A Simple and Effective Pruning Approach for Large Language Models

Mingjie Sun [1 *] **Zhuang Liu** [2 *] **Anna Bair** [1] **J. Zico Kolter** [1 3]

Abstract

As their size increases, Large Languages Models (LLMs) are natural candidates for network pruning methods: approaches that drop a subset of network weights while striving to preserve performance. Existing methods require either retraining, which is rarely affordable for billion-scale LLMs, or solving a weight reconstruction problem reliant on second-order information, which may also be computationally expensive. In this paper, we introduce a novel, straightforward yet effective pruning method, termed Wanda (Pruning by **W**eights **and a**ctivations), designed to induce sparsity in pretrained LLMs. Motivated by the recent observation of emergent large magnitude features in LLMs, our approach prune weights with the smallest magnitudes multiplied by the corresponding input activations, on a per-output basis. Notably, Wanda requires *no* retraining or weight update, and the pruned LLM can be used *as is*. We conduct a thorough evaluation of our method on LLaMA across various language benchmarks. Wanda significantly outperforms the established baseline of magnitude pruning and competes favorably against recent methods involving intensive weight update. Code is available at `https://github.com/locuslab/wanda`.

1. Introduction

Large language models (Touvron et al., 2023; Brown et al., 2020; OpenAI, 2023) have recently reshaped the field of NLP with their remarkable performance across a range of complex language benchmarks (Wei et al., 2022b; Bubeck et al., 2023). Existing methods for compressing LLMs (Dettmers et al., 2022; Xiao et al., 2023; Frantar et al., 2023) mostly focus on model quantization, where parameters are quantized into lower bit-level representations.

Network pruning (Hassibi et al., 1993; LeCun et al., 1989), another popular branch of network compression, has received little focus in compressing LLMs. This seems to contradict the trend of model compression in the pre-LLM era. A quick review of existing pruning methods reveals a possible reason: they typically require retraining (Han et al., 2015; Blalock et al., 2020; Jonathan & Michael, 2019), which are limited by the sheer computational resources of LLMs. A recent LLM pruning approach, SparseGPT (Frantar & Alistarh, 2023), does not require traditional retraining, but still demands an intensive weight update process.

The argument concerning the need for retraining and weight update does not fully capture the challenges of pruning LLMs. One might reasonably expect to obtain a fairly high-performing initialization point for retraining using existing popular pruning methods. However, a recent study (Frantar & Alistarh, 2023) finds that magnitude pruning (Han et al., 2015; Zhu & Gupta, 2017; Gale et al., 2019), a well-established pruning approach, fails dramatically on LLMs with relatively low levels of sparsity. Considering the past success of magnitude pruning on smaller networks, this result suggests that LLMs, despite having 100 to 1000 times more parameters, are much more difficult to prune directly.

In this work, we address this challenge by introducing a straightforward and effective approach, termed Wanda (Pruning by **W**eights **and a**ctivations). This technique successfully prunes LLMs to high degrees of sparsity without *any* need for modifying the remaining weights. This is inspired by insights from a recent study (Dettmers et al., 2022) observing that a small subset of hidden state features are exceptionally large in magnitude, a property unique to LLMs. We introduce a novel pruning metric, where each weight is evaluated by the product of its magnitude and the norm of the corresponding input activations, estimated using a small set of calibration data. Our method uses this metric to induce sparsity in pretrained LLMs by comparing weights locally within each output of linear layers and removing lower priority weights. Our approach is computationally efficient, able to be executed in a single forward pass, and requires minimal memory overhead.

We experiment on LLaMA (Touvron et al., 2023), one of the most effective open-sourced LLM families. Our results demonstrate Wanda can find efficient sparse networks directly from pretrained LLMs, without any retraining or weight update. Wanda outperforms magnitude pruning by a large margin and also outperforms or matches the performance of a recent LLM pruning method (Frantar & Alistarh, 2023), while requiring a lower computational cost.

[*]Equal contribution [1]Carnegie Mellon University [2]Meta AI, FAIR [3]Bosch Center for AI.

Work presented at the ES-FoMo Workshop at ICML 2023.

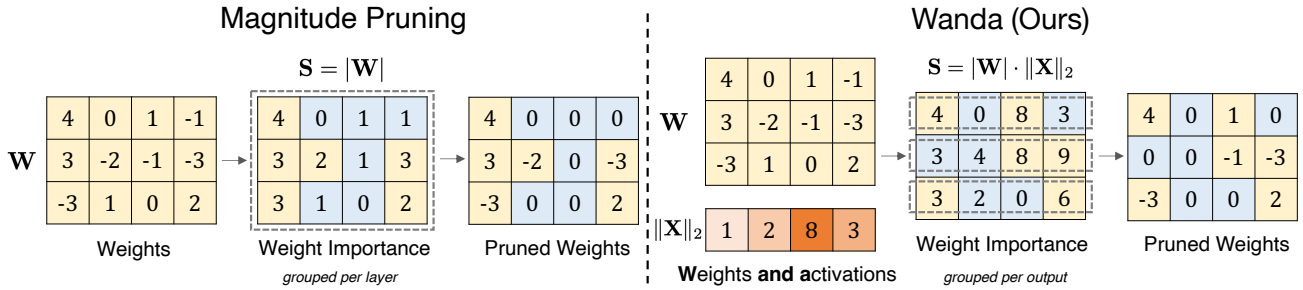

Figure 1: Illustration of our proposed method Wanda (Pruning by **W**eights **and a**ctivations), compared with the magnitude pruning approach. Given a weight matrix $\mathbf{W}$ and input feature activations $\mathbf{X}$, we compute the weight importance as the elementwise product between the weight magnitude and the norm of input activations ($|\mathbf{W}| \cdot \|\mathbf{X}\|_2$). Weight importance scores are compared on a per-output basis (within each row in $\mathbf{W}$), rather than globally across the entire matrix.

2. Pruning by Weights and Activations

In this section, we motivate and describe our proposed pruning approach, Wanda (Pruning by **W**eights **and a**ctivations). In Figure 1, we show an overview of our pruning approach.

We first start with a motivating example. Consider a neuron with two inputs and corresponding weights: $\mathbf{y} = \mathbf{w}_1\mathbf{x}_1 + \mathbf{w}_2\mathbf{x}_2$, where $|\mathbf{w}_1| \leq |\mathbf{w}_2|$. Now suppose the goal is to select one weight for removal while incurring less change on the output. The standard approach of magnitude pruning would always remove weight $\mathbf{w}_1$, which may be a good strategy if input features $\mathbf{x}_1$ and $\mathbf{x}_2$ have similar magnitudes. However, as recently observed in LLMs (Dettmers et al., 2022), the two input features can differ significantly in scale. For instance, it is possible that $|\mathbf{x}_1| \gg |\mathbf{x}_2|$, and as a result, $|\mathbf{w}_1\mathbf{x}_1| \gg |\mathbf{w}_2\mathbf{x}_2|$. In this case, we should remove weight $\mathbf{w}_2$ instead, because this clearly exerts a smaller influence on the neuron output $\mathbf{y}$ than removing weight $\mathbf{w}_1$.

This motivating example with the simplest linear layer hints at a major limitation of magnitude pruning: it does not take into account input activations, which could play an equally important role as weight magnitudes in determining the neuron output. For pruning LLMs, this is especially critical considering the emergent large magnitude features found within them. Thus, as the first part of our method, we propose a new pruning metric to handle such a limitation, while also maintaining the simplicity of magnitude pruning.

Pruning Metric. Consider a fully connected layer with weight $\mathbf{W}$ of shape $(C_{\texttt{out}}, C_{\texttt{in}})$. For language models, this linear layer takes in input activations $\mathbf{X}$ with a shape of $(N \times L, C_{\texttt{in}})$, where N and L are batch and sequence dimensions respectively. For each individual weight, we propose to evaluate its importance by the product of its magnitude and the corresponding input feature norm. Specifically, the score for the current weight $\mathbf{W}_{ij}$ is defined by:

$$\mathbf{S}_{ij} = |\mathbf{W}_{ij}| \cdot \|\mathbf{X}_j\|_2 \tag{1}$$

where $|\cdot|$ represents the absolute value operator, $\|\mathbf{X}_j\|_2$ evaluates the ℓ_2 norm of jth features aggregated across $N \times L$ different tokens, and the final score is computed by the product of these two scalar values. We find that ℓ_2 norm tends to work better than other norm functions in measuring activation magnitudes, for instance, ℓ_1 and ℓ_∞ norm. We hypothesize this is likely because ℓ_2 is generally a smoother metric (Cortes et al., 2009; Lewkowycz & Gur-Ari, 2020).

Algorithm 1 PyTorch code for Wanda

```
# W: weight matrix (C_out, C_in);
# X: input matrix (N * L, C_in);
# s: desired sparsity, between 0 and 1;

def prune(W, X, s):
  metric = W.abs() * X.norm(p=2, dim=0)

  _, sorted_idx = torch.sort(metric, dim=1)
  pruned_idx = sorted_idx[:,:int(C_in * s)]
  W.scatter_(dim=1, index=pruned_idx, src=0)

  return W
```

Pruning Granularity. We argue that carefully chosen pruning granularity, i.e. the set of weights among which to compare, has an important role in pruning LLMs. Despite the common practice of comparing weights per layer or globally, we suggest that pruning LLMs should benefit from more local granularity levels. In our method, we compare weights per output neuron. Specifically, for a weight $\mathbf{W}_{ij}$ that connects input j to output i inside the linear layer, the group of comparison for this weight is defined as all weights connecting to output i:

$$\mathbf{G}_{ij} = \{\mathbf{W}_{uv} \,|\, u = i\} \tag{2}$$

Under this granularity, for a pre-defined sparsity ratio $s\%$, we eliminate $s\%$ of the weights connected to *each output*. This practice may seem counter-intuitive, since we are basically pruning under a stricter sparsity pattern. However, we find that it is *consistently better* than layer-wise pruning for LLMs. A possible reason is that removing weights at a uniform ratio per output helps prevent imbalanced pruning across different output features.

Method	Fine-tuning	Weight Update	Calibration Data	Pruning Metric $\mathbf{S}_{ij}$	Complexity
Magnitude	✗	✗	✗	$\lvert\mathbf{W}_{ij}\rvert$	$O(1)$
SparseGPT	✗	✓	✓	$\left[\lvert\mathbf{W}\rvert^2/\text{diag}\left[(\mathbf{X}\mathbf{X}^T+\lambda\mathbf{I})^{-1}\right]\right]_{ij}$	$O(d^3_{\texttt{hidden}})$
Wanda	✗	✗	✓	$\lvert\mathbf{W}_{ij}\rvert\cdot\lVert\mathbf{X}_j\rVert_2$	$O(d^2_{\texttt{hidden}})$

Table 1: Properties of practical LLM pruning algorithms.

			LLaMA			
Method	Weight Update	Sparsity	7B	13B	30B	65B
Dense	-	0%	5.68	5.09	4.77	3.56
Magnitude	✗	50%	17.29	20.21	7.54	5.90
SparseGPT	✓	50%	**7.22**	6.21	5.31	**4.57**
Wanda	✗	50%	7.26	**6.15**	**5.24**	**4.57**
Magnitude	✗	4:8	16.84	13.84	7.62	6.36
SparseGPT	✓	4:8	8.61	**7.40**	6.17	5.38
Wanda	✗	4:8	**8.57**	**7.40**	**5.97**	**5.30**
Magnitude	✗	2:4	42.13	18.37	9.10	7.11
SparseGPT	✓	2:4	**11.00**	**9.11**	7.16	6.28
Wanda	✗	2:4	11.53	9.58	**6.90**	**6.25**

Table 2: WikiText validation perplexity of pruning methods for LLaMA model family.

Procedure. Wanda can be implemented and integrated seamlessly within a *single* forward pass of the LLM model, where feature norm statistics $\|\mathbf{X}_j\|_2$ are estimated with a set of calibration data. We provide the PyTorch code of our approach in Algorithm 1. Given a pretrained LLM, we compute our pruning metric from the initial to the final layers of the network. After pruning a preceding layer, the subsequent layer receives updated input activations, based on which its pruning metric will be computed. A recent method for pruning LLMs, SparseGPT (Frantar & Alistarh, 2023), requires iterative weight update with a sophisticated procedure in the gradual pruning process, while Wanda requires no weight update. The sparse LLM after pruning is ready to use without further training or weight adjustment. Last, Wanda can also be easily extended to structured N:M Sparsity (see Appendix C for details).

A comparison of LLM pruning methods are in Table 1. Overall, our pruning approach Wanda has several attractive properties as an approach for pruning LLMs:

1. It maintains the simplicity of magnitude pruning in the pre-LLM era, requiring no gradient computation via back-propagation or any second-order Hessian inverses, but is also highly effective in discovering sparse networks in pretrained LLMs.
2. Wanda can be done with a single forward pass of the LLM. At each layer, pruned weights can be decided in one shot without an iterative procedure. In practice, Wanda can be 300 times faster in pruning LLMs compared with SparseGPT (Frantar & Alistarh, 2023).
3. Our approach entails no weight updates on pruned networks, suggesting that LLMs have effective and *exact* sparse sub-networks, instead of them merely existing in the neighborhood of the original weights.

3. Experiments

We evaluate our approach on LLaMA (Touvron et al., 2023) and compare against two baselines listed in Table 1 (see Appendix A for more details on experimental setup). We provide results for representative LLM model families released before LLaMA in Appendix E.5. We provide additional results for pruning image classifiers in Appendix F.

3.1. Language Modeling

Unstructured Sparsity. For each of the LLaMA models, we prune it to unstructured 50% sparsity for all methods. Results are shown in Table 2. Without any weight update, Wanda outperforms the established pruning approach of magnitude pruning by a large margin. For instance, for LLaMA-7B, Wanda is able to find sparse networks with a perplexity of 7.26, significantly better than the magnitude pruning baseline 17.29. Our method also performs on par with or in most cases better than the prior reconstruction-based method SparseGPT.

Structured N:M Sparsity. We now turn our eyes to structured N:M sparsity. Note that N:M sparsity by definition is a more restrictive sparsity pattern than unstructured sparsity. Thus, it is expected that such structured sparsity patterns would lead to worse results. Results for structured 4:8 and 2:4 sparsity are shown in the lower parts of Table 2. We can see that Wanda can be easily generalized to structured N:M sparsity. Across 4:8 and 2:4 sparsity, our method

consistently finds highly effective sparse sub-networks, outperforming baseline approaches in most cases, especially for larger models (e.g. 30B and 65B).

Remark. Note that in certain cases, Wanda obtains results slightly better than but somewhat close to those of SparseGPT. This may be related to the fact that our pruning metric shares an *implicit* connection to the OBS reconstruction error in Equation 3, as identified and explained in Appendix D. While perplexity improvements in these cases over SparseGPT may not be considered substantial, Wanda is much simpler and more computationally efficient.

3.2. Ablation Study

Pruning Configuration. Our method differs from previous pruning methods in two aspects: the pruning metric and the pruning granularity. While we achieve superior performance over prior approaches, it is not clear what the real source of success is. Thus we revisit the configurations of magnitude pruning, SparseGPT, and our method. We identify a total of three pruning metrics and three pruning granularities. Note that SparseGPT compares weights locally within a block of 128 consecutive columns, which we denote as 'col, 128'. Similarly, we refer to our granularity as 'row, 1'.

	Pruning Granularity		
Pruning Metric	layer	col, 128	row, 1
$\lvert\mathbf{W}_{ij}\rvert$	17.29	16.82	13.41
$\left[\lvert\mathbf{W}\rvert^2/\mathrm{diag}(\mathbf{H}^{-1})\right]_{ij}$	7.91	8.02	7.41
$\lvert\mathbf{W}_{ij}\rvert \cdot \lVert\mathbf{X}_j\rVert$	7.95	8.12	**7.26**

Table 3: Ablation on pruning configuration.

A pruning configuration with a specified metric and granularity uniquely defines a pruning algorithm. Thus we consider 9 unique pruning configurations obtained by the combination of 3 metrics and 3 granularities. To isolate the effect of OBS based weight update, we simply zero out pruned weights for this ablation. The results are shown in Table 3, where we highlight the configuration of our approach. We can see that our pruning configuration is indeed the optimal.

Robustness to Calibration Samples. We vary the number of calibration samples by selecting different sample sizes ranging between 1 and 256. Results are summarized in Figure 2. We see a clear difference in trend as the size of calibration data changes, where Wanda is much more robust with few calibration samples. Notably, even with a *single* sample, pruned networks from Wanda have a perplexity of 7.66. This may be because input norm statistics $\lVert\mathbf{X}_j\rVert$ could be much easier to estimate than the full inverse hessian $\mathbf{H}^{-1}$ of the local layer-wise reconstruction problem.

Weight Update. We test if we could obtain better pruned models by incorporating the OBS weight update process proposed in SparseGPT (Frantar & Alistarh, 2023). We apply the OBS weight update procedure used in SparseGPT on the kept weights produced by both magnitude pruning and our approach Wanda. We evaluate the perplexity on both the calibration data and the validation data from WikiText.

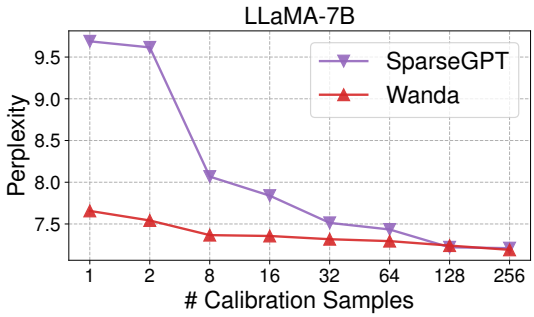

Figure 2: Ablation on calibration samples.

Results are summarized in Table 4. The result on magnitude pruning does validate the effectiveness of the weight update procedure, where it improves perplexity significantly on both data splits. However, when the pruned model is obtained by our method Wanda, it no longer brings improvement but leads to an increase of perplexity (from 7.26 to 7.32) on validation data, suggesting that in this case the updated model may overfit to the calibration samples.

		Weight Update	
Method	Data Split	✗	✓
Magnitude	Calibration	22.13	**13.45**
	Validation	17.59	**12.56**
Wanda	Calibration	8.62	**8.38**
	Validation	**7.26**	7.32

Table 4: Ablation on OBS based weight update.

Results are summarized in Table 4. The result on magnitude pruning does validate the effectiveness of OBS based weight update, where it improves perplexity significantly on both data splits. However, when the pruning mask is initially good, i.e. obtained by our method Wanda, the weight update procedure no longer brings improvement but leads to an increase of perplexity (from 7.26 to 7.32) on validation data, suggesting that in this case the updated pruned networks are overfitted to the calibration samples.

4. Conclusion

In this work, we propose a simple and effective method for pruning Large Language Models (LLMs). Inspired by the recent discovery of emergent large features in LLMs, our approach, termed Wanda (Pruning by **W**eights **and** **a**ctivations), removes weights with the smallest magnitudes multiplied by the corresponding input activation norms, on a per-output basis. Without the need for any retraining or weight update procedures, Wanda is able to identify effective sparse networks within pretrained LLMs. Wanda could be used as a general pruning approach beyond LLMs. We hope our work contributes to a better understanding of sparsity in LLMs.

Acknowledgments. We thank Yonghao Zhuang for valuable discussions. Mingjie Sun and Anna Bair were supported by funding from the Bosch Center for Artificial Intelligence.

References

Biderman, S., Schoelkopf, H., Anthony, Q., Bradley, H., O'Brien, K., Hallahan, E., Khan, M. A., Purohit, S., Prashanth, U. S., Raff, E., Skowron, A., Sutawika, L., and van der Wal, O. Pythia: A suite for analyzing large language models across training and scaling. *arXiv preprint arXiv:2304.01373*, 2023.

Blalock, D., Ortiz, J. J. G., Frankle, J., and Guttag, J. What is the state of neural network pruning? In *Proceedings of Machine Learning and Systems (MLSys)*, 2020.

Bondarenko, Y., Nagel, M., and Blankevoort, T. Understanding and overcoming the challenges of efficient transformer quantization. *arXiv:2109.12948*, 2021.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

Bubeck, S., Chandrasekaran, V., Eldan, R., Gehrke, J., Horvitz, E., Kamar, E., Lee, P., Lee, Y. T., Li, Y., Lundberg, S., Nori, H., Palangi, H., Ribeiro, M. T., and Zhang, Y. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*, 2023.

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 preprint arXiv:1803.05457*, 2018.

Cortes, C., Mohri, M., and Rostamizadeh, A. L2 regularization for learning kernels. In *Conference on Uncertainty in Artificial Intelligence*, 2009.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. ImageNet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Dettmers, T., Lewis, M., Belkada, Y., and Zettlemoyer, L. LLM.int8(): 8-bit matrix multiplication for transformers at scale. In *Advances in Neural Information Processing Systems*, 2022.

Dettmers, T., Svirschevski, R., Egiazarian, V., Kuznedelev, D., Frantar, E., Ashkboos, S., Borzunov, A., Hoefler, T., and Alistarh, D. Spqr: A sparse-quantized representation for near-lossless llm weight compression. *arXiv preprint arXiv:2306.03078*, 2023.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

Fan, A., Grave, E., and Joulin, A. Reducing transformer depth on demand with structured dropout. In *International Conference on Learning Representations*, 2020.

Frankle, J., Dziugaite, G. K., Roy, D. M., and Carbin, M. Stabilizing the lottery ticket hypothesis. In *International Conference on Machine Learning*, 2020.

Frantar, E. and Alistarh, D. Spdy: Accurate pruning with speedup guarantees. In *International Conference on Machine Learning*, 2022.

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

Frantar, E., Singh, S. P., and Alistarh, D. Optimal Brain Compression: A framework for accurate post-training quantization and pruning. In *Advances in Neural Information Processing Systems*, 2022.

Frantar, E., Ashkboos, S., Hoefler, T., and Alistarh, D. GPTQ: Accurate post-training compression for generative pretrained transformers. In *International Conference on Learning Representations*, 2023.

Gale, T., Elsen, E., and Hooker, S. The state of sparsity in deep neural networks. In *International Conference on Machine Learning*, 2019.

Gao, L., Tow, J., Biderman, S., Black, S., DiPofi, A., Foster, C., Golding, L., Hsu, J., McDonell, K., Muennighoff, N., Phang, J., Reynolds, L., Tang, E., Thite, A., Wang, B., Wang, K., and Zou, A. A framework for few-shot language model evaluation. *Version v0. 0.1. Sept*, 2021.

Han, S., Pool, J., Tran, J., and Dally, W. J. Learning both weights and connections for efficient neural networks. In *Advances in Neural Information Processing Systems*, 2015.

Han, S., Mao, H., and Dally, W. J. Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding. In *International Conference on Learning Representations*, 2016.

Hassibi, B., Stork, D. G., and Wolff, G. J. Optimal brain surgeon and general network pruning. In *IEEE International Conference on Neural Networks*, 1993.

Hoefler, T., Alistarh, D., Ben-Nun, T., Dryden, N., and Peste, A. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *arXiv preprint arXiv:2102.00554*, 2021.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2021.

Hubara, I., Chmiel, B., Island, M., Banner, R., Naor, S., and Soudry, D. Accelerated sparse neural training: A provable and efficient method to find N:M transposable masks. In *Advances in Neural Information Processing Systems*, 2021.

Jonathan, F. and Michael, C. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2019.

Kusupati, A., Ramanujan, V., Somani, R., Wortsman, M., Jain, P., Kakade, S., and Farhadi, A. Soft threshold weight reparameterization for learnable sparsity. In *International Conference on Machine Learning*, 2020.

Kwon, W., Kim, S., Mahoney, M. W., Hassoun, J., Keutzer, K., and Gholami, A. A fast post-training pruning framework for transformers. In *Advances in Neural Information Processing Systems*, 2022.

LeCun, Y., Denker, J. S., and Solla, S. A. Optimal brain damage. In *Advances in Neural Information Processing Systems*, 1989.

Lewkowycz, A. and Gur-Ari, G. On the training dynamics of deep networks with l2 regularization. In *Advances in Neural Information Processing Systems*, 2020.

Lin, J., Tang, J., Tang, H., Yang, S., Dang, X., and Han, S. Awq: Activation-aware weight quantization for llm compression and acceleration. *arXiv preprint arXiv:2306.00978*, 2023.

Liu, Z., Li, J., Shen, Z., Huang, G., Yan, S., and Zhang, C. Learning efficient convolutional networks through network slimming. In *International Conference on Computer Vision*, 2017.

Liu, Z., Sun, M., Zhou, T., Huang, G., and Darrell, T. Rethinking the value of network pruning. In *International Conference on Learning Representations*, 2019.

Liu, Z., Mao, H., Wu, C.-Y., Feichtenhofer, C., Darrell, T., and Xie, S. A convnet for the 2020s. In *Conference on Computer Vision and Pattern Recognition*, 2022.

Louizos, C., Welling, M., and Kingma, D. P. Learning sparse neural networks through l0 regularization. In *International Conference on Learning Representations*, 2018.

Luo, Z., Kulmizev, A., and Mao, X. Positional artefacts propagate through masked language model embeddings. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, August 2021.

Merity, S., Xiong, C., Bradbury, J., and Socher, R. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

Mihaylov, T., Clark, P., Khot, T., and Sabharwal, A. Can a suit of armor conduct electricity? a new dataset for open book question answering. *arXiv preprint arXiv:1809.02789*, 2018.

Mishra, A., Latorre, J. A., Pool, J., Stosic, D., Stosic, D., Venkatesh, G., Yu, C., and Micikevicius, P. Accelerating sparse deep neural networks. *arXiv preprint arXiv:2104.08378*, 2021.

Molchanov, P., Tyree, S., Karras, T., Aila, T., and Kautz, J. Pruning convolutional neural networks for resource efficient inference. In *International Conference on Learning Representations*, 2017.

OpenAI. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Pool, J. and Yu, C. Channel permutations for n:m sparsity. In *Advances in Neural Information Processing Systems*, 2021.

Puccetti, G., Rogers, A., Drozd, A., and Dell'Orletta, F. Outliers dimensions that disrupt transformers are driven by frequency. *arXiv preprint arXiv:2205.11380*, 2022.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 2020.

Renda, A., Frankle, J., and Carbin, M. Comparing rewinding and fine-tuning in neural network pruning. In *International Conference on Learning Representations*, 2020.

Sakaguchi, K., Bras, R. L., Bhagavatula, C., and Choi, Y. Winogrande: An adversarial winograd schema challenge at scale. *arXiv preprint arXiv:1907.10641*, 2019.

Sanh, V., Wolf, T., and Rush, A. M. Movement pruning: Adaptive sparsity by fine-tuning. In *Advances in Neural Information Processing Systems*, 2020.

Schaeffer, R., Miranda, B., and Koyejo, S. Are emergent abilities of large language models a mirage? *arXiv preprint arXiv:2304.15004*, 2023.

Singh, S. P. and Alistarh, D. Woodfisher: Efficient second-order approximation for neural network compression. In *Advances in Neural Information Processing Systems*, 2020.

Timkey, W. and Schijndel, M. v. All bark and no bite: Rogue dimensions in transformer language models obscure representational quality. *arXiv:2109.04404*, 2021.

Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., and Jégou, H. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning*, 2021.

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. *arXiv preprint arXiv:2302.13971*, 2023.

Vaswani, A., Sharzeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polusukhin, I. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. R. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.

Wei, J., Tay, Y., Bommasani, R., Raffel, C., Zoph, B., Borgeaud, S., Yogatama, D., Bosma, M., Zhou, D., Metzler, D., Chi, E. H., Hashimoto, T., Vinyals, O., Liang, P., Dean, J., and Fedus, W. Emergent abilities of large language models. In *Transactions on Machine Learning Research*, 2022a.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q., and Zhou, D. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, 2022b.

Xia, M., Artetxe, M., Zhou, C., Victoria Lin, X., Pasunuru, R., Chen, D., Zettlemoyer, L., and Stoyanov, V. Training trajectories of language models across scales. *arXiv preprint arXiv:2212.09803*, 2022a.

Xia, M., Zhong, Z., and Chen, D. Structured pruning learns compact and accurate models. In *Association for Computational Linguistics (ACL)*, 2022b.

Xiao, G., Lin, J., Seznec, M., Wu, H., Demouth, J., and Han, S. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International Conference on Machine Learning*, 2023.

Xiuying, W., Zhang, Y., Zhang, X., Gong, R., Zhang, S., Zhang, Q., Yu, F., and Liu, X. Outlier suppression: Pushing the limit of low-bit transformer language models. In *Advances in Neural Information Processing Systems*, 2022.

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., et al. OPT: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

Zhou, A., Ma, Y., Zhu, J., Liu, J., Zhang, Z., Yuan, K., Sun, W., and Li, H. Learning N:M fine-grained structured sparse neural networks from scratch. In *International Conference on Learning Representations*, 2021.

Zhu, F., Pool, J., Andersch, M., Appleyard, J., and Xie, F. Sparse persistent rnns: Squeezing large recurrent networks on-chip. In *International Conference on Learning Representations*, 2018.

Zhu, M. and Gupta, S. To prune, or not to prune: exploring the efficacy of pruning for model compression. *arXiv preprint arXiv:1710.01878*, 2017.

A. Experimental Details

A.1. Setup for Pruning LLMs

Our experiments are conducted on NVIDIA RTX A6000 GPUs with 50 GB memory. For pruning LLaMA models, we first load these models onto GPUs in 16-bit floating-point format. All subsequent procedures (e.g. pruning) are conducted directly on GPUs. For zero-shot tasks, we use the evaluation framework from https://github.com/EleutherAI/lm-evaluation-harness/.

Models. We evaluate Wanda on the LLaMA (Touvron et al., 2023) model family, a series of Transformer language models at various parameter levels, often referred to as LLaMA-7B/13B/30B/65B. We apply our pruning method to all four LLaMA models. We note that our approach is not limited to LLaMA, but is applicable to any architectures that consist of many linear layers, including other Transformer-based LLMs. We provide results for other LLM families in Appendix E.

Evaluation. We first measure the performance of pruned networks by their language modeling ability: *perplexity* computed on a held-out validation set. Following previous works on LLM compression (Xiao et al., 2023; Frantar & Alistarh, 2023; Frantar et al., 2023), we report the perplexity metric on WikiText (Merity et al., 2016) validation set. While perplexity has been shown to be a stable and robust metric for language models (Xia et al., 2022a), we also evaluate their zero-shot ability. We use the public evaluation benchmark EleutherAI LM Harness (Gao et al., 2021). For zero-shot performance, we evaluate on seven common sense benchmarks: BoolQ (Clark et al., 2019), RTE (Wang et al., 2018), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2019), ARC Easy and Challenge (Clark et al., 2018), and OpenbookQA (Mihaylov et al., 2018). We report the accuracy on each benchmark as well as the overall mean accuracy.

Baselines. We compare our method Wanda to two prior pruning methods:

- Magnitude pruning (Han et al., 2015) is a simple and strong baseline in which weights are discarded based on their magnitudes. We follow the previous practice of magnitude pruning on Transformers (Gale et al., 2019; Sanh et al., 2020), where weights are compared locally among the current linear layer. Magnitude pruning has been demonstrated to outperform many other pruning methods (Gale et al., 2019; Blalock et al., 2020).
- SparseGPT (Frantar & Alistarh, 2023) is a second-order pruning method based on solving a layer-wise reconstruction problem. To scale existing second-order based approaches (Frantar et al., 2022) to LLMs, an efficient weight update procedure was proposed that iterates between weight removal and weight update at each layer. To our knowledge, it is the only pruning method so far that has been demonstrated to work with LLMs.

Both Wanda and SparseGPT require calibration data to estimate certain input statistics (see Table 1). To control this variable factor, we use the *exact same* set of calibration data as SparseGPT, which consists of 128 sequences (2048 tokens each) sampled from the first shard of the C4 training data (Raffel et al., 2020).

Sparsity. For all pruning methods, we follow the setup of SparseGPT (Frantar & Alistarh, 2023), where a uniform sparsity is imposed for all layers and there is no subsequent retraining. We skip the first embedding layer and the final classification head, as is common in pruning Transformers (Sanh et al., 2020; Frantar & Alistarh, 2023). Our primary approach to induce sparsity is through unstructured pruning. Considering the potential need for practical speedup, we also conduct evaluations on structured N:M sparsity (Zhou et al., 2021; Hubara et al., 2021). Specifically, we provide comparisons on 4:8 and 2:4 sparsity patterns. The magnitude pruning baseline is extended to structured N:M sparsity in a similar spirit to our method which is described in the previous section.

A.2. Setup for Additional Analysis

Pruning Image Classifiers. We use the pretrained ConvNeXt-B model available at https://github.com/facebookresearch/ConvNeXt. For ViT, we use the DeiT-B model from https://github.com/facebookresearch/deit. The ConvNeXt-B and DeiT-B classifiers have 89M and 86M parameters respectively. As described in Section F, we focus on pruning linear layers, which comprise the majority of the parameters in these two architectures. For ConvNeXt, the 1×1 convolutions in the its blocks can be viewed as linear layers, which account for approximately 95% of the total model parameters. For calibration data on ImageNet, we sample 4096 images from the training set, using them as the calibration set for our entire analysis. We find 4096 samples leads to a stable result for our pruning metric, beyond which we notice only marginal effect.

LoRA Fine-tuning. We adopt the LoRA fine-tuning code on LLaMA available at https://github.com/tloen/alpaca-lora. Our fine-tuning objective is the pretraining autoregressive language modeling objective. For the LoRA adapter, we update two weight matrices, W_q and W_v, in the self-attention module with a rank of 8. The fine-tuning process

is conducted on the C4 training set, operating within a restricted computational budget that involves a single A6000 GPU and a time constraint of 5 hours.

B. Preliminaries

Magnitude Pruning (Han et al., 2015; 2016) is a standard pruning technique to induce sparsity in neural networks. Different from structured pruning approaches (Xia et al., 2022b; Liu et al., 2017), magnitude pruning removes individual weights based on their magnitudes, where weights with magnitudes below a certain threshold are removed. In practice, this threshold is typically determined by comparing weights globally (Liu et al., 2019) or layer-wise (Zhu & Gupta, 2017). Despite its simplicity, magnitude pruning has been used to find extremely sparse networks (Jonathan & Michael, 2019; Zhu & Gupta, 2017) and now stands out as a strong baseline approach (Gale et al., 2019; Blalock et al., 2020; Hoefler et al., 2021) for neural network sparsification.

Emergent Large Magnitude Features is a property specific to Transformer (Vaswani et al., 2017) based large language models. Once models reach a certain scale (in practice, around 6B parameters), a small set of hidden state features emerges with significantly larger magnitudes than the remaining ones. The existence of these outlier features in large language models is demonstrated in Dettmers et al. (2022). These outlier features exhibit several intriguing characteristics. First, they have very large magnitudes, about 100 times larger than typical hidden state values (Xiao et al., 2023). Second, they are usually sparse and exist in certain feature dimensions. Finally, these outlier features are essential for the predictive capability of LLMs: zeroing out these features at inference time results in significant degradation of language modeling performance (Dettmers et al., 2022).

C. Structured N:M Sparsity.

While our method Wanda so far has been developed for unstructured sparsity, it can be easily extended to structured N:M sparsity (Zhou et al., 2021; Hubara et al., 2021; Pool & Yu, 2021). N:M sparsity requires that at most N out of every M contiguous weights to be non-zero. It can leverage NVIDIA's sparse tensor cores (Mishra et al., 2021) to accelerate matrix multiplication in practice. Wanda can be naturally extended to structured N:M sparsity, where we only need to compare weights using the same metric among every M consecutive weights, for all weights connected to an output.

D. Connection to SparseGPT (Frantar & Alistarh, 2023)

We discuss Wanda's connection with a few existing works. SparseGPT (Frantar & Alistarh, 2023) formalizes the problem of pruning LLMs by solving a local layer-wise reconstruction problem, where their pruning metric and weight update procedure is inspired from Optimal Brain Surgeon (OBS) (Hassibi et al., 1993). The pruning metric in SparseGPT is:

$$\mathbf{S}_{ij} = \left[|\mathbf{W}|^2 / \mathrm{diag}\left((\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\right)\right]_{ij} \tag{3}$$

Here $\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}$ in the denominator is the Hessian $\mathbf{H}$ for the layer-wise reconstruction problem and λ is the Hessian dampening factor to avoid the collapse of inverse computation. With careful inspection, we observe that our metric in Equation 1 is similar to the above when λ is 0 and only the diagonal elements of the Hessian matrix $\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}$ are retained. This is because the diagonal of $\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}$ is $\mathrm{diag}(\|\mathbf{X}_j\|_2^2)$, and the denominator in Equation 3 is now simplified to $(\|\mathbf{X}_j\|_2^2)^{-1}$. The resulting metric is the square of our proposed metric. This simplification substantially reduces the required computation of weight importance, eliminating the need for computing any matrix inverses.

In the 1980s, LeCun et al. (LeCun et al., 1989) have set up a pioneering framework for neural network pruning named Optimal Brain Damage (OBD). It uses second-order information but ignores off-diagonal elements in Hessians for faster approximation. Later, Optimal Brain Surgeon (OBS) (Hassibi et al., 1993) develops upon OBD partly by taking into account the off-diagonal elements. Wanda can be seen as a renaissance of the pioneering work of Optimal Brain Damage (OBD) (LeCun et al., 1989) – it may be viewed as applying a process similar to OBD to each neuron, with *local* output reconstruction as the objective function, whereas the original OBD uses the *global* training objective.

E. Additional Results

E.1. Varying Sparsity Levels

We conduct experiments with varying levels of sparsity for unstructured pruning, the results of which are depicted in Figure 3. It can be seen that Wanda and SparseGPT show similar trends of perplexity increase as the sparsity level gets

higher. However, magnitude pruning displays a considerably more severe degradation trend.

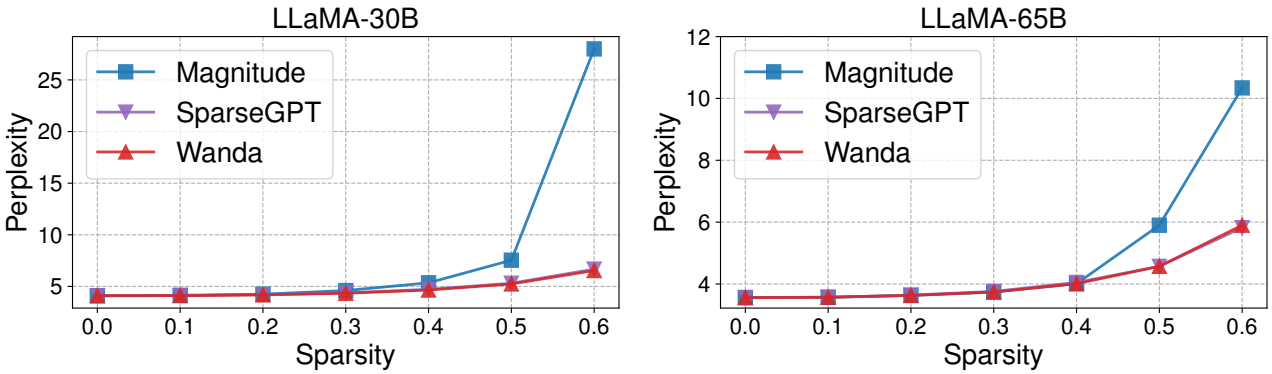

Figure 3: Results of the two largest LLaMA models with varying sparsity levels, where we compare the degradation trend of pruned networks between our approach Wanda and baseline approaches.

E.2. Zero-Shot Tasks

We evaluate pruned models on downstream zero-shot tasks via prompting. Results are summarized in Table 5, where models are pruned to unstructured 50% sparsity. Averaging the accuracy over the 7 tasks under consideration, the Wanda method is able to identify effective pruned networks, showing competitiveness with baseline methods. Note that as the model gets larger in size, the accuracy drop compared to the original dense model keeps getting smaller. For task-wise performance, we observe that there are certain tasks where our approach Wanda gives *consistently* better results across all LLaMA models, i.e. HellaSwag, ARC-c and OpenbookQA, while among the remaining tasks, there is not a fixed superior between the two methods.

Params	Method	BoolQ	RTE	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA	Mean
7B	Dense	71.7	53.4	58.3	68.0	67.7	38.6	28.0	55.1
	Magnitude	53.5	56.3	44.6	54.5	53.7	32.3	21.6	45.2
	SparseGPT	**71.5**	**56.8**	52.7	**65.8**	**64.3**	36.0	25.0	**53.2**
	Wanda	70.3	53.3	**52.9**	64.7	64.1	**37.0**	**26.4**	52.7
13B	Dense	68.3	65.3	60.8	70.0	73.6	44.0	30.6	58.9
	Magnitude	62.1	45.8	44.3	58.9	49.8	33.1	27.4	45.9
	SparseGPT	66.7	52.0	55.1	70.9	66.9	39.2	26.0	53.8
	Wanda	**67.1**	**61.0**	**56.6**	**71.7**	**68.4**	**41.7**	**28.0**	**56.4**
30B	Dense	66.9	61.4	64.8	72.4	75.3	46.9	29.4	59.6
	Magnitude	66.6	53.4	49.7	63.4	68.8	40.0	26.2	52.6
	SparseGPT	**71.0**	61.4	61.1	**72.0**	73.9	46.0	31.2	59.5
	Wanda	70.3	**66.8**	**62.3**	71.1	**74.8**	**46.5**	**32.4**	**60.6**
65B	Dense	81.8	71.8	65.2	76.9	75.4	47.2	36.4	65.0
	Magnitude	79.8	66.8	62.5	70.3	70.4	46.7	31.4	61.1
	SparseGPT	81.2	**70.4**	62.5	**74.1**	**74.8**	44.7	32.2	62.8
	Wanda	**82.2**	69.1	**64.5**	73.9	74.6	**45.9**	**32.8**	**63.3**

Table 5: Accuracies (%) for 7 zero-shot tasks with unstructured 50% sparsity.

E.3. Pruning Speed

The theoretical computational complexity of Wanda is lower than SparseGPT (Table 1). Here we compare their empirical pruning speed. We measure the total pruning time only (excluding the forward pass process shared by both methods) on NVIDIA A6000 GPUs. We present the results in Table 6. Wanda incurs negligible time overhead compared with SparseGPT. The fast speed is particularly useful when pruning needs to be performed on a real-time basis, e.g., processing different samples with different sparsity levels.

	LLaMA			
Method	7B	13B	30B	65B
SparseGPT	203.1	339.0	810.3	1353.4
Wanda	**0.54**	**0.91**	**2.9**	**5.6**

Table 6: Comparison of time overhead (in seconds) between Wanda and SparseGPT, excluding the shared forward pass process.

E.4. Pruning Configuration

Wanda differs from previous methods in both the pruning metric and the pruning granularity. We conduct experiments on a variety of pruning metrics and grnularities to ablate their effectiveness. The three pruning metrics can be found in Table 1. SparseGPT adopts a local granularity level inside a layer, where weights connected to 128 consecutive *input* channels form a group. Wanda groups weights connected with a single *output* channel. Therefore, we ablate two blocksize options (128 and 1) and the input/output choice. For simplicity, we use (input/output, blocksize) to denote the local pruning granularity level, e.g., (input, 1). For this experiment, we do not perform weight update for SparseGPT to focus on metric and granularity.

	Pruning Granularity				
Pruning Metric	layer	(input, 1)	(input, 128)	(output, 1)	(output, 128)
Magnitude: $\|\mathbf{W}_{ij}\|$	<u>17.29</u>	**8.86**	16.82	13.41	17.47
SparseGPT: $\left[\|\mathbf{W}\|^2/\text{diag}(\mathbf{H}^{-1})\right]_{ij}$	7.91	8.86	<u>8.02</u>	**7.41**	7.74
Wanda: $\|\mathbf{W}_{ij}\| \cdot \|\mathbf{X}_j\|$	7.95	8.86	8.12	**<u>7.26</u>**	7.71

Table 7: Ablation on pruning metric and granularity. **Bold** results denote the best granularity found for each pruning metric. <u>Underscored</u> results indicate the default granularity used by each method.

The results are shown in Table 7. Wanda's default configuration delivers the best pruned model (perplexity 7.26). Interestingly, for the magnitude metric, comparing weights of the same input neuron (input, 1) yields a perplexity of 8.86, significantly better than other granularity options. Three methods also produce equivalent pruning results as under this granularity – the input is the same, thus weight ranking only depends on weight magnitude. This finding further highlights the importance of using a proper pruning granularity for pruning LLMs, even for the classical magnitude pruning.

E.5. Pythia and OPT

While prior public LLM models may not be as powerful as LLaMA, we test the generality of our approach by evaluating it on two representative LLM families before the release of LLaMA, namely Pythia (Biderman et al., 2023) and OPT (Zhang et al., 2022). We select two models with approximately the same level of total parameters: Pythia-12b [1] and OPT-13b [2]. The results (unstructured sparsity) are shown in Table 8. We can see that magnitude pruning deteriorates significantly faster on Pythia-12b and OPT-13b in comparison to LLaMA models. For instance, even a sparsity ratio of 20% pushes the perplexity of pruned models to a meaningless level (greater than 1000). In contrast, our method Wanda consistently finds *exact* and *effective* sparse networks within both pretrained LLMs.

F. Analysis

In this section, we first study whether Wanda can be a general pruning approach beyond LLMs. Next, we show exploratory results on using parameter efficient fine-tuning techniques to recover performance drop during the pruning procedure.

While the main focus of this work is on pruning LLMs, the surprising effectiveness of Wanda leads to a clear question: how would Wanda perform against magnitude pruning on tasks where the latter has been widely used? We thus conduct a study on ImageNet-1K (Deng et al., 2009), a standard image classification task where magnitude pruning has been extensively studied and accepted as a strong baseline (Gale et al., 2019; Blalock et al., 2020).

We consider two modern vision architectures: ConvNeXt (Liu et al., 2022) and Vision Transformer (ViT) (Dosovitskiy et al.,

[1] https://huggingface.co/EleutherAI/pythia-12b
[2] https://huggingface.co/facebook/opt-13b

Model	Dense	Pruning Method	Weight Update	Sparsity				
				10%	20%	30%	40%	50%
Pythia-12b	8.59	Magnitude	✗	127.76	2e5	7e5	2e5	3e5
		SparseGPT	✓	**8.59**	8.65	8.86	9.39	**11.02**
		Wanda	✗	**8.59**	**8.60**	**8.85**	**9.31**	11.27
OPT-13b	10.13	Magnitude	✗	14.45	9e3	1e4	1e4	1e4
		SparseGPT	✓	10.11	10.10	10.12	**10.35**	**11.19**
		Wanda	✗	**10.09**	**10.07**	**10.09**	10.63	11.42

Table 8: Results for pruning Pythia and OPT models, where we show the perplexity evaluation on WikiText. Notice that magnitude pruning fails catastropically even at low sparsity levels (e.g. 20%) for both models.

2021). We choose these two architectures mainly for two reasons: first, as LLMs are based on Transformers (Vaswani et al., 2017), we would like to test if our observations on LLMs so far still hold on Transformers for other tasks; second, as we are evaluating on image classification, we are interested in examining how they work with ConvNet models, with ConvNeXt being a representative architecture.

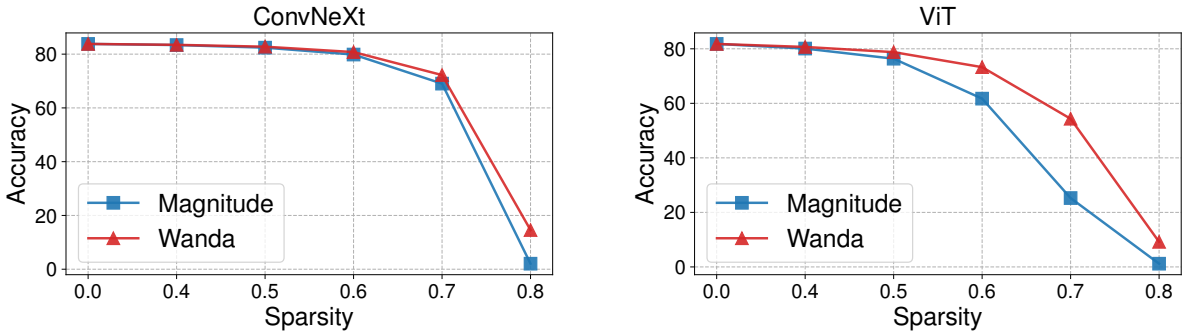

Figure 4: Results for pruning image classification models.

We use two ImageNet-1K pretrained models: ConvNeXt-B and ViT-B, which have a top-1 accuracy of 83.8% and 81.8% (Touvron et al., 2021) respectively. We prune the linear layers only (for ConvNeXt, this includes equivalent 1×1 convolution layers). We find that for both models, pruning uniformly on a per layer basis is *consistently* better than a per output basis. We thus compare our proposed metric and the standard magnitude metric by pruning on a *per layer* basis. Results are shown in Figure 4. Our novel pruning metric leads to better results than magnitude pruning, especially at high sparsities (e.g. 70%, 80%). However, we find differences in the accuracy of pruned networks can easily be mitigated via retraining for a few epochs, indicating that Wanda might be more suitable for cases where retraining is not computationally feasible, such as in LLMs.

We now present detailed results on pruning image classifiers in Figure 5, where we report the accuracy of pruned models without any subsequent fine-tuning. For clarification purposes, in our analysis, we use Magnitude and Wanda to refer to the magnitude metric and our proposed metric in Equation 1, respectively. We can see that for both ConvNeXt and DeiT, pruning in a layerwise fashion is better than pruning per output (denoted as 'row' in the figure legend).

In addition, we extend our analysis to a ViT model from the original paper Dosovitskiy et al. (2021): ViT-L trained on ImageNet-1K with a top-1 accuracy 78.9%. To differentiate from DeiT, we refer to this pretrained model as original ViT. Results are shown in Figure 6. Interestingly, in this case, we observe that pruning on a per output basis is superior to layerwise pruning, implying that the choice of pruning granularity may be different for different models.

Notably, we also find that the accuracy differences between Wanda and magnitude pruning in pruning image classifiers can be effectively mitigated by fine-tuning for a few epochs.

F.1. LoRA Fine-tuning

We explore using parameter efficient fine-tuning (PEFT) techniques to recover performance of pruned LLM models. We use a popular PEFT method LoRA (Hu et al., 2021), which has been widely adopted for task specific fine-tuning of LLMs.

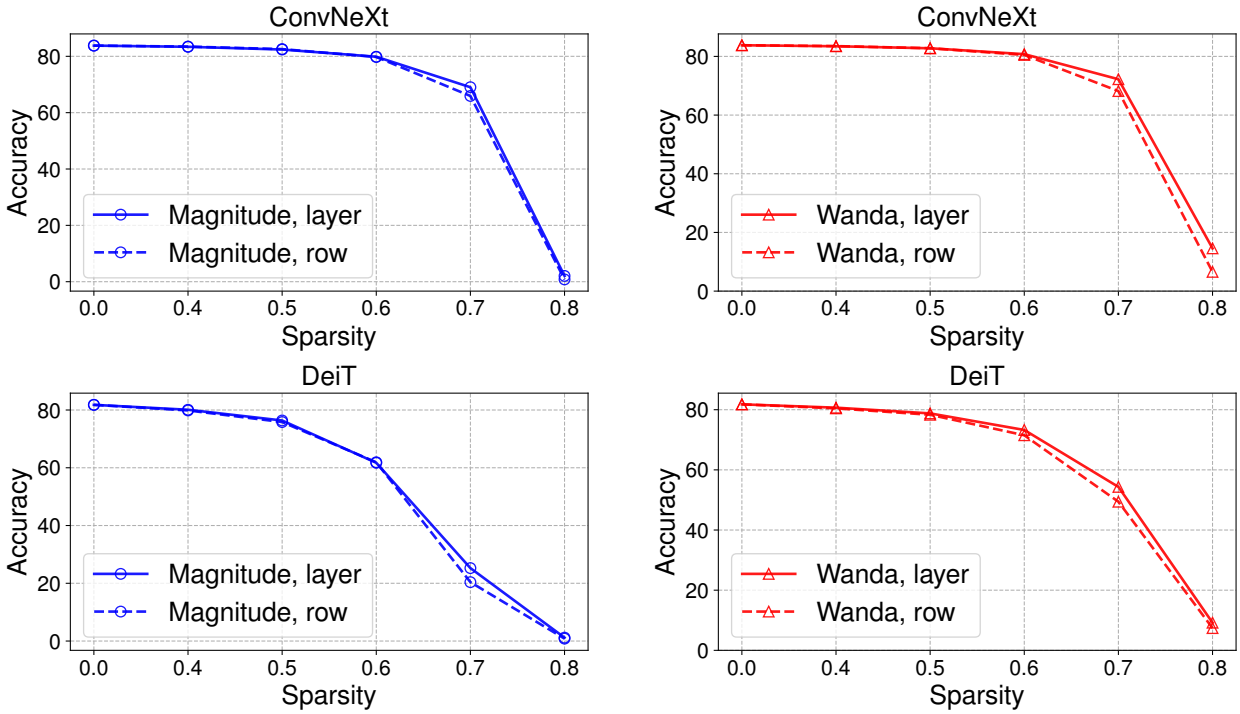

Figure 5: Results for pruning two image classifiers: ConvNeXt-B and DeiT-B.

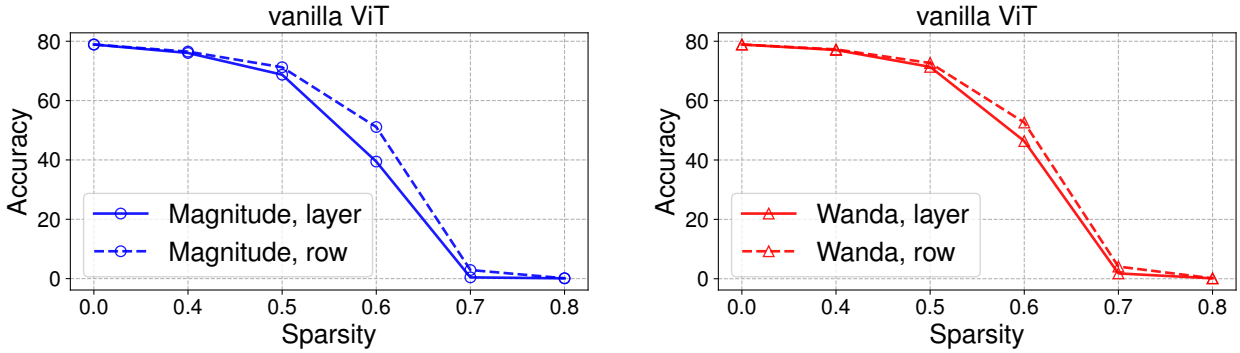

Figure 6: Results for pruning the image classifier ViT-L in Dosovitskiy et al. (2021).

However, here we are interested in recovering the performance loss of LLMs during pruning, thus we perform a more general "fine-tuning" where the pruned networks are trained with an autoregressive objective on C4 dataset. We enforce a limited computational budget (1 GPU and 5 hours). We find that we are able to restore performance of pruned LLaMA-7B (unstructured 50% sparsity) with a non-trivial amount, reducing zero-shot WikiText perplexity from 7.26 to 6.87. The additional parameters introduced by LoRA is only 0.06%, leaving the total sparsity level still at around 50% level.

G. Related Work

Network Pruning and Sparsity. Pruning is a popular technique for compressing neural networks through the elimination of weights, yielding sparse networks (LeCun et al., 1989; Hassibi et al., 1993; Han et al., 2015; Liu et al., 2019; Zhu et al., 2018). Pruning methods can be broadly categorized into *structured* and *unstructured* approaches. Structured pruning methods (Xia et al., 2022b; Liu et al., 2017; Molchanov et al., 2017; Fan et al., 2020) remove entire structured components of a network, facilitating efficient GPU speedups, while unstructured methods like magnitude pruning operate at the individual weight level, maintaining performance even at higher sparsity levels. Existing pruning methods usually require either modifications to the training procedure (Louizos et al., 2018; Sanh et al., 2020; Gale et al., 2019; Kusupati et al., 2020), retraining the pruned networks to regain accuracy (Han et al., 2015; 2016; Liu et al., 2019), or an even more computationally intensive iterative retraining process (Jonathan & Michael, 2019; Zhu & Gupta, 2017; Renda et al., 2020; Frankle et al.,

2020). However, scaling these methods to LLMs with billions of parameters presents a challenge, as the required training process demands substantial computational resources (Touvron et al., 2023; Zhang et al., 2022).

Pruning with Limited Data. Most related to our approach is a recent line of work on pruning with limited data (Hubara et al., 2021; Frantar et al., 2022; Frantar & Alistarh, 2022; Kwon et al., 2022). Such methods require no modification to the original training procedure and also no retraining of the pruned networks on the full training dataset. The primary aim of these methods is to preserve performance during the pruning procedure, assuming access to a limited and small amount of data, also referred to as the calibration data. In order to mitigate the accuracy drop, a layer-wise reconstruction problem (Hubara et al., 2021) is solved to minimize the change of output evaluated on the calibration data. Existing popular solvers (Frantar et al., 2022; Singh & Alistarh, 2020) for the layer-wise reconstruction problem rely on heavy computation of second-order Hessian inverses, which do not scale to the large hidden state size of LLMs. SparseGPT (Frantar & Alistarh, 2023) develops an efficient weight update procedure for LLMs via synchronized second-order Hessian updates.

Emergent Properties of LLMs. Our work is also related to recent studies on the existence of large magnitude outlier features in Transformer-based language models (Timkey & Schijndel, 2021; Bondarenko et al., 2021; Xiuying et al., 2022; Luo et al., 2021; Puccetti et al., 2022). Dettmers et al. (2022) demonstrates that when LLMs exceed a certain parameter scale (e.g. 6B), large magnitude features start to emerge and strongly affect all layers, which is considered as one of the emergent properties of LLMs (Dettmers et al., 2022; Wei et al., 2022a; Schaeffer et al., 2023). They further identify these emergent features as the cause of existing quantization methods' failures. This observation has spurred the development of various quantization schemes (Dettmers et al., 2022; Xiao et al., 2023; Lin et al., 2023; Dettmers et al., 2023) tailored specifically for LLMs to manage these outlier features. Our work extends this understanding, demonstrating that these emergent large magnitude features should also serve as pivotal indicators for determining which weights to prune in LLMs.