Mohammed VI Polytechnic University

College of Computing

Wanda: A Simple and Effective Pruning Approach for Large Language Models

Reproducing Unstructured Pruning Experiments

AI Course Project

CSCI-B-M315 - Advanced AI Topics – 2025/2026

Authors:
Zineb ABERCHA
Omar Alfarouq BOUHADI

Supervised by:
Prof. Hamza KEURTI

Code Repository:

https://github.com/tweizy/wanda-prunning-experiment

January 2026

Abstract

Large Language Models (LLMs) perform well across many natural language processing tasks, but their billions of parameters make deployment difficult. This report reproduces and validates the Wanda (Weights AND Activations) pruning technique proposed by Sun et al. (2024), testing its effectiveness for unstructured pruning on LLMs. We ran experiments on LLaMA-2-7B and LLaMA-3.1-8B, measuring performance at three sparsity levels (30%, 50%, and 70%) using perplexity on WikiText-2 and zero-shot accuracy on five benchmark tasks. Our results show that Wanda outperforms magnitude-based pruning in most cases, with large improvements at 50% sparsity: 45.8% lower perplexity for LLaMA-2-7B and 76.2% lower perplexity for LLaMA-3.1-8B compared to magnitude pruning. We also discuss what happens at extreme sparsity levels and explain why activation-aware pruning works better than using weight magnitude alone.

Contents

1 Introduction

1.1 Motivation

The emergence of Large Language Models (LLMs) such as GPT-4, LLaMA, and Claude has changed how we approach natural language processing. These models can generate text, reason through problems, and complete various tasks. However, they typically contain billions of parameters, which means they need a lot of computational power to run. For example, the LLaMA-2-7B model has about 6.7 billion parameters and needs at least 14GB of GPU memory just to load it in half-precision format.

This high resource requirement makes it hard to deploy these models on edge devices, mobile phones, or in situations where cloud computing costs matter. As a result, techniques for making models smaller (like quantization, knowledge distillation, and pruning) have become important for practical use.

1.2 The Pruning Challenge

Neural network pruning aims to remove redundant parameters while preserving model accuracy. Traditional approaches include:

- **Magnitude Pruning**: Removes weights with the smallest absolute values, assuming small weights contribute less to model output.
- **Gradient-based Pruning**: Uses gradient information to identify important weights.
- **Second-order Methods**: Leverage Hessian information to estimate weight importance (e.g., OBS, OBD).

While magnitude pruning is simple and computationally efficient, it suffers from a fundamental limitation: **weight magnitude alone does not determine importance**. A small weight connected to highly activated inputs may contribute more to the model's output than a large weight connected to rarely-activated inputs.

1.3 Contributions of This Work

In this project, we reproduce and validate the Wanda pruning method [Sun et al., 2024], which addresses the limitations of magnitude pruning by incorporating input activation statistics. Our specific contributions include:

1. Implementation of both Wanda and magnitude pruning algorithms for LLaMA-family models.
2. Comprehensive evaluation on LLaMA-2-7B and LLaMA-3.1-8B across three sparsity levels.
3. Analysis of perplexity degradation on WikiText-2 and zero-shot accuracy on five benchmark tasks.
4. Discussion of failure modes at extreme sparsity and limitations of unstructured pruning.

2 Background and Related Work

2.1 Large Language Model Architecture

Modern LLMs are predominantly based on the Transformer architecture [Vaswani et al., 2017], consisting of stacked decoder layers. Each layer contains:

- **Multi-Head Self-Attention**: Computes attention weights across input tokens.
- **Feed-Forward Network (FFN)**: Two linear transformations with a non-linear activation.
- **Layer Normalization**: Stabilizes training and inference.

The LLaMA model family [Touvron et al., 2023a,b] uses the following architectural choices:

- Pre-normalization using RMSNorm instead of LayerNorm
- SwiGLU activation function in the FFN
- Rotary Position Embeddings (RoPE)
- No bias terms in linear layers

2.2 Neural Network Pruning

Pruning techniques can be categorized along several dimensions:

2.2.1 Structured vs. Unstructured Pruning

- **Structured Pruning**: Removes entire neurons, channels, or attention heads. Results in directly smaller weight matrices that can be accelerated on standard hardware.
- **Unstructured Pruning**: Removes individual weights, creating sparse weight matrices. Achieves higher sparsity at equivalent accuracy but requires specialized hardware or sparse kernels for acceleration.

This work focuses on **unstructured pruning**, following the Wanda paper's experimental setup.

2.2.2 Pruning Metrics

The choice of importance metric determines which weights are removed:

$$S_{\text{magnitude}}(w_{ij}) = |w_{ij}| \tag{1}$$

$$S_{\text{Wanda}}(w_{ij}) = |w_{ij}| \cdot \|X_j\|_2 \tag{2}$$

where w_{ij} is the weight connecting input j to output i, and X_j represents the j-th input feature across calibration samples.

2.3 The Wanda Method

Sun et al. [Sun et al., 2024] proposed Wanda (Weights AND Activations), observing that:

1. LLMs exhibit emergent large-magnitude features in hidden states.
2. These features cause certain input channels to have significantly higher activation norms.
3. Weights connected to these high-activation channels are more important, regardless of their magnitude.

The key insight is that the importance of a weight should be measured by its potential contribution to the output, not just its magnitude:

$$\text{Output contribution} \propto |w_{ij}| \times (\text{how often input } j \text{ is activated}) \tag{3}$$

Wanda requires only a small calibration set (typically 128 samples) to compute activation statistics, making it a highly efficient one-shot pruning method that requires no retraining.

2.4 Related Methods

- **SparseGPT** [Frantar & Alistarh, 2023]: Uses approximate second-order information with similar accuracy but higher computational cost.
- **GPTQ** [Frantar et al., 2022]: Quantization method using similar calibration approach.
- **LLM-Pruner** [Ma et al., 2023]: Structured pruning approach for LLMs.

3 Methodology

3.1 Pruning Algorithm

Our implementation follows the Wanda algorithm as described in Algorithm 1.

Algorithm 1 Wanda Pruning Algorithm

Require: Model $\mathcal{M}$, Calibration data $\mathcal{D}$, Target sparsity s
Ensure: Pruned model $\mathcal{M}'$

Phase 1: Capture layer inputs
for each layer l in $\mathcal{M}$ **do**
    Attach hook to capture input activations
end for
Forward pass calibration samples through $\mathcal{M}$
Phase 2: Layer-wise pruning
for each layer l in $\mathcal{M}$ **do**
    for each linear sublayer L in layer l **do**
        $X \leftarrow$ captured inputs to L across all samples
        $\text{norms}_j \leftarrow \sqrt{\sum_{\text{samples}} X_j^2}$ for each input channel j
        $W \leftarrow$ weight matrix of L
        $S_{ij} \leftarrow |W_{ij}| \times \text{norms}_j$ ▷ Wanda importance score
        threshold $\leftarrow$ s-th percentile of S per row
        $W_{ij} \leftarrow 0$ where $S_{ij} <$ threshold
    end for
    Forward layer l to update inputs for next layer
end for
return $\mathcal{M}'$

3.2 Implementation Details

Our implementation incorporates the following design choices:

1. **Layer-by-layer processing**: Activations are propagated through each pruned layer before processing the next, ensuring accurate activation statistics.

2. **Per-row sparsity**: Sparsity is applied uniformly across each row of the weight matrix, ensuring balanced capacity reduction.
3. **Stable sorting**: We use PyTorch's stable sort to ensure deterministic pruning decisions when weights have equal importance scores.
4. **Numerical stability**: A small epsilon (10^{-6}) is added before taking square roots to prevent numerical issues.

3.3 Baseline: Magnitude Pruning

For comparison, we implement standard magnitude pruning:

$$S_{\text{magnitude}}(w_{ij}) = |w_{ij}| \tag{4}$$

The same per-row sparsity allocation and layer-by-layer processing are applied for fair comparison.

3.4 Evaluation Metrics

3.4.1 Perplexity

Perplexity measures how well the model predicts a held-out test set:

$$\text{PPL} = \exp\left(-\frac{1}{N}\sum_{i=1}^{N}\log P(x_i|x_{<i})\right) \tag{5}$$

Lower perplexity indicates better language modeling capability. We evaluate on the WikiText-2 test set.

3.4.2 Zero-Shot Accuracy

We evaluate zero-shot performance on five benchmark tasks using the LM Evaluation Harness [Gao et al., 2023]:

- **PIQA** [Bisk et al., 2020]: Physical intuition QA
- **HellaSwag** [Zellers et al., 2019]: Commonsense natural language inference
- **ARC-Easy** [Clark et al., 2018]: Science question answering
- **BoolQ** [Clark et al., 2019]: Boolean question answering
- **RTE** [Dagan et al., 2005]: Recognizing textual entailment

4 Experimental Setup

4.1 Models

We evaluate two models from the LLaMA family:

Table 1: Model specifications

Model	Parameters	Layers	Hidden Size	Vocabulary
LLaMA-2-7B	6,738,415,616	32	4,096	32,000
LLaMA-3.1-8B	8,030,261,248	32	4,096	128,256

4.2 Dataset and Calibration

- **Calibration Set**: 64 random sequences from WikiText-2 training split
- **Sequence Length**: 4,096 tokens
- **Evaluation**: WikiText-2 test split for perplexity

4.3 Sparsity Levels

We evaluate three sparsity ratios representing conservative, moderate, and aggressive pruning:

- **30% Sparsity**: Conservative pruning, minimal accuracy loss expected
- **50% Sparsity**: Moderate pruning, half of parameters removed
- **70% Sparsity**: Aggressive pruning, stress-testing model resilience

4.4 Hardware and Software

Table 2: Experimental environment

Component	Specification
Platform	Lightning.ai
GPU	NVIDIA L40S (48GB VRAM)
CUDA Version	12.6
PyTorch Version	2.8.0
Transformers	4.57.3
LM Evaluation Harness	0.4.9
Precision	FP16
Random Seed	0

5 Results

5.1 Perplexity Evaluation

Table 3 presents the perplexity results on WikiText-2 for both models across all sparsity levels.

Table 3: Perplexity on WikiText-2 (lower is better). Wanda improvement shows the relative reduction in perplexity compared to magnitude pruning.

Model	Method	Perplexity	Δ from Dense	Wanda Improvement
LLaMA-2-7B	Dense (Baseline)	5.11	–	–
	Magnitude 30%	5.74	+12.1%	3.7%
	Wanda 30%	5.52	+8.0%	
	Magnitude 50%	11.83	+131.4%	**45.8%**
	Wanda 50%	6.42	+25.5%	
	Magnitude 70%	∞^*	–	–
	Wanda 70%	68.80	+1245.4%	
LLaMA-3.1-8B	Dense (Baseline)	5.85	–	–
	Magnitude 30%	8.50	+45.4%	25.7%
	Wanda 30%	6.32	+8.1%	
	Magnitude 50%	37.82	+546.8%	**76.2%**
	Wanda 50%	9.00	+53.8%	
	Magnitude 70%	263,197.6	–	99.96%
	Wanda 70%	96.75	+1554.3%	

* NaN value due to numerical overflow. The model produces undefined outputs at this sparsity level because too many important weights were removed.

1. **Wanda beats magnitude pruning** at all sparsity levels for perplexity (though at 30% sparsity for LLaMA-2, zero-shot accuracy is nearly identical).

2. **The advantage grows with sparsity**: At 30% sparsity, improvements are modest (3.7% to 25.7%), but at 50% sparsity, Wanda delivers large improvements (45.8% to 76.2%).

3. **LLaMA-3.1-8B breaks down faster with magnitude pruning**: At 50% sparsity, magnitude pruning increases perplexity by 546.8% for LLaMA-3 versus 131.4% for LLaMA-2. This suggests LLaMA-3 uses its parameters more efficiently, so removing them without considering activations hurts more.

4. **70% sparsity is too aggressive** for both methods. However, Wanda still produces valid outputs while magnitude pruning on LLaMA-2 produces NaN (numerical overflow).

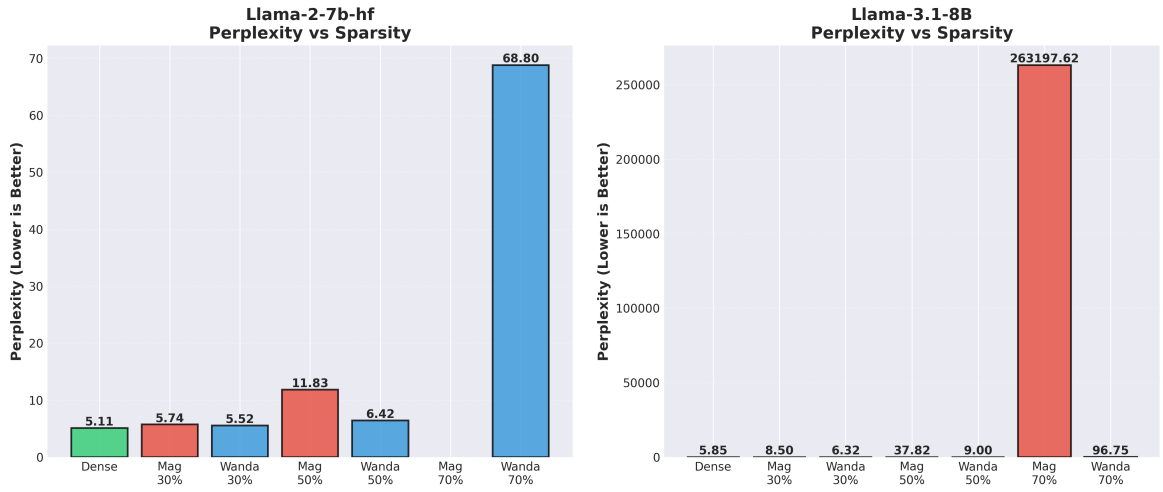

Figure 1: Perplexity comparison across methods and sparsity levels for LLaMA-2-7B (left) and LLaMA-3.1-8B (right). Note: The LLaMA-3 plot shows an extreme bar for magnitude pruning at 70% sparsity (perplexity = 263,197). This value is so large that it dwarfs all other bars, making them appear flat in comparison.

Note on extreme perplexity values: At 70% sparsity, magnitude pruning causes catastrophic model failure on both models. For LLaMA-2-7B, the perplexity returned NaN (undefined) due to numerical overflow. For LLaMA-3.1-8B, the perplexity reached 263,197, which means the model is essentially outputting random tokens. A perplexity this high indicates the model has lost nearly all language understanding. For context, a random baseline on WikiText-2 would have a perplexity around 50,000 (roughly the vocabulary size). The fact that magnitude pruning produces worse-than-random results shows that removing 70% of weights based only on magnitude destroys critical computational pathways. In contrast, Wanda at 70% sparsity still produces perplexity values of 68.8 (LLaMA-2) and 96.7 (LLaMA-3), which are poor but still represent coherent language modeling.

5.2 Zero-Shot Accuracy

Table 4 presents the average zero-shot accuracy across five benchmark tasks.

Table 4: Average zero-shot accuracy across five tasks (higher is better).

Model	Method	Accuracy (%)	Retention	Wanda Δ
LLaMA-2-7B	Dense (Baseline)	70.78	100.0%	–
	Magnitude 30%	69.09	97.6%	−0.01 pp
	Wanda 30%	69.08	97.6%	
	Magnitude 50%	62.26	88.0%	**+3.95 pp**
	Wanda 50%	66.21	93.5%	
	Magnitude 70%	38.82	54.9%	+3.46 pp
	Wanda 70%	42.28	59.7%	
LLaMA-3.1-8B	Dense (Baseline)	75.27	100.0%	–
	Magnitude 30%	71.74	95.3%	**+2.76 pp**
	Wanda 30%	74.49	99.0%	
	Magnitude 50%	56.43	75.0%	**+9.63 pp**
	Wanda 50%	66.05	87.8%	
	Magnitude 70%	40.28	53.5%	+2.14 pp
	Wanda 70%	42.42	56.4%	

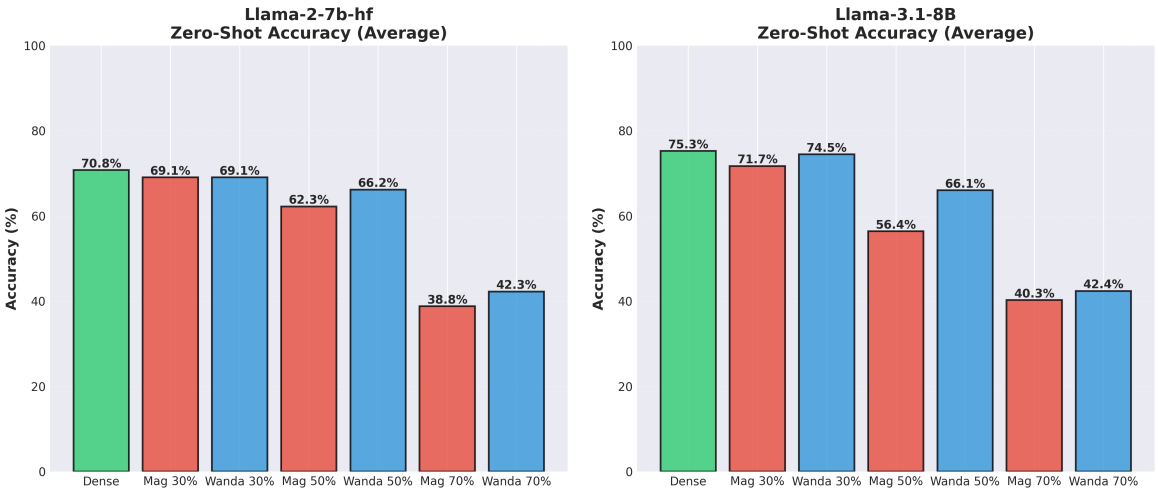

Figure 2: Zero-shot accuracy summary for LLaMA-2-7B (left) and LLaMA-3.1-8B (right).

5.3 Per-Task Breakdown

Table 5 provides detailed per-task accuracy at 50% sparsity, where Wanda's advantage is most pronounced.

Table 5: Per-task zero-shot accuracy at 50% sparsity (%).

Model	Method	PIQA	HellaSwag	ARC-Easy	BoolQ	RTE
LLaMA-2-7B	Dense	78.13	57.10	75.46	79.33	63.90
	Magnitude	74.59	53.09	67.85	63.43	52.35
	Wanda	76.06	51.82	72.56	76.09	54.51
LLaMA-3.1-8B	Dense	79.27	60.67	82.20	83.09	71.12
	Magnitude	70.46	43.27	62.21	52.42	53.79
	Wanda	73.88	50.53	71.84	79.14	54.87

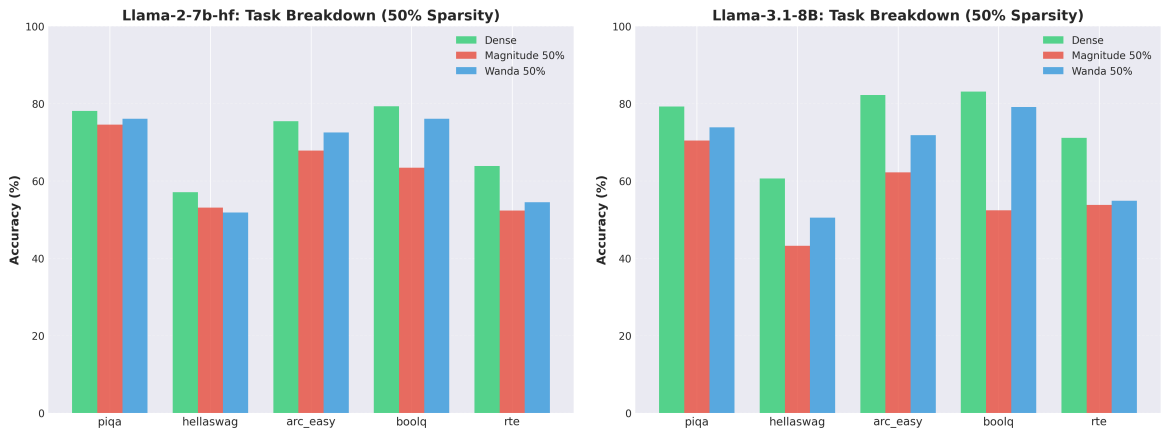

Figure 3: Per-task breakdown at 50% sparsity for LLaMA-2-7B (left) and LLaMA-3.1-8B (right).

5.3.1 Task-Specific Observations

- **BoolQ shows the largest improvement**: Wanda preserves 96% of dense accuracy versus 80% for magnitude pruning on LLaMA-2-7B at 50% sparsity (+12.66 percentage points).
- **HellaSwag is consistently challenging**: Both methods show significant drops, suggesting commonsense reasoning relies on distributed knowledge that is sensitive to pruning.
- **PIQA (physical reasoning) is most robust**: Both methods retain >90% of baseline at 50% sparsity.

5.4 Performance Degradation Analysis

Figure 4 visualizes the relative performance degradation as sparsity increases.

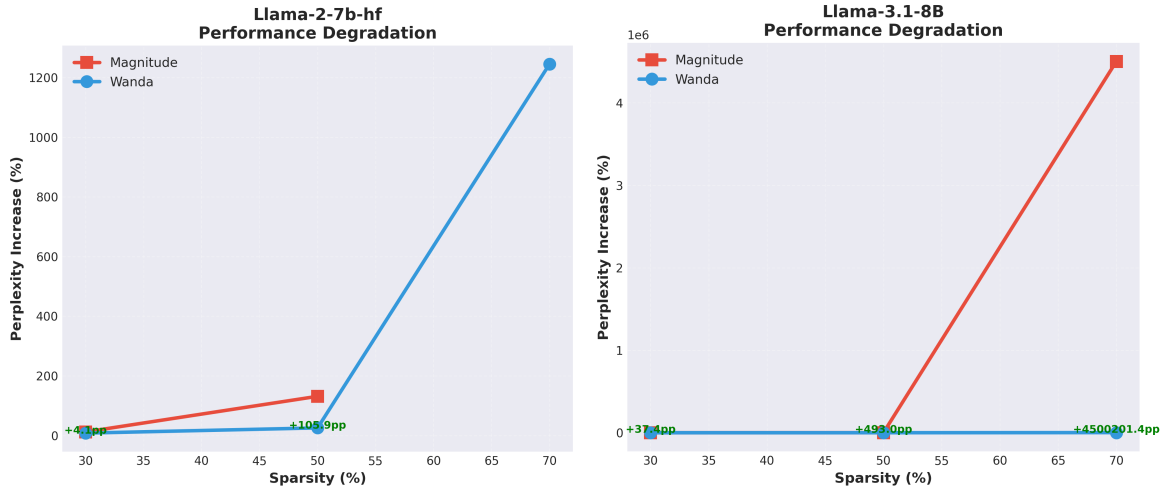

Figure 4: Perplexity increase relative to dense baseline for LLaMA-2-7B (left) and LLaMA-3.1-8B (right). Wanda consistently shows lower degradation.

5.5 Accuracy Retention

Figure 5 shows the percentage of original zero-shot accuracy retained at each sparsity level.

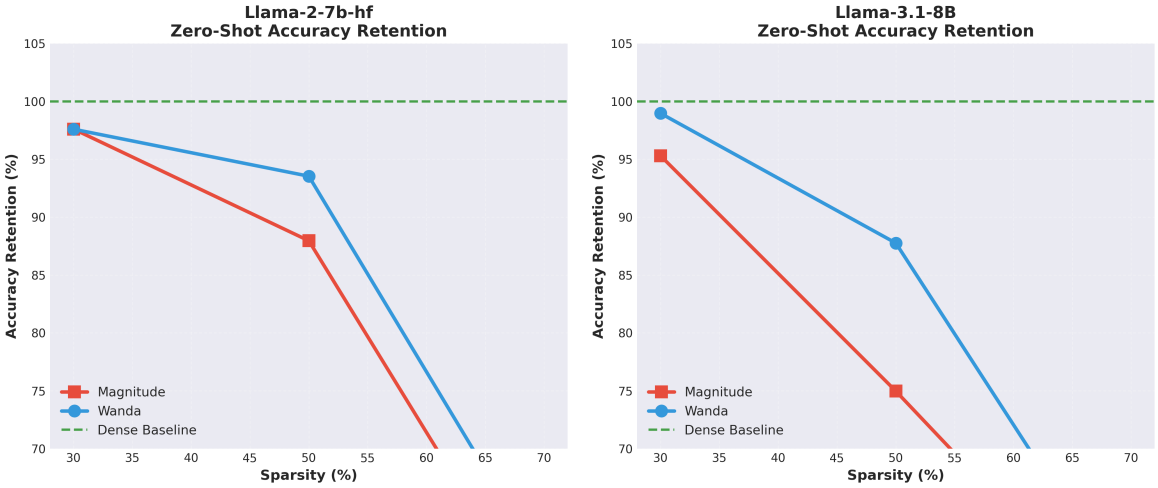

Figure 5: Zero-shot accuracy retention relative to dense baseline.

5.6 Computational Timing

Table 6 presents the execution time for key operations.

Table 6: Execution time in seconds for LLaMA-2-7B.

Operation	Magnitude	Wanda
Model Loading	53.7s	
Pruning (30%)	29.2s	59.3s
Pruning (50%)	28.4s	58.6s
Pruning (70%)	27.4s	57.3s
Perplexity Evaluation	∼32s	
Zero-Shot Evaluation	∼530s	

Wanda pruning takes approximately 2× longer than magnitude pruning due to the additional forward passes required to compute activation statistics. However, this is a one-time cost that yields significant accuracy improvements.

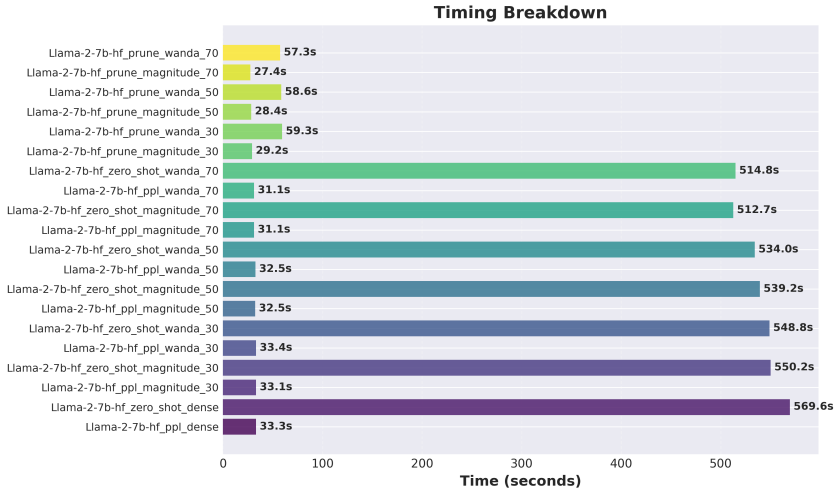

Figure 6: Timing breakdown for LLaMA-2-7B experiments.

6 Discussion

6.1 Why Wanda Outperforms Magnitude Pruning

The fundamental insight of Wanda is that **weight importance is context-dependent**. Consider two weights:

- **Weight A**: Large magnitude ($|w| = 0.5$), but connected to a rarely-activated input channel.
- **Weight B**: Medium magnitude ($|w| = 0.3$), but connected to a frequently-activated input channel.

Magnitude pruning would preserve Weight A and remove Weight B. However, Weight B contributes more to the model's actual computations on real data. Wanda corrects this by computing:

$$S_{\text{Wanda}}(A) = 0.5 \times \|X_{\text{low}}\|_2 < S_{\text{Wanda}}(B) = 0.3 \times \|X_{\text{high}}\|_2 \tag{6}$$

This explains why Wanda's advantage increases at higher sparsity levels: as more weights are removed, the cost of incorrect pruning decisions compounds, and magnitude pruning makes systematically worse decisions.

6.2 Analysis of the NaN Result

At 70% sparsity with magnitude pruning on LLaMA-2-7B, the perplexity evaluation returned NaN (Not a Number). This occurs because:

1. **Numerical overflow**: When 70% of weights are removed based solely on magnitude, the model's output logits can become extremely large or small.
2. **Loss of critical pathways**: Magnitude pruning may remove weights that are quantitatively small but qualitatively essential for maintaining stable forward propagation.
3. **Accumulated errors**: Through 32 transformer layers, small numerical instabilities compound into undefined values.

This result actually **highlights Wanda's advantage**: by considering activation patterns, Wanda preserves the weights that maintain numerical stability, producing coherent (if degraded) outputs even at 70% sparsity where magnitude pruning completely fails.

6.3 The 30% Sparsity Anomaly

At 30% sparsity on LLaMA-2-7B, Wanda and magnitude pruning achieve nearly identical zero-shot accuracy (69.08% vs 69.09%), with magnitude marginally better. This can be explained by:

1. **Low sparsity redundancy**: At 30% sparsity, most important weights are preserved by both methods. The "incorrectly" pruned weights by magnitude pruning have minimal impact.
2. **Statistical noise**: A difference of 0.01 percentage points is within measurement noise given the finite size of evaluation datasets.
3. **Calibration distribution**: Wanda's advantage depends on calibration data (WikiText-2) accurately representing the activation patterns needed for downstream tasks.

6.4 Model Architecture Effects

LLaMA-3.1-8B shows greater sensitivity to magnitude pruning compared to LLaMA-2-7B:

- At 30% sparsity: LLaMA-3 perplexity increases 45.4% vs 12.1% for LLaMA-2
- At 50% sparsity: LLaMA-3 perplexity increases 546.8% vs 131.4% for LLaMA-2

This suggests that LLaMA-3's improved performance comes from more efficient parameter utilization, leaving less "redundant" capacity that can be safely pruned without activation awareness.

6.5 Failure Modes and Limitations

6.5.1 High Sparsity Limits

Both methods show severe degradation beyond 70% sparsity. This is a hard limit: removing 70% of parameters leaves the model unable to do complex reasoning, no matter which weights you keep.

6.5.2 Task-Specific Degradation

Certain tasks (e.g., RTE, HellaSwag) show greater sensitivity to pruning than others (e.g., PIQA). Tasks requiring:

- Complex multi-hop reasoning
- Fine-grained semantic distinctions
- Long-range dependencies

are more affected by parameter reduction.

6.5.3 Calibration Data Dependency

Wanda's effectiveness depends on calibration data representing the target distribution. If calibration data differs significantly from deployment scenarios (e.g., calibrating on Wikipedia text but deploying for code generation), performance gains may diminish.

6.5.4 Unstructured Sparsity Hardware Limitations

While unstructured pruning achieves higher accuracy at equivalent sparsity, it does not directly translate to inference speedups on standard hardware. Sparse matrix operations require:

- Specialized CUDA kernels (e.g., cuSPARSE)
- Hardware support (e.g., NVIDIA Ampere's structured sparsity)
- Quantization-aware sparse formats

For practical deployment, structured pruning (e.g., 2:4 sparsity) may be preferable despite slightly lower accuracy.

7 Conclusion

This project successfully reproduced the Wanda pruning method and validated its effectiveness against magnitude-based pruning on two state-of-the-art LLMs. Our key findings confirm the original paper's claims:

1. **Wanda consistently outperforms magnitude pruning**, with improvements ranging from 3.7% to 76.2% in perplexity depending on model and sparsity level.
2. **The advantage is most pronounced at moderate-to-high sparsity** (50%), where Wanda delivers 45.8% lower perplexity on LLaMA-2-7B and 76.2% lower perplexity on LLaMA-3.1-8B.
3. **Zero-shot accuracy improvements follow similar patterns**, with Wanda providing up to 9.63 percentage points higher accuracy at 50% sparsity.
4. **At extreme sparsity (70%)**, Wanda maintains coherent outputs while magnitude pruning produces numerical failure, demonstrating superior preservation of model stability.

The additional computational cost of Wanda (approximately 2× pruning time) is justified by the significant quality improvements, especially for deployment scenarios where model size is constrained.

7.1 Future Work

- Evaluate structured sparsity variants (2:4, 4:8) for hardware-accelerated inference
- Combine Wanda with quantization for compound compression
- Investigate task-specific calibration to improve downstream performance
- Apply brief post-pruning fine-tuning to recover additional accuracy

References

Sun, M., Liu, Z., Bair, A., & Kolter, J. Z. (2024). A Simple and Effective Pruning Approach for Large Language Models. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., ... & Lample, G. (2023). LLaMA: Open and Efficient Foundation Language Models. *arXiv preprint arXiv:2302.13971*.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., ... & Scialom, T. (2023). Llama 2: Open Foundation and Fine-Tuned Chat Models. *arXiv preprint arXiv:2307.09288*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is All You Need. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Frantar, E., & Alistarh, D. (2023). SparseGPT: Massive Language Models Can Be Accurately Pruned in One-Shot. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Frantar, E., Ashkboos, S., Hoefler, T., & Alistarh, D. (2022). GPTQ: Accurate Post-Training Quantization for Generative Pre-trained Transformers. *arXiv preprint arXiv:2210.17323*.

Ma, X., Fang, G., & Wang, X. (2023). LLM-Pruner: On the Structural Pruning of Large Language Models. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Gao, L., Tow, J., Abbasi, B., Biderman, S., Black, S., DiPofi, A., ... & Zou, A. (2023). A Framework for Few-shot Language Model Evaluation. *Zenodo*. https://doi.org/10.5281/zenodo.10256836

Bisk, Y., Zellers, R., Bras, R. L., Gao, J., & Choi, Y. (2020). PIQA: Reasoning about Physical Commonsense in Natural Language. In *Proceedings of AAAI*.

Zellers, R., Holtzman, A., Bisk, Y., Farhadi, A., & Choi, Y. (2019). HellaSwag: Can a Machine Really Finish Your Sentence? In *Proceedings of ACL*.

Clark, P., Cowhey, I., Etzioni, O., Khot, T., Sabharwal, A., Schoenick, C., & Tafjord, O. (2018). Think You Have Solved Question Answering? Try ARC, the AI2 Reasoning Challenge. *arXiv preprint arXiv:1803.05457*.

Clark, C., Lee, K., Chang, M.-W., Kwiatkowski, T., Collins, M., & Toutanova, K. (2019). BoolQ: Exploring the Surprising Difficulty of Natural Yes/No Questions. In *Proceedings of NAACL*.

Dagan, I., Glickman, O., & Magnini, B. (2005). The PASCAL Recognising Textual Entailment Challenge. In *Machine Learning Challenges Workshop*.

Meta AI. (2024). Llama 3.1 Model Card. https://github.com/meta-llama/llama-models

Merity, S., Xiong, C., Bradbury, J., & Socher, R. (2016). Pointer Sentinel Mixture Models. *arXiv preprint arXiv:1609.07843*.

A Additional Visualizations

A.1 Wanda vs. Magnitude Heatmap

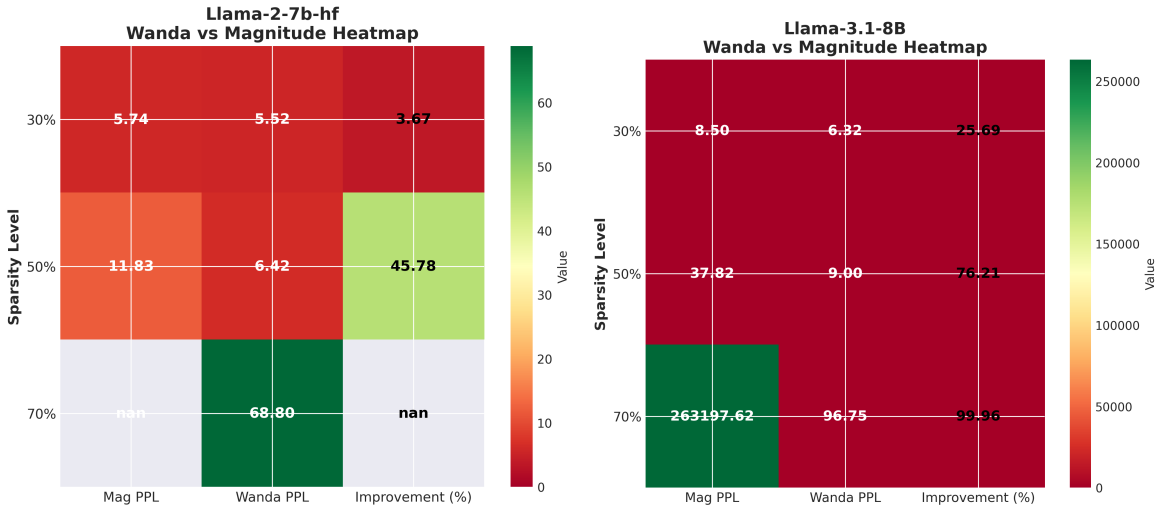

Figure 7: Perplexity comparison heatmap showing Wanda's improvement at each sparsity level.

A.2 Per-Task Breakdown at 30% and 70% Sparsity

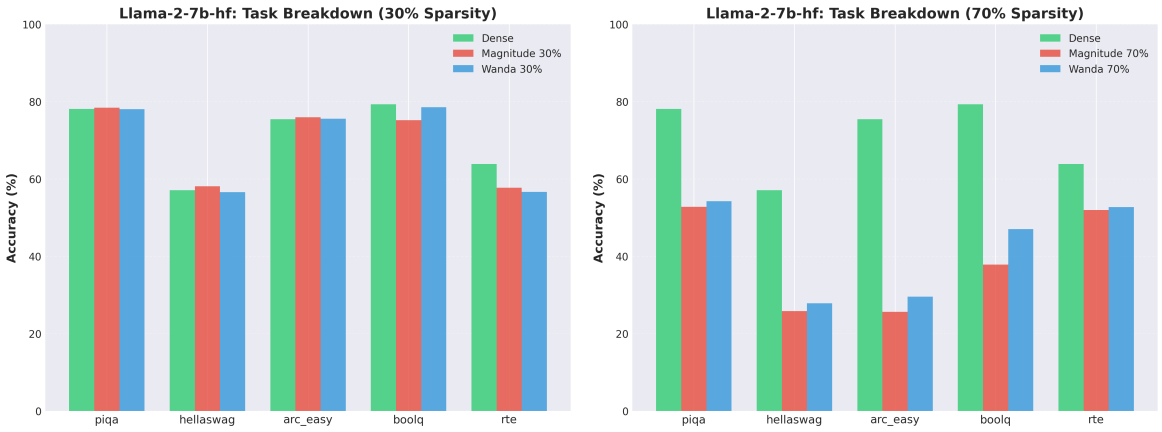

Figure 8: Per-task breakdown for LLaMA-2-7B at 30% (left) and 70% (right) sparsity.

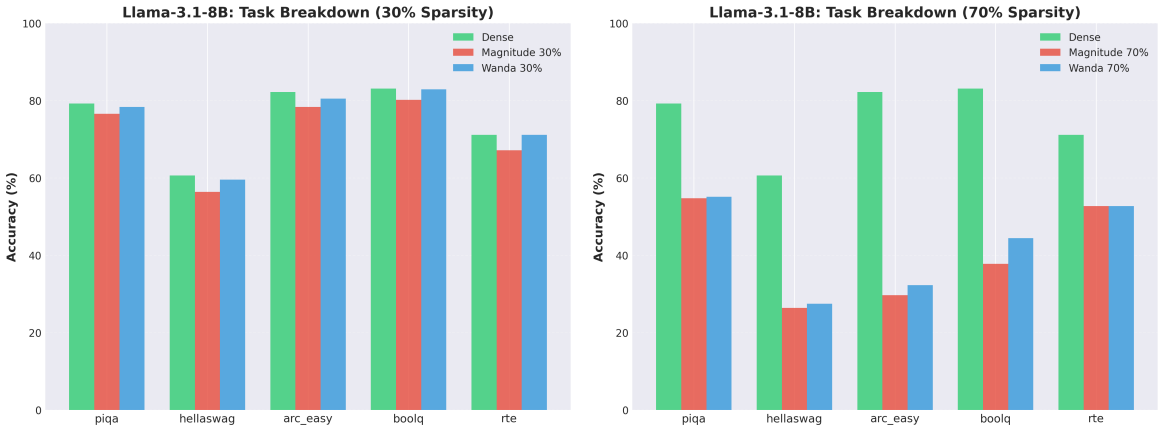

Figure 9: Per-task breakdown for LLaMA-3.1-8B at 30% (left) and 70% (right) sparsity.

B Complete Results Tables

B.1 LLaMA-2-7B Zero-Shot Results

Table 7: Complete zero-shot accuracy for LLaMA-2-7B (%)

Method	PIQA	HellaSwag	ARC-Easy	BoolQ	RTE	Average
Dense	78.13	57.10	75.46	79.33	63.90	70.78
Magnitude 30%	78.40	58.13	75.97	75.20	57.76	69.09
Wanda 30%	78.02	56.56	75.59	78.56	56.68	69.08
Magnitude 50%	74.59	53.09	67.85	63.43	52.35	62.26
Wanda 50%	76.06	51.82	72.56	76.09	54.51	66.21
Magnitude 70%	52.77	25.87	25.63	37.86	51.99	38.82
Wanda 70%	54.24	27.89	29.55	47.03	52.71	42.28

B.2 LLaMA-3.1-8B Zero-Shot Results

Table 8: Complete zero-shot accuracy for LLaMA-3.1-8B (%)

Method	PIQA	HellaSwag	ARC-Easy	BoolQ	RTE	Average
Dense	79.27	60.67	82.20	83.09	71.12	75.27
Magnitude 30%	76.61	56.37	78.37	80.18	67.15	71.74
Wanda 30%	78.35	59.55	80.51	82.94	71.12	74.49
Magnitude 50%	70.46	43.27	62.21	52.42	53.79	56.43
Wanda 50%	73.88	50.53	71.84	79.14	54.87	66.05
Magnitude 70%	54.73	26.44	29.71	37.83	52.71	40.28
Wanda 70%	55.17	27.48	32.32	44.43	52.71	42.42