	0.02/0.00/0.02	65.69/47.87	15.55
GRIFFIN	25.69/7.85/20.89	3.31/0.78/3.07	79.22/66.62	27.91
Gemma 7B	26.86/9.15/22.03	17.45/4.15/15.94	79.04/65.25	30.78
Magnitude	1.49/0.01/1.47	0.00/0.00/0.00	2.92/1.50	7.02
GRIFFIN	25.86/7.81/20.93	18.26/4.75/16.58	78.52/64.62	27.37
Mistral 7B	28.67/10.21/23.64	0.28/0.01/0.28	80.70/67.30	24.56
Magnitude	3.58/0.27/3.31	0.26/0.03/0.26	61.99/45.93	17.18
GRIFFIN	26.59/8.70/22.17	1.26/0.21/1.17	80.15/66.50	23.92
OPT 6.7B	23.60/7.04/19.46	13.85/1.54/13.04	68.70/54.98	18.53
Magnitude	1.63/0.00/1.54	1.20/0.00/1.17	31.53/16.52	7.28
GRIFFIN	21.17/5.42/17.58	13.01/1.06/12.26	68.99/55.00	17.40
OPT 13B	25.14/7.93/20.80	13.22/1.18/12.46	69.51/55.67	20.58
Magnitude	1.23/0.00/1.21	1.29/0.00/1.29	39.38/27.07	8.87
GRIFFIN	22.11/6.28/18.29	12.92/1.13/12.20	69.07/54.83	20.16
ReluLlama 2 7B	25.10/7.81/20.76	20.95/6.79/19.24	78.49/66.73	23.31
Magnitude	9.09/0.22/8.20	8.50/0.14/8.17	19.43/6.48	7.21
GRIFFIN	21.83/5.88/18.09	16.85/4.96/14.69	78.35/67.10	22.29

C. More Evaluations

We further evaluate GRIFFIN across more metrics, tasks, and sparsity levels.

C.1. Generation

Extending Table 1 to include more evaluation metrics at 50% FF sparsity, Table 3 shows the same generation tasks but with additional metrics. GRIFFIN performs better than the baseline across all models and tasks.

C.2. Classification

As our method is designed specifically for generation, we alter classification evaluations to simulate generation. In typical classification tasks, LLMs do not enter the generative phase since the final token output of the prompt phase indicates the class. Consequently, directly applying GRIFFIN for classification tasks trivially yields the exact performance of the original model. Therefore, we treat all tokens but the final input token as the prompt. Then, the model is forced to go into the generation phase for one step to produce the class.

Table 4 contains classification results across several datasets at 50% FF sparsity. Here, the baseline does fairly well but still significantly lags behind GRIFFIN in most cases.

C.3. Varying FF Sparsity

In this section, we look into the relationship between GRIFFIN sparsity levels and performance degradation. This translates to varying k when we select the top-k of our statistic $\boldsymbol{s}$. To compare the performance degradation across multiple tasks, we plot the ratio of the final performance metrics between GRIFFIN and the full model in Figure 5. We see most of the performance is preserved at 50% FF sparsity in Llama 2 7B, Gemma 7B, and Mistral 7B. Different tasks have different tipping points where performance sharply drops, which may be related to the difficulty of the task (Yin et al., 2024).

Table 4. Classification tasks (0-shot unnormalized accuracy) at 50% FF sparsity.

MODEL	HELLASWAG	PIQA	COPA	ARC-E	ARC-C	BOOLQ
LLAMA 2 7B	57.16	78.07	87.00	76.35	43.34	77.71
MAGNITUDE	57.12	77.31	84.00	70.33	40.27	66.54
GRIFFIN	57.11	77.69	86.00	74.54	42.75	73.15
LLAMA 2 13B	60.06	79.05	90.00	79.46	48.46	80.61
MAGNITUDE	60.00	79.00	88.00	74.07	46.25	70.52
GRIFFIN	60.10	79.11	89.00	77.19	46.84	78.50
GEMMA 7B	60.61	80.30	88.00	82.74	50.09	83.49
MAGNITUDE	46.24	73.12	57.00	45.20	32.76	62.84
GRIFFIN	60.62	79.98	88.00	81.65	50.09	81.90
MISTRAL 7B	61.21	80.58	92.00	80.89	50.43	83.61
MAGNITUDE	61.15	80.36	86.00	74.20	48.89	60.40
GRIFFIN	61.18	80.52	91.00	79.25	50.00	80.06
OPT 6.7B	50.48	76.28	81.00	65.53	30.55	66.12
MAGNITUDE	49.21	72.63	79.00	47.60	27.13	40.15
GRIFFIN	50.44	75.63	80.00	63.93	30.55	65.44
OPT 13B	52.46	75.90	86.00	67.05	32.94	65.81
MAGNITUDE	51.31	74.21	81.00	49.41	28.07	38.75
GRIFFIN	52.42	76.17	86.00	66.92	33.19	67.65

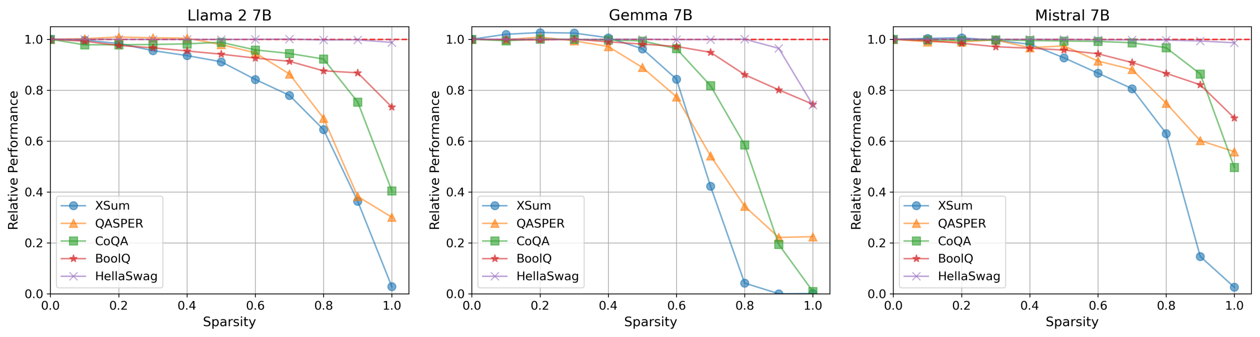

Figure 5. Relative performance of GRIFFIN for Llama 2 7B (left), Gemma 7B (center), and Mistral 7B (right) as we enforce varying degrees of sparsity per FF block. For all tasks, the original model's performance for each task is normalized to 1.

D. Ablations and Analysis

D.1. Batching

We can also extend GRIFFIN with batching. Within a batch, we sum the statistic $\boldsymbol{s}_i$ of each sample, scaled inversely by each sample's root prompt length, S_i, and use that to select neurons. In other words, the same experts for all samples within a batch would be selected based on $\boldsymbol{s} = \sum_i \boldsymbol{s}_i / \sqrt{S_i}$. Results in Figure 6 indicate that GRIFFIN is robust to batching, as performance degrades very slowly as we increase the batch size.

D.2. Sampling-based Selection

Here, we verify that given the statistic $\boldsymbol{s}$, top-k expert selection produces better results than sampling-based methods. The methods we compare against include sampling based on the weights in $\boldsymbol{s}$ and combining top-k selection for half of the experts followed by weighted sampling. Based on Table 5, we can see that sampling generally degrades performance much more.

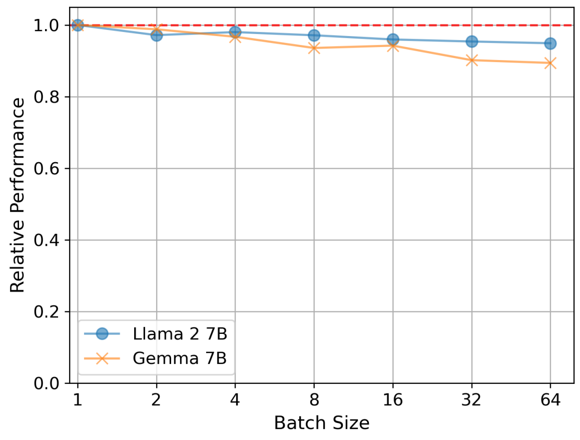

Figure 6. Relative (compared to single sample inference) 1-shot XSum Rouge-1 scores for increasing batch sizes. For each batch, the same 50% of FF expert neurons are selected.

Table 5. Comparison between different expert selection methods at 50% FF sparsity.

SELECTION METHOD	XSUM (ROUGE-1/2/L)	CNN/DAILYMAIL (ROUGE-1/2/L)	COQA (F1/EM)	QASPER (F1)
Llama 2 7B				
FULL	27.15/9.06/22.62	10.08/0.13/9.55	77.35/63.88	26.31
TOP-k	**24.75/7.41/20.55**	**10.97/0.66/10.37**	**77.18**/63.58	**25.76**
SAMPLING	21.04/5.22/17.12	8.78/0.49/8.28	76.15/62.53	24.46
TOP-k + SAMPLING	24.35/7.08/20.07	10.45/0.48/9.88	77.12/**64.17**	25.22
Gemma 7B				
FULL	26.86/9.15/22.03	17.45/4.15/15.94	79.04/65.25	30.78
TOP-k	**25.86/7.81/20.93**	**18.26/4.75/16.58**	**78.52/64.62**	**27.37**
SAMPLING	20.25/5.16/16.79	8.34/1.71/7.72	75.02/59.93	24.97
TOP-k + SAMPLING	24.47/7.43/19.98	10.93/2.60/9.98	76.76/62.12	27.09

D.3. Prompt vs. Generation Length

We find that GRIFFIN can potentially be made more robust for long generation by lengthening the prompt. To see this, we use language modeling on the concatenated version of WikiText to simulate generation. For a length S input into the FF block, we designate the first P tokens as the prompt and the last G tokens as the generated portion such that $P + G = S$. The prompt partition is used to calculate our statistic $\boldsymbol{s}$ and determine the experts. The prompt partition uses the full FF block while the generation partition only uses the selected experts. When comparing the original model with GRIFFIN, we only compute the perplexity of the outputs from the generation partition since the other outputs will be identical. Based on Figure 7, GRIFFIN gets closer to the full model outputs when the prompt length increases and generation length decreases, meaning the difficulty with long generation can be suppressed with longer prompts.

D.4. Sparsity in Random Sequences

As further exploration into flocking, we investigate this phenomenon with random inputs. As input sequences, we use a sample from concatenated WikiText, a permuted version of that sample, and completely random sequence where tokens are uniformly sampled from the vocabulary. Seen in Figure 8, this structure exists in permuted and random inputs, perhaps even more consistently than in unperturbed sequences. This suggests something within language actually diversifies the activations, the cause of which would be of interest for future work.

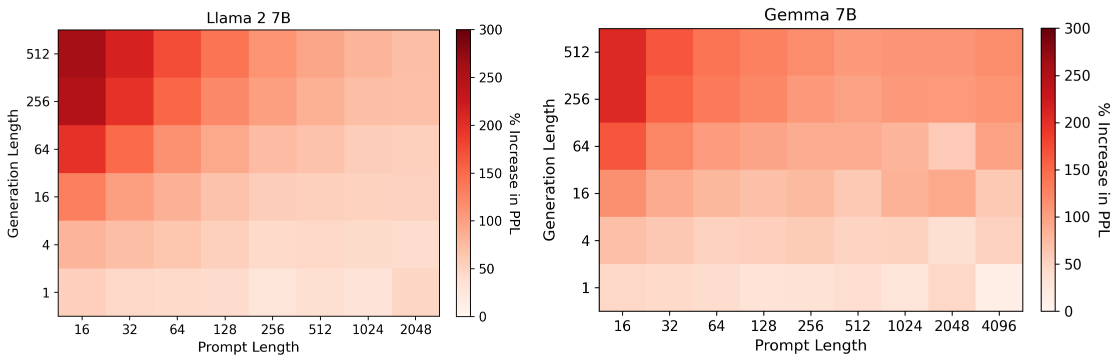

Figure 7. Prompt length vs. generation length for Llama 2 7B (left) and Gemma 7B (right) as measured by increase in perplexity (PPL) from the full model on concatenated WikiText at 50% FF sparsity.

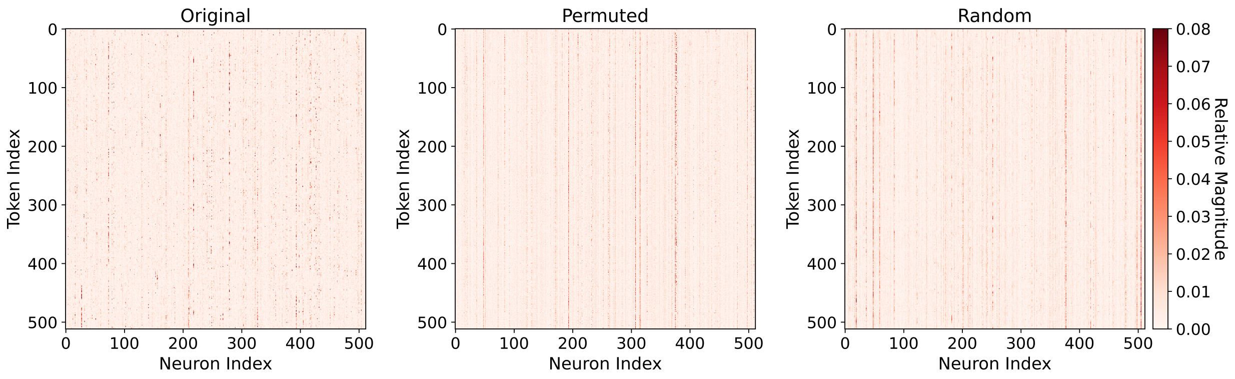

Figure 8. First 512 tokens and their relative FF activation magnitudes in layer 18 of Gemma 7B when inputting the original WikiText sequence (left), permuted sequence (center), and random tokens (right). Best viewed zoomed in.

E. More Flocking Examples

We provide more example of flocking across different layers of the LLM. Figure 9 and Figure 10 show flocking in Gemma 7B. Figure 11 and Figure 12 show flocking in Llama2 7B. Flocking in Gemma 7B is more visually distinct while activations in Llama2 7B are more distributed.

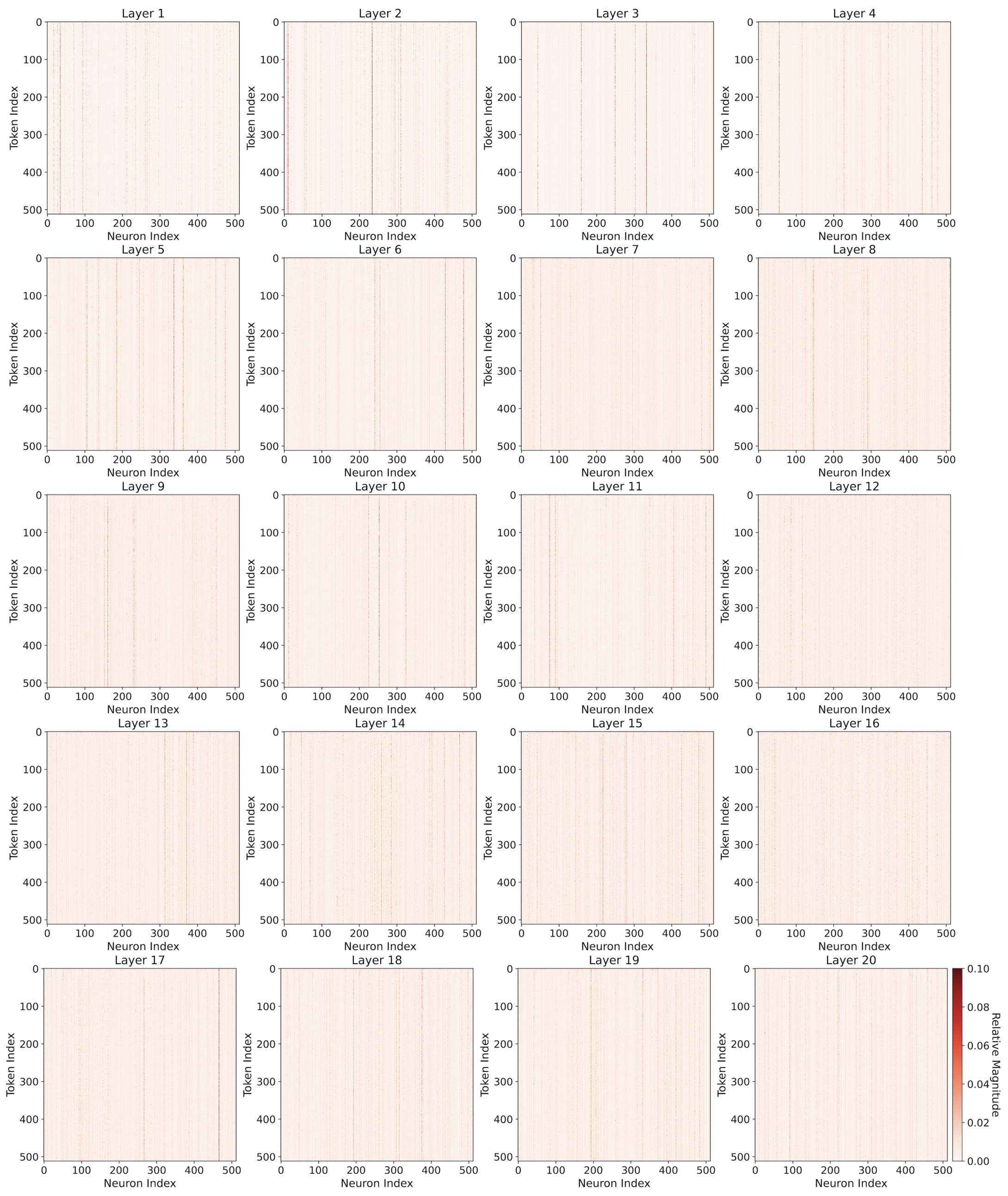

Figure 9. First 512 tokens and their relative FF activation magnitudes in layers 1 to 20 of Gemma 7B when inputting a sequence from PG-19.

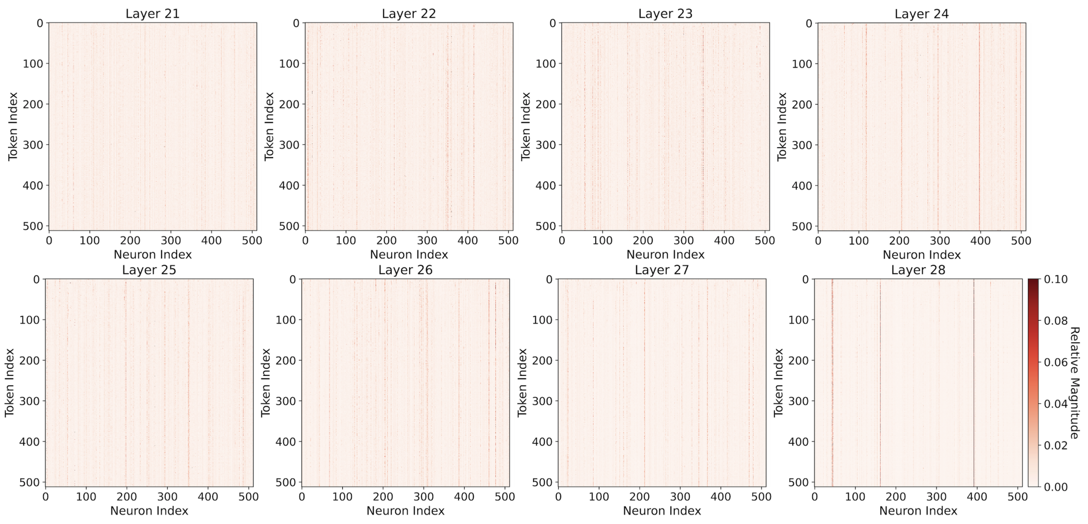

Figure 10. First 512 tokens and their relative FF activation magnitudes in layers 21 to 28 of Gemma 7B when inputting a sequence from PG-19.

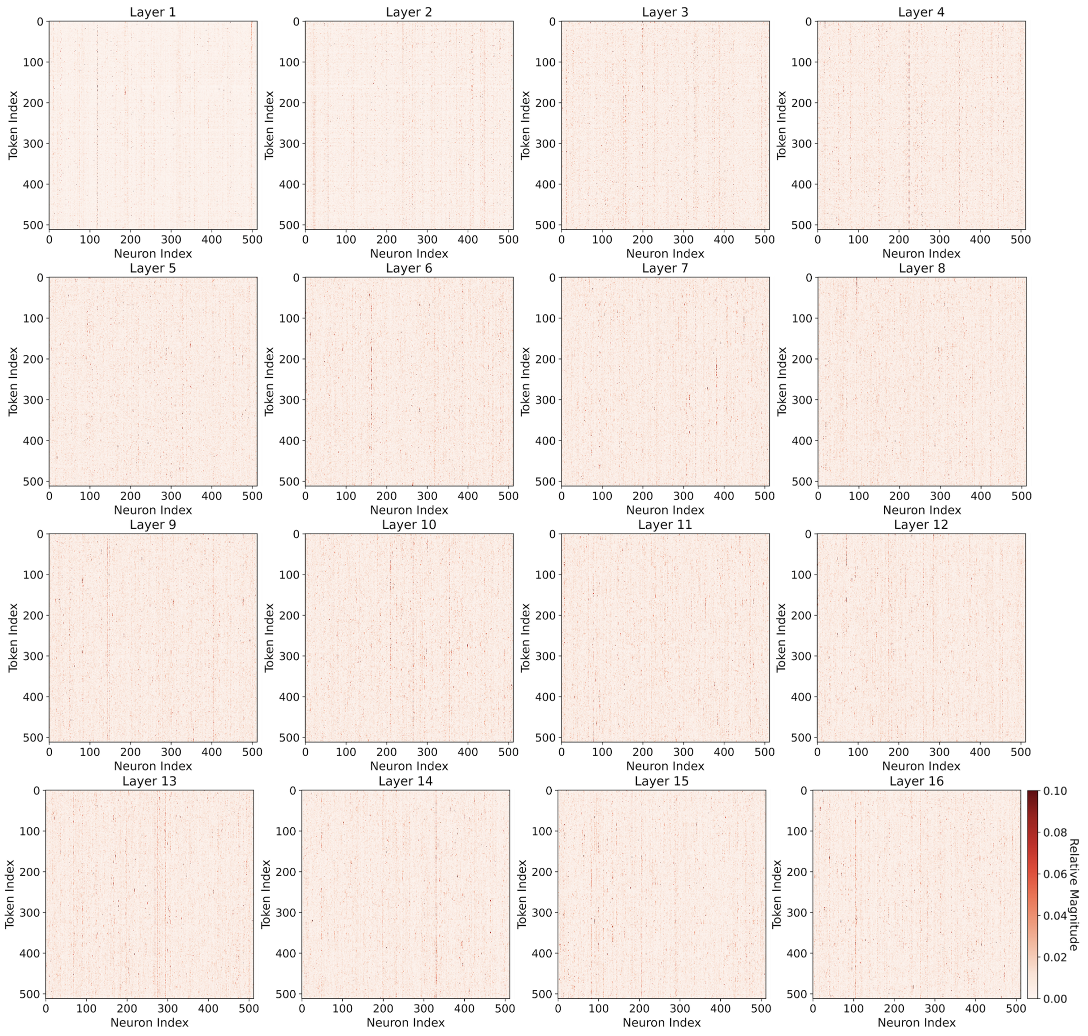

Figure 11. First 512 tokens and their relative FF activation magnitudes in layers 1 to 16 of Llama 2 7B when inputting a sequence from PG-19.

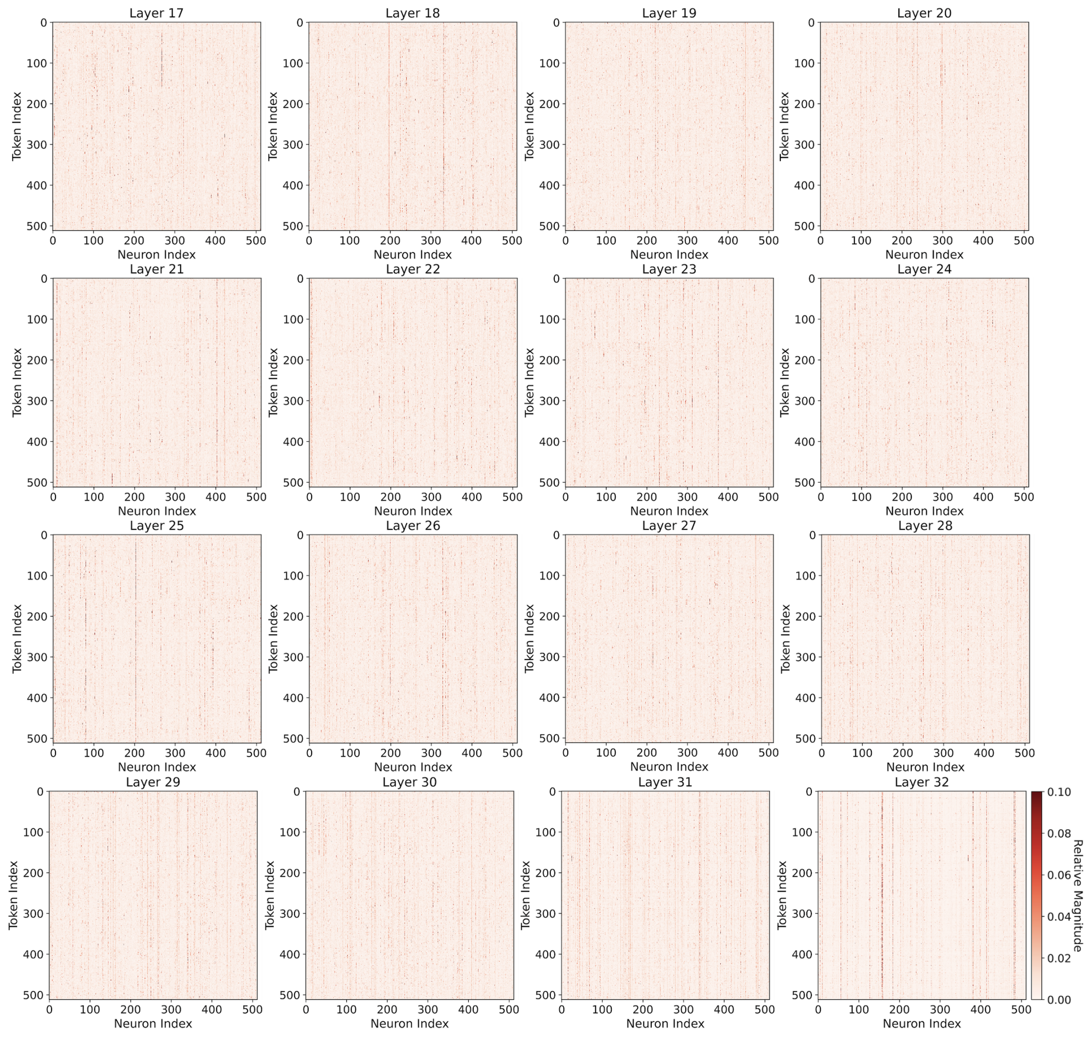

Figure 12. First 512 tokens and their relative FF activation magnitudes in layers 17 to 32 of Llama 2 7B when inputting a sequence from PG-19.