

Improving Efficiency of Ternary Large Language Models through Pruning

Pietro Tropeano

University of Copenhagen
Copenhagen, Denmark
pitr@di.ku.dk

Maria Maistro

University of Copenhagen
Copenhagen, Denmark
mm@di.ku.dk

Tuukka Ruotsalo

University of Copenhagen
Copenhagen, Denmark
LUT University / Lahti, Finland
tr@di.ku.dk

Christina Lioma

University of Copenhagen
Copenhagen, Denmark
c.lioma@di.ku.dk

Nikolaos Aletras

University of Sheffield
Sheffield, United Kingdom
n.aletras@sheffield.ac.uk

Abstract

As the environmental impact of Large Language Models (LLMs) grows, improving model efficiency has become critical for sustainable NLP. Recent advances in model compression have shown that ternary quantisation (i.e., using only $\{-1, 0, 1\}$ as weights) substantially reduces memory footprint, inference time, and energy usage of LLMs with minimal performance degradation. However, whether these already efficient models can be further pruned to reduce resource requirements without compromising effectiveness remains unexplored. We investigate this question across six models, four sparsity levels, and six structured and unstructured pruning techniques. We find that even minimal pruning yields substantial efficiency gains: pruning fewer than 5% of parameters reduces inference time and energy consumption by up to 21%, and memory usage by up to 13%, with minimal accuracy degradation. We also find that, for pruned models with less than a 5% accuracy drop, the most energy and inference-time efficient configurations are achieved by pruning entire attention layers, whereas the most memory efficient configurations are obtained by pruning entire MLP layers. These findings highlight pruning as a complementary technique to ternary quantisation for reducing environmental footprint of LLMs while maintaining model effectiveness. The code is available at: <https://github.com/pietrotrope/TernaryLLMsPruning>.

1 Introduction

Large Language Models (LLMs) offer impressive performance across a wide range of tasks (team, 2026; Liu et al., 2025; Yang et al., 2025a; Mistral-

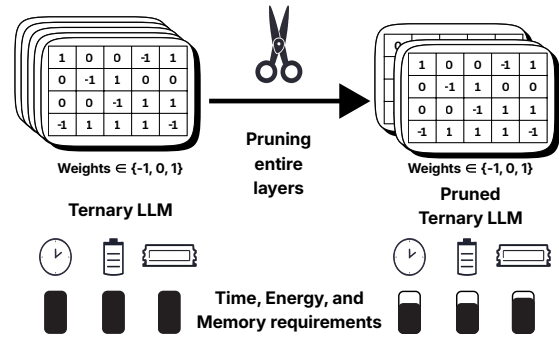


Figure 1: Efficiency gains of pruning ternary models. Pruning a small amount of entire LLMs’ MLP or attention layers reduces time, energy, and memory usage without substantially compromising their accuracy.

AI, 2025). However, they require significant computational resources when deployed. To mitigate this, model compression techniques such as quantisation and pruning are widely used (Dettmers et al., 2024; Frantar and Alistarh, 2023). Quantisation reduces the number of bits used to represent model parameters (Lang et al., 2024) and pruning removes parts of the model to reduce its size (Zhu et al., 2024). Both methods can be applied during or after training (Dettmers et al., 2024; Ma et al., 2024; Frantar and Alistarh, 2023; Kurtic et al., 2022), yet, when pruning LLMs, post-training pruning is the most common solution due to high training costs.

Ternary quantisation such as the BitNet family of ternary models (Ma et al., 2024; Wang et al., 2023), uses only three values (i.e., $\{-1, 0, 1\}$) to represent weights. This extreme quantisation technique achieves accuracy comparable with similarly sized full precision models while requiring only

1.58 bits per weight. Ternary models are designed for deployment in edge devices and low-resource settings (Ma et al., 2024), enabling broader accessibility and more environmentally sustainable use compared to full-precision models.

Although ternary quantisation reduces inference time and memory footprint, it does not reduce the number of model parameters. Pruning can complement quantisation by removing parameters directly, potentially further reducing memory usage, inference time, and energy consumption. Thus, pruning is relevant for sustainable NLP, as improving LLMs efficiency will reduce their environmental impact.

To this end, we contribute the first comprehensive analysis of post-training pruning for ternary LLMs. We study the extent to which ternary LLMs can be pruned without substantially compromising their effectiveness, and we characterise the resulting trade-offs between effectiveness and efficiency. Our study evaluates six pruning algorithms applied to six ternary models across four sparsity levels, measuring both downstream performance on seven datasets and the models’ reduction in inference time, energy, and memory requirements.

We find that (i) layer pruning is an effective solution to prune ternary LLMs, allowing to remove a small number of parameters without substantially compromising accuracy. (ii) Even modest pruning, especially when removing entire layers, improves ternary LLMs’ efficiency (Fig. 1). (iii) Layer pruning results in the best accuracy-efficiency trade-offs. Among pruned LLMs with limited accuracy degradation, pruning entire attention layers produces the best improvements in inference time and energy efficiency, whereas pruning entire MLP layers yields the greatest memory savings. Overall our findings highlight new opportunities to improve efficiency of low-resource LLMs.

2 Related Work

Model quantisation is a common compression approach that reduces the number of bits used to represent parameters (Lang et al., 2024), leading to practical memory and inference time efficiency improvements. While common quantisation methods reduce bits to 8, 4, or 2 (Xiao et al., 2023; Frantar et al., 2023; Chee et al., 2023), recent work, such as BitNet (Ma et al., 2024), has explored 1.58-bit models, also known as ternary models. These models allow only three possible states ($\{-1, 0, 1\}$), with accuracy that matches similarly sized full-precision

models. Ternary models are natively sparse because one of their possible weight states is 0; the value conventionally used to prune weights.

Pruning is another compression technique that removes parameters or entire network structures according to a heuristic. Pruning algorithms can be unstructured or structured. Unstructured pruning removes individual parameters. Examples of such algorithms are Wanda (Sun et al., 2024), Ria (Zhang et al., 2024), and SparseGPT (Frantar and Alistarh, 2023). Structured pruning removes entire structures such as neurons or layers (Michel et al., 2019; Ma et al., 2023; Ashkboos et al., 2024). Unstructured pruning generally yields better accuracy-sparsity trade-offs (Hu et al., 2025a). Yet it does not offer inference speedups unless paired with custom hardware and sparsity patterns (Yang et al., 2025b). Structured pruning, on the other hand, provides tangible efficiency improvements by removing entire structures (Hu et al., 2025a). A hybrid approach, N:M semi-structured pruning, enables hardware acceleration by enforcing at most N non-zero parameters for every M contiguous parameters (Zhou et al., 2021). However, as current hardware accelerates only 2:4 sparse matrices, this limits pruning to remove exactly 50% of the weights and restricts more fine-grained sparsity exploration.

Pruning and quantisation are often combined, typically jointly, or by pruning first, then quantising the model (Han et al., 2016; van Baalen et al., 2020; Zhang et al., 2026). Zhang et al. (2026) applies semi-structured pruning during training of ternary LLMs. Yet, to the best of our knowledge, post-training pruning of ternary LLMs is unexplored, and addressing this gap is the focus of our work.

3 Study Design

To systematically analyse the interplay between extreme quantisation and sparsity, we design a comprehensive evaluation framework that subjects ternary LLMs to varying degrees of pruning. Our aim is to study how pruning ternary LLMs affects both *effectiveness* and *efficiency*.

3.1 Ternary Models

We use six ternary LLMs, covering a wide range of sizes (from 1B, up to 8B), and all four LLM families using the BitNet architecture (Wang et al., 2023). Specifically, we use BitNet 2B (Ma et al., 2024), Llama 3 8B 1.58b (Mekroui et al., 2024), the 1B and 3B Falcon Edge models (Falcon-LLM

Team, 2025), and the 3B and 7B versions of Falcon 3 1.58b (Falcon-LLM Team, 2024).

3.2 Pruning Methods

We compress ternary LLMs using six widely-used algorithms covering three families of pruning approaches: (1) Unstructured pruning, which removes individual parameters; (2) Neuron pruning, which eliminates entire neurons (corresponding to rows in the weight matrices); and (3) Layer pruning, which removes entire layers from the network. We chose these algorithms to prune ternary models at different granularities, ranging from individual parameters to neurons to entire layers.

Unstructured Pruning. We prune individual parameters using two of the best-performing unstructured pruning algorithms for LLMs: Wanda (Sun et al., 2024) and Ria (Zhang et al., 2024). We exclude unstructured pruning methods that rely on weight reconstruction, i.e., algorithms that modify the remaining weights after pruning to recover accuracy, such as SparseGPT (Frantar and Alistarh, 2023), because these updated weights would violate the ternary constraints. Wanda determines which parameters to prune based on the absolute magnitude of weights (0 or 1 in ternary models) and activations. Ria extends Wanda by incorporating both row-wise and column-wise normalised weight magnitudes, averaging these two scores and using activations to guide pruning. We use the standard configurations for both algorithms. As Sun et al. (2024) and Zhang et al. (2024), we compute activations with 128 samples of 2048 tokens from the C4 dataset (Raffel et al., 2020).

Neuron Pruning. To prune neurons, we use the algorithm from Hu et al. (2025a) on neurons, which is a straightforward approach to remove neurons without retraining or violating ternary constraints. The algorithm computes each neuron’s score by summing its parameter scores (using Wanda’s scoring function) and removes neurons with the lowest scores. When a neuron is removed from an MLP block, the corresponding weights in subsequent linear layers are also pruned, eliminating the features produced by that neuron. For attention layers, we apply an analogous strategy by removing features that are consistently inactive, i.e., those that remain zero across 128 sequences of 2048 tokens. We use two variants for this method: one to prune attention layers and one for MLP. Since we work with ternary LLMs, we do not update weights after pruning, as in Hu et al. (2025a). Weights updates would

Dataset	Task	# samples	# Classes
ARC Easy	Multiple Choice QA	2376	4
ARC Challenge	Multiple Choice QA	1172	4
OpenBookQA	Multiple Choice QA	500	4
BoolQ	Question Answering (Yes/No)	3270	2
RTE	Natural Language Inference	277	2
Hellaswag	Natural Language Inference	10000	4
Winogrande	Commonsense reasoning	1267	2
Wikitext-2	Language Modelling	4358	-

Table 1: Datasets’ task, number of samples, and classes.

result in non-ternary weights.

Layer Pruning. For layer pruning, we use the lightweight method proposed by He et al. (2026). We select this approach because, unlike other methods, it does not yield domain-specific models (Hu et al., 2025b), and does not require further training (Lu et al., 2024). This pruning algorithm iteratively removes layers from an LLM based on their relative importance. Layer importance is estimated as the cosine similarity between its input and output representations (Men et al., 2025); where lower similarity implies higher importance. This relies on a small calibration set to obtain layer inputs and outputs. Since He et al. (2026) demonstrate that their method is robust to the quantity of calibration data, we use eight sequences of 2048 tokens sampled from the C4 dataset. Similar to Neuron Pruning, we use one variant that removes attention layers and another that removes MLP layers.

3.3 Model Sparsity Setup

To fairly compare different pruners, all pruned models must achieve roughly the same final sparsity. Otherwise, applying the same pruning threshold, a hyperparameter that controls how many parameters are removed, can remove very different numbers of parameters across methods, making efficiency comparisons unfair. This is not straightforward, as each method uses a pruning threshold, but its interpretation differs across strategies. In unstructured pruning, the threshold identifies the percentage of parameters to remove per layer, regardless of whether some of these parameters are already zero (i.e., already pruned). In neuron pruning, it indicates the fraction of neurons to remove, ignoring the sparsity of the remaining neurons. In layer pruning, it defines the number of layers to remove, independent of their internal sparsity.

As a result, the pruning threshold does not directly determine final sparsity. For instance, in unstructured pruning (e.g., Wanda), neurons that are already sparser than the pruning threshold are left unchanged. When combined with other neu-

rons that are pruned, the average sparsity of the layer will exceed the pruning threshold. Practically, if one neuron is already 45% sparse and the pruning threshold is 30%, the neuron is not pruned, and its higher sparsity adds to the overall layer, pushing the final average sparsity above 30%.

Prior work typically evaluates pruning methods within a single pruning family (e.g., structured or unstructured) on dense models (He et al., 2026; Sun et al., 2024; Frantar and Alistarh, 2023), making sparsity standardisation unnecessary. In contrast, we study already sparse models using different pruning families; therefore, we need to consider the target sparsity. To address this, we first select thresholds for attention-layer pruning. For other methods, we perform a binary search to find thresholds that achieve the closest final sparsity, stopping once the target is reached within 1% tolerance. This ensures that all pruned models achieve roughly the same sparsity. We present the final sparsities and pruning thresholds in the Appendix (see Tabs. 3-8).

3.4 Effectiveness Evaluation

We use seven question-answering and natural language inference datasets: ARC Easy and ARC Challenge (Clark et al., 2018), OpenBookQA (Mihaylov et al., 2018), BoolQ (Clark et al., 2019), RTE (Wang et al., 2018), Hellaswag (Zellers et al., 2019), and Winogrande (Sakaguchi et al., 2021). We measure 0-shot accuracy on the test set of all datasets, using LM Evaluation Harness (Gao et al., 2024) standard prompt templates. We also evaluate language modelling effectiveness by computing perplexity on the Wikitext-2 dataset (Merity et al., 2017). Tab. 1 presents an overview of the datasets. Except for layer pruning, which removes layers in a fixed order and produces identical models, all results are averaged over three random seeds.

3.5 Efficiency Evaluation

We evaluate efficiency by measuring memory, GPU time, and energy required by the models. Following (Ashkboos et al., 2024; Sun et al., 2024; Frantar and Alistarh, 2023), we use matrix multiplication time, which requires the most computation in LLMs, as a proxy for inference time. We extend this approach by tracking memory and energy consumption, and by measuring all layers individually rather than using a single representative matrix per layer type.

We compute efficiency as follows. For layer pruning, we exclude the computation time of removed layers and aggregate the inference time of

the remaining layers. For neuron pruning, we instantiate smaller layers with the pruned neurons removed, measure the resulting matrix multiplication costs, and sum the times across all layers. All evaluations use real activations sampled from 128 sequences of 2048 tokens from the C4 dataset.

We compute memory savings based on the layer’s total number of parameters (including zeros) and the number of bits used to represent them. GPU time and energy are tracked on full-precision matrices (i.e., 32 bits) using Carbontracker (Anthony et al., 2020), a framework that tracks resource usage of Deep Learning models.

We note that unstructured pruning approaches do not yield more efficient models directly, as they require dedicated hardware to leverage sparsity, which currently does not exist. Hence, we only track the efficiency of structured pruning algorithms. Furthermore, unlike Sun et al. (2024) and Frantar and Alistarh (2023), we exclude NVIDIA Cutlass acceleration for N:M sparsity (where a matrix admits at most N non-zero parameters for every contiguous M parameters). This is necessary as our pruned models are not N:M sparse, and enforcing this structure would constrain unstructured pruning to a 50% sparsity.

Accuracy-Efficiency trade-off. We study the accuracy-efficiency trade-off of our pruned LLMs using the measure Performance per Resource Unit (PePR) (Selvan et al., 2025). PePR is defined as:

$$PePR(R, P) = \frac{P}{1 + R} \quad (1)$$

where $R = (R - R_{min}) / (R_{max} - R_{min})$ is the min-max normalised resource cost in the experimental setting. In our experiments, we track memory and energy consumption and apply normalisation separately for each model, considering its various pruned versions. P is a normalised measure of model performance, in our case, accuracy. The PePR score ranges between 0 and 1. Lower scores indicate a worse trade-off between effectiveness and efficiency, while a score of 0 corresponds to a model with 0% accuracy. A score of 1 indicates a model that achieves 100% accuracy while being the most efficient in terms of the chosen resource.

4 Results

4.1 Effectiveness

Figs. 2 (a-g) show the accuracy of pruned and unpruned BitNet across all datasets (see Figs. 6-10 in Appendix for other models). Across datasets and

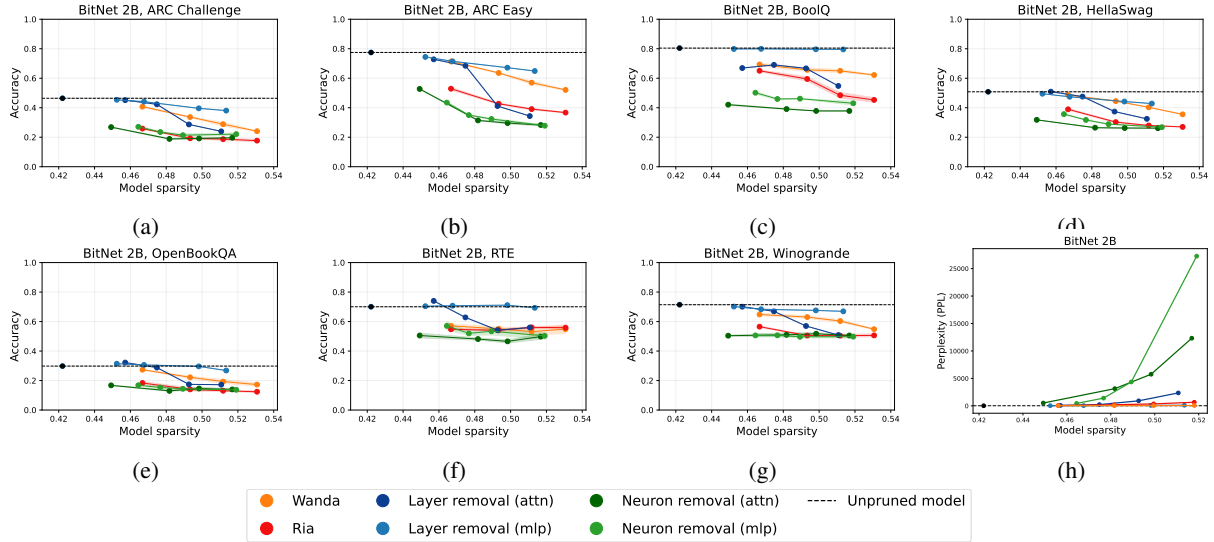


Figure 2: Panels (a)-(g): **Accuracy** of pruned and unpruned **BitNet** on seven datasets (y-axis), plotted against the **relative sparsity** of the pruned models compared to the original model (x-axis). Panel (h): **Perplexity** of pruned and unpruned BitNet on the Wikitext-2 dataset (y-axis) versus relative sparsity (x-axis). Unpruned models are shown with dashed lines. All shaded regions indicate variability across three runs ($\pm$ standard deviation).

models, pruning MLP layers consistently yields the highest accuracy given a model sparsity. In contrast, neuron removal on attention layers results in the lowest accuracy across sparsity levels.

Overall, MLP layer pruning can increase model sparsity up to 5% while keeping accuracy drops below 5%. The only exception is Falcon-E 3B, for which our experimental setup limits the sparsity increases to 4%. Nevertheless, the same trend holds when increasing sparsity up to 4%. These results demonstrate that *ternary models, despite being inherently sparse, can be further sparsified through pruning with high accuracy retention*.

Fig. 2 (h) shows models’ perplexity on Wikitext-2 (see Appendix, Figs. 6-10 for other models). Unstructured pruning (Wanda and Ria) achieve the lowest perplexity when pruning the most parameters, whereas neuron removal yields the highest perplexity. For both neuron and layer removal, pruning attention layers leads to higher perplexity than pruning MLP. This trend holds across models.

4.2 Efficiency

We next quantify the impact of pruning on ternary models efficiency by analysing their inference time, energy consumption, and memory requirements on 128 samples from the C4 dataset. We remove outliers that deviate by more than three standard deviations from the mean matrix multiplication time. These outliers account for less than 2.2% of the data and are sporadic measurements with abnormally

large runtimes, which would skew the estimates.

Fig. 3 shows the average inference time (first column), energy consumption (second column), and memory requirements (third column) of pruned BitNet and LLaMA models at a fixed target sparsity (as the trends are consistent across models, the remaining models are provided in Appendix, Figs. 11, 12, and 13). Across all models, pruning attention layers (dark blue lines) consistently yields the lowest inference time for a given target sparsity.¹ In contrast, pruning entire neurons (green lines) in the attention and MLP layers results in slower inference in most settings. We hypothesise that this behaviour arises because GPUs are highly optimised for matrix operations with specific dimensions; removing neurons may increase computation time rather than reducing it if the shape of the new matrix is not aligned with GPU-optimised cores (NVIDIA, 2023). Similarly, when it comes to energy consumption (second column), except for Falcon-3 7B, removing attention layers produces the most energy-efficient models across sparsity levels. Last, for a fixed target sparsity, all pruning algorithms achieve similar reductions in memory footprint, except for BitNet, where MLP neuron removal leads to greater memory savings than the other pruners. Overall, this means that in our settings, *pruning attention layers is the most effective*

¹This advantage would likely be greater in deployment, where pruning attention layers reduces KV-cache overhead; here we measure only matrix multiplication time.

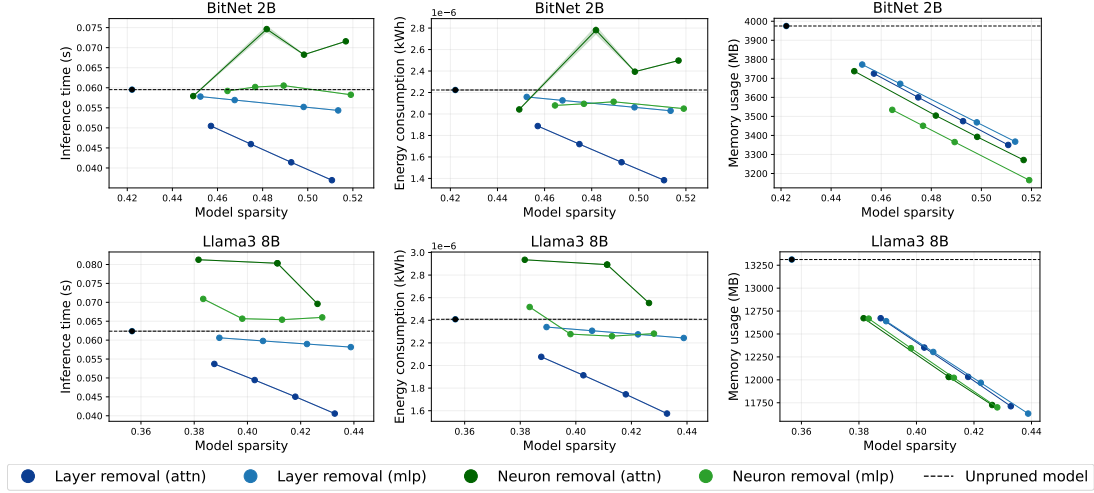


Figure 3: Efficiency of pruned models as a function of sparsity. The first, second, and third columns show inference time, energy, and memory, respectively, with sparsity on the x-axis. Coloured lines represent different pruning methods, while the black dotted line indicates the unpruned model. The black dot marks the sparsity of the unpruned model. For time and energy, all shaded regions indicate variability across 128 samples ($\pm$ standard deviation).

strategy for reducing time and energy requirements.

4.3 Effectiveness-efficiency Trade-off

Next, we evaluate accuracy-efficiency trade-off of pruned models using the PePR score (see Sec. 3.4).

Performance Per Resource Unit. Fig. 4 shows the PePR time (first column), PePR energy (second column), and PePR memory (third column) scores of different pruned versions of BitNet as a function of their final sparsity on two datasets. Similar trends are observed across all datasets and model architectures (see Appendix, Figs. 14-31).

When pruning a small fraction of parameters, removing entire attention layers (dark blue lines) yields the best performance per unit of inference time and energy. However, as sparsity increases, pruning entire MLP layers (light blue lines) often produces models that require less time and energy to achieve the same performance. Across all sparsity levels, pruning MLP layers is the strategy that most frequently maximises performance per unit of memory. In contrast, neuron-level pruning (green lines) significantly harms model accuracy, consistently resulting in worse PePR scores than the original unpruned model (black dot).

Across all models and datasets, we consistently find pruned configurations that outperform the original model in time or memory efficiency while achieving higher PePR scores, indicating better accuracy per unit of time, memory, or energy.

Overall, pruning either MLP or attention layers yields the best accuracy-efficiency trade-offs.

Accuracy against Efficiency. Performance per resource unit alone does not provide a complete picture of a model’s suitability for deployment, as it collapses accuracy and efficiency into a single scalar value. This aggregation does not show which model best satisfies specific deployment constraints, such as strict limits on memory usage or energy consumption. Given a concrete deployment bottleneck, a more informative approach is to directly examine model accuracy as a function of the constrained resource. Accordingly, we next study the accuracy-time, accuracy-energy, and accuracy-memory trade-offs of the pruned models.

Fig. 5 shows the accuracy of pruned BitNet models on two classification tasks against their mean inference time (first column), energy (second column), and memory requirements (third column). Similar trends hold across other datasets and model architectures (Appendix, Figs. 32-49).

Across all three efficiency aspects, in the best case, removing entire attention layers yields models that remain close in accuracy to the unpruned model (black point), with accuracy drops not exceeding 5% (except for BoolQ, where the drop reaches 11%). This pruning strategy increases sparsity by less than 4% while delivering substantial efficiency gains, including up to a 21% inference time and energy consumption reduction. Pruning MLP layers also preserves accuracy but results in more limited reductions in inference time and energy usage. These trends hold consistently across all models, except Falcon-3 3B, where removing at-

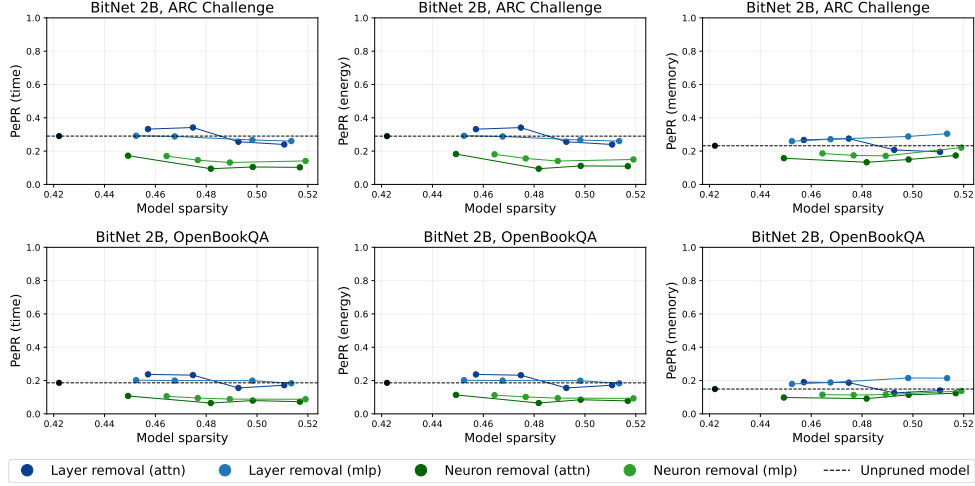


Figure 4: PePR score of pruned models as a function of sparsity. The first, second, and third columns show PePR scores for time, energy, and memory, respectively, with sparsity on the x-axis. Results on the first row are for ARC challenge, while results on the second row are for OpenbookQA. Coloured lines represent different pruners, while the black dotted line indicates the unpruned model. The black dot marks the sparsity of the original unpruned model.

Model	Pruning Algorithm	Sparsity	Accuracy	Accuracy Drop	Energy (1e5 Kwh)	Time (s)	Memory (MB)
BitNet 2B	Unpruned	0.422	0.609	-	0.222	0.060	3975
BitNet 2B	Layer removal (attn)	0.457	0.589	0.020	0.189	0.050	3725
BitNet 2B	Layer removal (mlp)	0.452	0.602	0.007	0.216	0.058	3773
BitNet 2B	Layer removal (mlp)	0.468	0.590	0.020	0.213	0.057	3671
BitNet 2B	Layer removal (mlp)	0.498	0.570	0.039	0.206	0.055	3469
Llama3 8B	Unpruned	0.357	0.543	-	0.241	0.062	13312
Llama3 8B	Layer removal (attn)	0.388	0.532	0.011	0.208	0.054	12672
Llama3 8B	Layer removal (attn)	0.403	0.510	0.033	0.191	0.049	12352
Llama3 8B	Layer removal (mlp)	0.389	0.532	0.010	0.234	0.061	12640
Llama3 8B	Layer removal (mlp)	0.406	0.515	0.028	0.231	0.060	12304
Llama3 8B	Layer removal (mlp)	0.422	0.506	0.036	0.228	0.059	11968

Table 2: Configurations of pruned BitNet 2B and Llama3 8B where the drop in accuracy does not exceed 5%.

tention layers always leads to substantial accuracy drops on ARC Easy and BoolQ.

In terms of memory efficiency, for accuracy drops of up to 5%, pruning MLP layers leads to the best memory-accuracy trade-off. Across the evaluated models, it reduces memory requirements by 5.6-13%. The largest reduction is achieved for Llama 8B, where memory usage decreases by more than 1.3 GB (13%) while maintaining an accuracy drop below 5% (see Appendix, Figs. 43, 49).

These results show that modest levels of pruning can yield substantial deployment benefits: in our settings, *pruning entire MLPs is the most effective approach to reduce memory footprint (13%), without substantially lowering accuracy, whereas pruning entire attention layers provides the largest reductions in inference time and energy consumption (21%) while maintaining competitive accuracy.*

5 Recommendations

Our experiments provide a comprehensive view of how pruning affects ternary models’ accuracy and efficiency. Notably, we reveal consistent patterns across architectures, datasets, and resource constraints, offering practical deployment guidelines.

Pruning a small number of ternary models’ parameters preserves accuracy while substantially improving efficiency (See Tab. 2 for examples on BitNet 2B and Llama3 8B). Pruning attention layers yields the largest reductions in inference time and energy consumption (21%), while pruning MLP layers achieves the greatest memory savings (13%), both without exceeding a 5% accuracy loss. For example, serving one million queries² with the most energy-efficient pruned BitNet model (removing eight attention layers) reduces estimated CO₂ emissions from 851 to 725 g in the USA, 764 to 650 g

²Individual queries served without batching, assuming an average sequence length of 2,048 tokens.

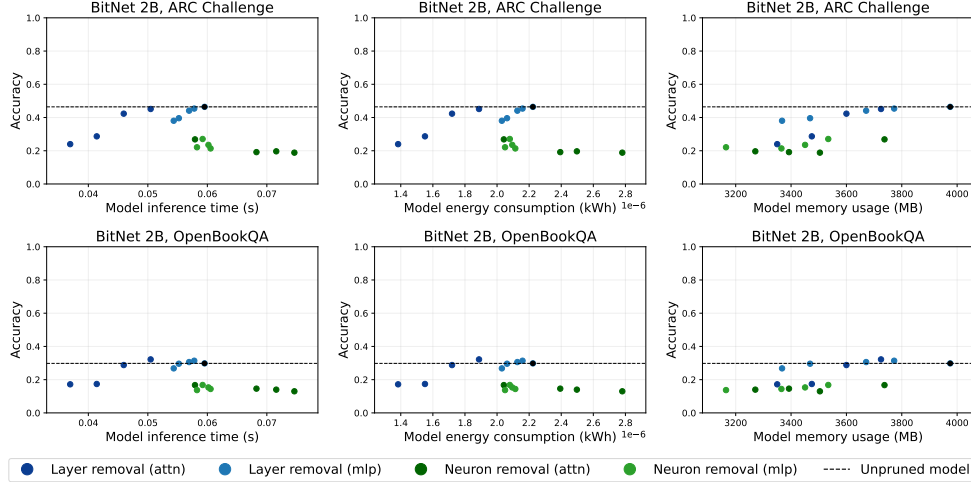


Figure 5: Efficiency of pruned models against model accuracy. The first, second, and third columns show inference time, energy, and memory, respectively, with accuracy on the y-axis. Results on the first row are for ARC challenge. Results on the second row are for OpenbookQA. Coloured dots represent different pruning methods, while the black line indicates unpruned model’s accuracy. Black dot marks time, energy, and memory for the unpruned model.

in Germany, and 1,242 to 1,058 g in China. For the USA, this reduction is equivalent to decreasing the emissions associated with driving a passenger car from approximately 8.0 km to 6.8 km.³

Since entire layers are pruned, we expect efficiency improvements to generalise to inference frameworks such as BitNet.cpp (Wang et al., 2024), which provides optimised kernels for inference with packed ternary models (i.e., models that store multiple ternary weights per byte). In contrast, the other pruning methods would not reduce the memory footprint of packed ternary models. For instance, neuron pruning degrades accuracy, offers poor efficiency trade-offs, and is incompatible with BitNet.cpp, as it alters matrix sizes and breaks optimised kernels, making it unsuitable for low-resource deployment. Unstructured pruning cannot currently yield efficiency gains on low-resource hardware either, as it requires strict N:M sparsity patterns and specialised hardware to improve efficiency. Overall, when balancing efficiency and accuracy, attention layer pruning is the best strategy to minimise time and energy constraints, whereas MLP layer pruning is best for memory constraints.

Lastly, when pruning ternary LLMs, we recommend (i) pruning entire attention or MLP layers first, depending on the deployment constraint. (ii) Increase sparsity moderately (typically below 5%) to maintain accuracy while gaining efficiency. (iii) Evaluate multiple pruning configurations us-

ing both PePR scores and effectiveness-efficiency models comparison to find the optimal trade-off for the target deployment environment.

6 Conclusions

In this work, we studied whether already compact ternary LLMs can be pruned to improve memory footprint, inference time, and energy consumption, while preserving effectiveness. We evaluated six pruning algorithms across six ternary LLMs and seven datasets, analysing accuracy, GPU inference time, energy consumption, and memory footprint. We found that layer pruning can remove a moderate number of parameters (e.g., 5%) from ternary LLMs, improving their efficiency without substantially compromising accuracy. Ternary LLMs are more sensitive to pruning than reported for full-precision models (Hu et al., 2025a), layer pruning yielded the highest accuracy in our experiments. Importantly, pruning specific layer types can help target deployment bottlenecks while preserving accuracy. Across models, the best memory trade-offs within a 5% accuracy drop were achieved by pruning MLP layers, while the best inference time and energy trade-offs were obtained by pruning attention layers, providing practical guidance for deployment in resource-constrained environments.

Future work will explore pruning approaches that combine attention and MLP layer removal to better understand their effectiveness-efficiency trade-offs. Additionally, pruning methods specif-

³CO₂ emissions and driving distances are estimated using country-specific emission factors from CarbonTracker.

ically tailored to ternary models should be developed, including strategies for weight reconstruction that respect ternary constraints, as common methods like SparseGPT cannot be directly applied. Further work will also explore the end-to-end decoding efficiency of pruned ternary models on specific inference frameworks and formats such as BitNet.cpp

Limitations

This study investigates the effectiveness-efficiency trade-offs of pruning ternary models using unstructured, neuron, and layer pruning. While we cover a variety of pruning strategies, additional heuristics within each family could be explored. We evaluated efficiency by tracking GPU time, energy, and memory for matrix multiplications, but other approaches could produce different results, such as measuring CPU resource usage or efficiency in alternative model formats like BitNet.cpp. As our experiments focus on ternary models, the findings may not directly generalise to full-precision models, though studying their effectiveness-efficiency trade-offs would be valuable. Finally, we only evaluated ternary LLMs based on the BitNet architecture (Wang et al., 2023), and further research is needed for other ternary models, such as the Spectra suite (Kaushal et al., 2024).

Ethics Statement

This study focuses on investigating the effectiveness-efficiency trade-offs of pruning ternary LLMs to support future deployment in resource-constrained environments. While pruning improves efficiency and accessibility, these models remain subject to limitations and potential risks, including biases, hallucinations, misinformation, or inappropriate outputs. Models should be deployed responsibly with appropriate safety measures in place. All experiments were conducted on publicly available datasets and models, without using private or sensitive data. We will share our code (upon acceptance) to support reproducibility and further research.

Acknowledgments

This work was funded by the Algorithms, Data & Democracy project (Villum & Velux foundations). We also thank the IR Lab of the University of Copenhagen, in particular, Theresia Veronika Rampisela and Shivam Adarsh for their feedback

on a draft of this paper. Lastly, we thank the anonymous reviewers for their valuable comments and suggestions that helped improve earlier versions of the paper.

References

- Lasse F. Wolff Anthony, Benjamin Kanding, and Raghavendra Selvan. 2020. [Carbontracker: Tracking and Predicting the Carbon Footprint of Training Deep Learning Models](#). *Preprint*, arXiv:2007.03051.
- Saleh Ashkboos, Maximilian Croci, Marcelo Gennari do Nascimento, Torsten Hoefer, and James Hensman. 2024. [SliceGPT: Compress Large Language Models by Deleting Rows and Columns](#). In *International Conference on Learning Representations*, volume 2024, pages 11682–11701.
- Jerry Chee, Yaohui Cai, Volodymyr Kuleshov, and Christopher De Sa. 2023. [QuIP: 2-Bit Quantization of Large Language Models With Guarantees](#). In *Advances in Neural Information Processing Systems*, volume 36, pages 4396–4429. Curran Associates, Inc.
- Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. [BoolQ: Exploring the Surprising Difficulty of Natural Yes/No Questions](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2924–2936, Minneapolis, Minnesota. Association for Computational Linguistics.
- Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. [Think you have Solved Question Answering? Try ARC, the AI2 Reasoning Challenge](#). *Preprint*, arXiv:1803.05457.
- Tim Dettmers, Ruslan A Svirschevski, Vage Egiazarian, Denis Kuznedelev, Elias Frantar, Saleh Ashkboos, Alexander Borzunov, Torsten Hoefer, and Dan Alistarh. 2024. [SpQR: A Sparse-Quantized Representation for Near-Lossless LLM Weight Compression](#). In *The Twelfth International Conference on Learning Representations*.
- Falcon-LLM Team. 2024. [The Falcon 3 Family of Open Models](#).
- Falcon-LLM Team. 2025. [Falcon-E, a series of powerful, universal and fine-tunable 1.58bit language models](#).
- Elias Frantar and Dan Alistarh. 2023. [SparseGPT: Massive Language Models Can be Accurately Pruned in One-Shot](#). In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 10323–10337. PMLR.

- Elias Frantar, Saleh Ashkboos, Torsten Hoefer, and Dan Alistarh. 2023. [OPTQ: Accurate quantization for generative pre-trained transformers](#). In *The Eleventh International Conference on Learning Representations*.
- Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac’h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, and 5 others. 2024. [The Language Model Evaluation Harness](#).
- Song Han, Huizi Mao, and William J. Dally. 2016. [Deep Compression: Compressing Deep Neural Network with Pruning, Trained Quantization and Huffman Coding](#). In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.
- Shwai He, Guoheng Sun, Zheyu Shen, and Ang Li. 2026. [Uncovering the Redundancy in Transformers via a Unified Study of Layer Dropping](#). *Transactions on Machine Learning Research*.
- Hanyu Hu, Pengxiang Zhao, Ping Li, Yi Zheng, Zhefeng Wang, and Xiaoming Yuan. 2025a. [FASP: Fast and Accurate Structured Pruning of Large Language Models](#). *Preprint*, arXiv:2501.09412.
- Lanxiang Hu, Tajana Rosing, and Hao Zhang. 2025b. [TrimLLM: Progressive Layer Dropping for Domain-Specific LLMs](#). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 667–681, Vienna, Austria. Association for Computational Linguistics.
- Ayush Kaushal, Tejas Vaidhya, Arnab Kumar Mondal, Tejas Pandey, Aaryan Bhagat, and Irina Rish. 2024. [Spectra: Surprising effectiveness of pretraining ternary language models at scale](#). *arXiv preprint arXiv:2407.12327*.
- Eldar Kurtic, Daniel Campos, Tuan Nguyen, Elias Frantar, Mark Kurtz, Benjamin Fineran, Michael Goin, and Dan Alistarh. 2022. [The Optimal BERT Surgeon: Scalable and Accurate Second-Order Pruning for Large Language Models](#). In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 4163–4181, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.
- Jiedong Lang, Zhehao Guo, and Shuyu Huang. 2024. [A Comprehensive Study on Quantization Techniques for Large Language Models](#). In *2024 4th International Conference on Artificial Intelligence, Robotics, and Communication (ICAIRC)*, pages 224–231.
- Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, and 1 others. 2025. [Deepseek-v3 technical report](#). *Preprint*, arXiv:2412.19437.
- Yao Lu, Hao Cheng, Yujie Fang, Zeyu Wang, Jiaheng Wei, Dongwei Xu, Qi Xuan, Xiaoniu Yang, and Zhaowei Zhu. 2024. [Reassessing Layer Pruning in LLMs: New Insights and Methods](#). *arXiv preprint arXiv:2411.15558*.
- Shuming Ma, Hongyu Wang, Lingxiao Ma, Lei Wang, Wenhui Wang, Shaohan Huang, Lifeng Dong, Ruiping Wang, Jilong Xue, and Furu Wei. 2024. [The era of 1-bit LLMs: All large language models are in 1.58 bits](#). *Preprint*, arXiv:2402.17764.
- Xinyin Ma, Gongfan Fang, and Xinchao Wang. 2023. [LLM-Pruner: On the Structural Pruning of Large Language Models](#). In *Advances in Neural Information Processing Systems*, volume 36, pages 21702–21720. Curran Associates, Inc.
- Mohamed Mekkouri, Marc Sun, Leandro von Werra, and Thomas Wolf. 2024. [1.58-Bit LLM: A New Era of Extreme Quantization](#).
- Xin Men, Mingyu Xu, Qingyu Zhang, Qianhao Yuan, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. 2025. [ShortGPT: Layers in Large Language Models are More Redundant Than You Expect](#).
- Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. [Pointer Sentinel Mixture Models](#). In *International Conference on Learning Representations*.
- Paul Michel, Omer Levy, and Graham Neubig. 2019. [Are Sixteen Heads Really Better than One?](#) In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.
- Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. [Can a Suit of Armor Conduct Electricity? A New Dataset for Open Book Question Answering](#). In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391, Brussels, Belgium. Association for Computational Linguistics.
- Mistral-AI. 2025. [Magistral](#). *Preprint*, arXiv:2506.10910.
- NVIDIA. 2023. [NVIDIA Docs: Matrix Multiplication Background User’s Guide](#).
- Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. [Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer](#). *Journal of Machine Learning Research*, 21(140):1–67.
- Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. [Winogrande: An adversarial winograd schema challenge at scale](#). *Communications of the ACM*, 64(9):99–106.

- Raghavendra Selvan, Bob Pepin, Christian Igel, Gabrielle Samuel, and Erik B Dam. 2025. [PePR: Performance Per Resource Unit as a Metric to Promote Small-scale Deep Learning](#). In *Proceedings of the 6th Northern Lights Deep Learning Conference (NLDL)*, volume 265 of *Proceedings of Machine Learning Research*, pages 220–229. PMLR.
- Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. 2024. [A Simple and Effective Pruning Approach for Large Language Models](#). In *The Twelfth International Conference on Learning Representations*.
- OpenAI team. 2026. [OpenAI GPT-5 System Card](#). Preprint, arXiv:2601.03267.
- Mart van Baalen, Christos Louizos, Markus Nagel, Rana Ali Amjad, Ying Wang, Tijmen Blankevoort, and Max Welling. 2020. [Bayesian Bits: Unifying Quantization and Pruning](#). In *Advances in Neural Information Processing Systems*, volume 33, pages 5741–5752. Curran Associates, Inc.
- Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2018. [GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding](#). In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.
- Hongyu Wang, Shuming Ma, Li Dong, Shaohan Huang, Huaijie Wang, Lingxiao Ma, Fan Yang, Ruiping Wang, Yi Wu, and Furu Wei. 2023. [Bitnet: Scaling 1-bit transformers for large language models](#). Preprint, arXiv:2310.11453.
- Jinheng Wang, Hansong Zhou, Ting Song, Shaoguang Mao, Shuming Ma, Hongyu Wang, Yan Xia, and Furu Wei. 2024. [1-bit AI infra: Part 1.1, fast and lossless bitnet b1. 58 inference on cpus](#). Preprint, arXiv:2410.16144.
- Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. 2023. [SmoothQuant: Accurate and efficient post-training quantization for large language models](#). In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 38087–38099. PMLR.
- An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025a. [Qwen3 technical report](#). arXiv preprint arXiv:2505.09388.
- Mingzhe Yang, Sihao Lin, Changlin Li, and Xiaojun Chang. 2025b. [Let LLM tell what to prune and how much to prune](#). In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 70833–70849. PMLR.
- Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. [HellaSwag: Can a Machine Really Finish Your Sentence?](#) In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, Florence, Italy. Association for Computational Linguistics.
- Di Zhang, Xun Wu, Shaohan Huang, Yudong Wang, Hanyong Shao, Yingbo Hao, Zewen Chi, Li Dong, Ting Song, Yan Xia, Zhifang Sui, and Furu Wei. 2026. [Sparse-BitNet: 1.58-bit LLMs are Naturally Friendly to Semi-Structured Sparsity](#). Preprint, arXiv:2603.05168.
- Yingtao Zhang, Haoli Bai, Haokun Lin, Jialin Zhao, Lu Hou, and Carlo Vittorio Cannistraci. 2024. [Plug-and-Play: An Efficient Post-training Pruning Method for Large Language Models](#). In *The Twelfth International Conference on Learning Representations*.
- Aojun Zhou, Yukun Ma, Junnan Zhu, Jianbo Liu, Zhijie Zhang, Kun Yuan, Wenxiu Sun, and Hongsheng Li. 2021. [Learning N:M Fine-grained Structured Sparse Neural Networks From Scratch](#). In *International Conference on Learning Representations*.
- Xunyu Zhu, Jian Li, Yong Liu, Can Ma, and Weiping Wang. 2024. [A Survey on Model Compression for Large Language Models](#). *Transactions of the Association for Computational Linguistics*, 12:1556–1577.

A Study Design

We present the pruning thresholds and final sparsities for all configurations of models and pruning algorithms in Tables 3-8.

All experiments are performed on NVIDIA H100, except for effectiveness, which is tracked using both A100 and H100 GPUs.

B Results

Figs. 6-10 show the accuracy of pruned and unpruned ternary models across all datasets.

Figs. 11, 12, and 13 show the average inference time, energy consumption, and memory requirements, respectively, at different sparsity levels for all pruned models.

Figs. 14 to 31 show the PePR time, energy, and memory, respectively, at different sparsity levels for all pruned models.

Figs. 32 to 49 show the inference time, energy consumption, and memory requirements, respectively, against accuracy for all pruned models.

Tabs. 9-14 show a summary of the accuracy, energy consumption, time, and memory measurements obtained for all pruned configurations of the models.

pruner	threshold	sparsity
Layer_removal (attn)	8	0.457
	12	0.475
	16	0.493
	20	0.511
Layer_removal (mlp)	2	0.452
	3	0.468
	5	0.498
	6	0.513
Neuron_removal (attn)	25	0.449
	50	0.482
	62	0.498
	75	0.517
Neuron_removal (mlp)	12	0.464
	15	0.477
	18	0.489
	25	0.519
Wanda	43	0.456
	46	0.482
	48	0.499
	50	0.518

Table 3: Pruning thresholds and final sparsities for bitnet-b1.58-2B-4T

pruner	threshold	sparsity
Layer_removal (attn)	8	0.437
	12	0.450
	16	0.464
	20	0.478
Layer_removal (mlp)	1	0.431
	2	0.452
	3	0.472
	25	0.428
Neuron_removal (attn)	50	0.448
	75	0.469
Neuron_removal (mlp)	6	0.438
	9	0.452
	12	0.466
	15	0.481
Wanda	43	0.443
	46	0.467
	48	0.485

Table 4: Pruning thresholds and final sparsities for Falcon-E-1B-Base

pruner	threshold	sparsity
Layer_removal (attn)	8	0.411
	12	0.418
	16	0.425
	20	0.433
Layer_removal (mlp)	1	0.413
	2	0.430
Neuron_removal (attn)	25	0.409
	50	0.423
Neuron_removal (mlp)	3	0.411
	6	0.426
Wanda	37	0.403
	40	0.415
	41	0.421
	43	0.437

Table 5: Pruning thresholds and final sparsities for Falcon-E-3B-Base

pruner	threshold	sparsity
Layer_removal (attn)	8	0.388
	12	0.403
	16	0.418
	20	0.433
Layer_removal (mlp)	2	0.389
	3	0.406
	4	0.422
	5	0.439
Neuron_removal (attn)	25	0.382
	50	0.411
	62	0.426
Neuron_removal (mlp)	6	0.383
	9	0.398
	12	0.413
	15	0.428
Wanda	37	0.385
	40	0.409
	43	0.437

Table 6: Pruning thresholds and final sparsities for Llama3-8B-1.58-100B-tokens

pruner	threshold	sparsity
Layer_removal (attn)	8	0.342
	12	0.371
	16	0.401
	20	0.430
Layer_removal (mlp)	2	0.333
	4	0.383
	5	0.409
	6	0.434
Neuron_removal (attn)	37	0.337
	62	0.378
	75	0.400
	88	0.423
Neuron_removal (mlp)	12	0.344
	18	0.376
	21	0.392
	28	0.430
Wanda	34	0.344
	37	0.372
	40	0.401
	43	0.430

Table 7: Pruning thresholds and final sparsities for Falcon3-3B-Base-1.58bit

pruner	threshold	sparsity
Layer_removal (attn)	8	0.325
	12	0.335
	16	0.345
	20	0.356
Layer_removal (mlp)	1	0.326
	2	0.349
Neuron_removal (attn)	25	0.320
	50	0.338
Neuron_removal (mlp)	3	0.320
	6	0.337
	9	0.355
Wanda	31	0.320
	34	0.345
	35	0.354

Table 8: Pruning thresholds and final sparsities for Falcon3-7B-Base-1.58bit

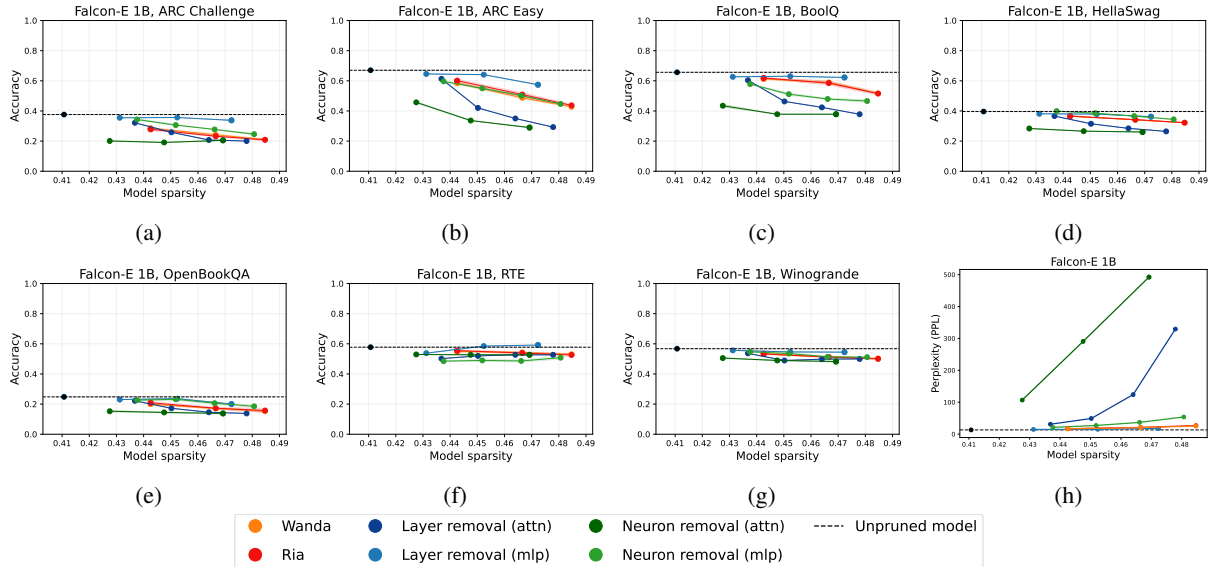


Figure 6: Panels (a)-(g): **Accuracy** of pruned and unpruned **Falcon-E 1B** on seven datasets (y-axis), plotted against the **relative sparsity** of the pruned models compared to the original model (x-axis). Panel (h): **Perplexity** of pruned and unpruned BitNet on the Wikitext-2 dataset (y-axis) versus relative sparsity (x-axis). Unpruned models are shown with dashed lines. All shaded regions indicate variability across three runs ($\pm$ standard deviation).

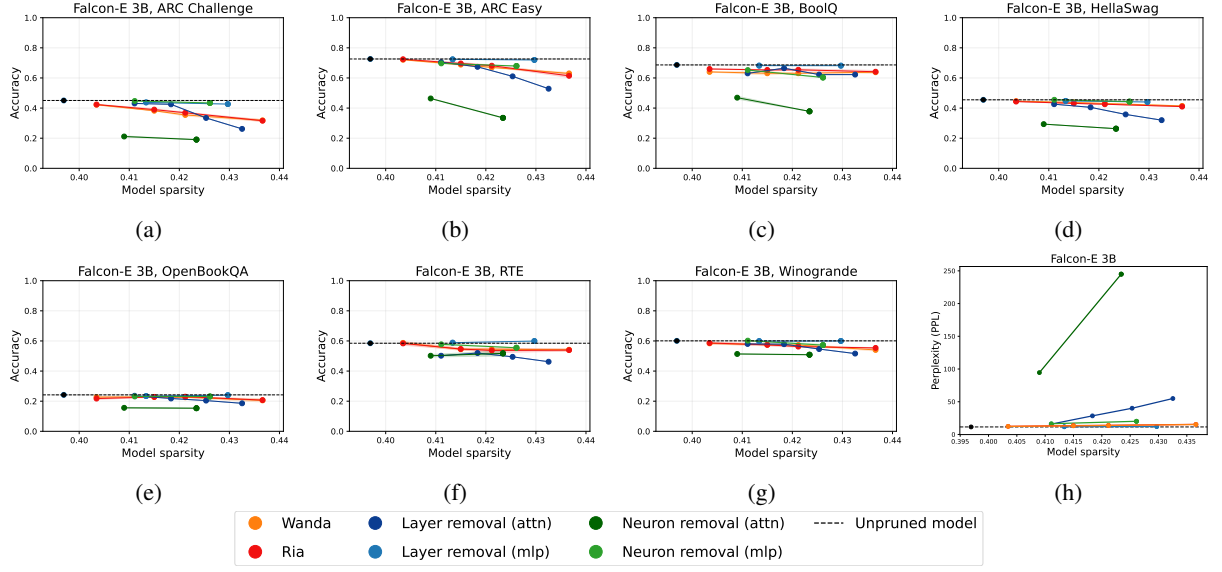


Figure 7: Panels (a)-(g): **Accuracy** of pruned and unpruned **Falcon-E 3B** on seven datasets (y-axis), plotted against the **relative sparsity** of the pruned models compared to the original model (x-axis). Panel (h): **Perplexity** of pruned and unpruned BitNet on the Wikitext-2 dataset (y-axis) versus relative sparsity (x-axis). Unpruned models are shown with dashed lines. All shaded regions indicate variability across three runs ($\pm$ standard deviation).

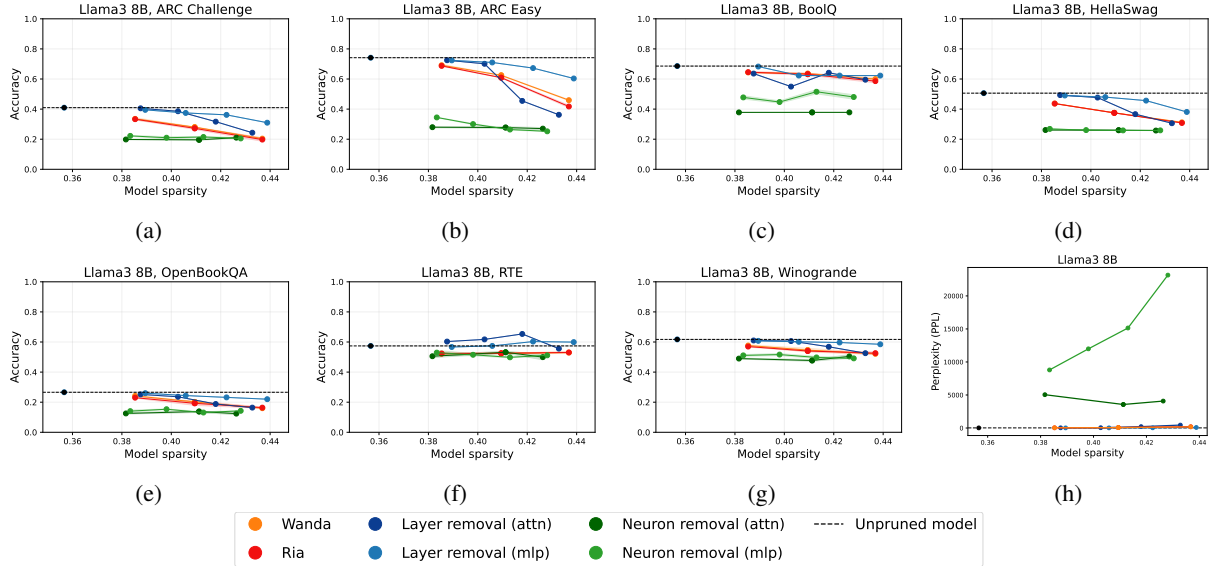


Figure 8: Panels (a)-(g): **Accuracy** of pruned and unpruned **Llama3** on seven datasets (y-axis), plotted against the **relative sparsity** of the pruned models compared to the original model (x-axis). Panel (h): **Perplexity** of pruned and unpruned BitNet on the Wikitext-2 dataset (y-axis) versus relative sparsity (x-axis). Unpruned models are shown with dashed lines. All shaded regions indicate variability across three runs ($\pm$ standard deviation).

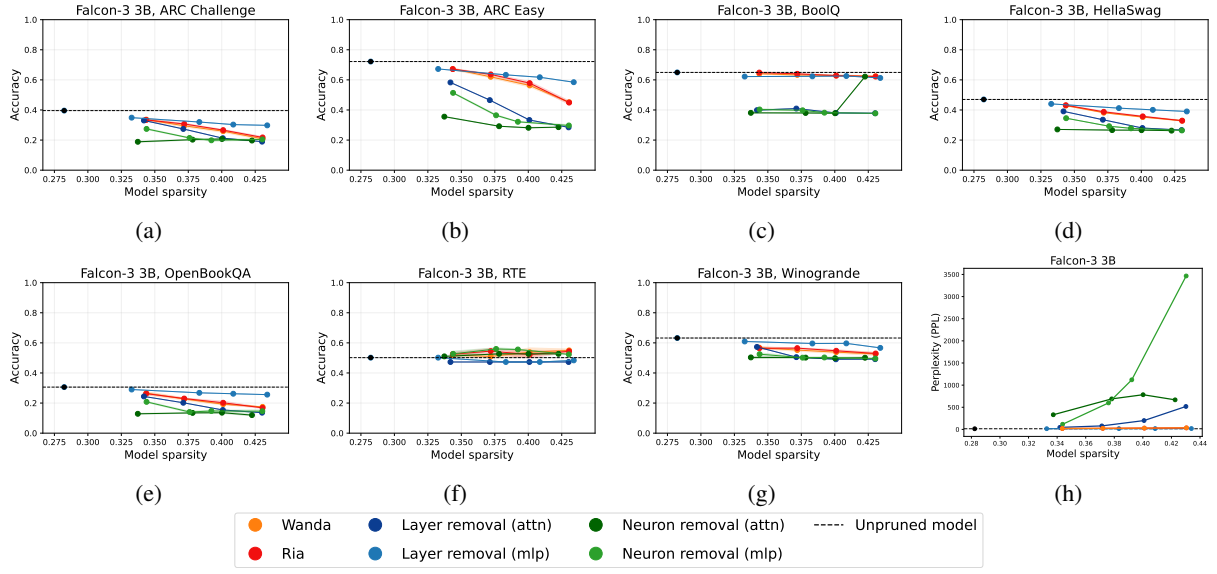


Figure 9: Panels (a)-(g): **Accuracy** of pruned and unpruned **Falcon3 3B** on seven datasets (y-axis), plotted against the **relative sparsity** of the pruned models compared to the original model (x-axis). Panel (h): **Perplexity** of pruned and unpruned BitNet on the Wikitext-2 dataset (y-axis) versus relative sparsity (x-axis). Unpruned models are shown with dashed lines. All shaded regions indicate variability across three runs ($\pm$ standard deviation).

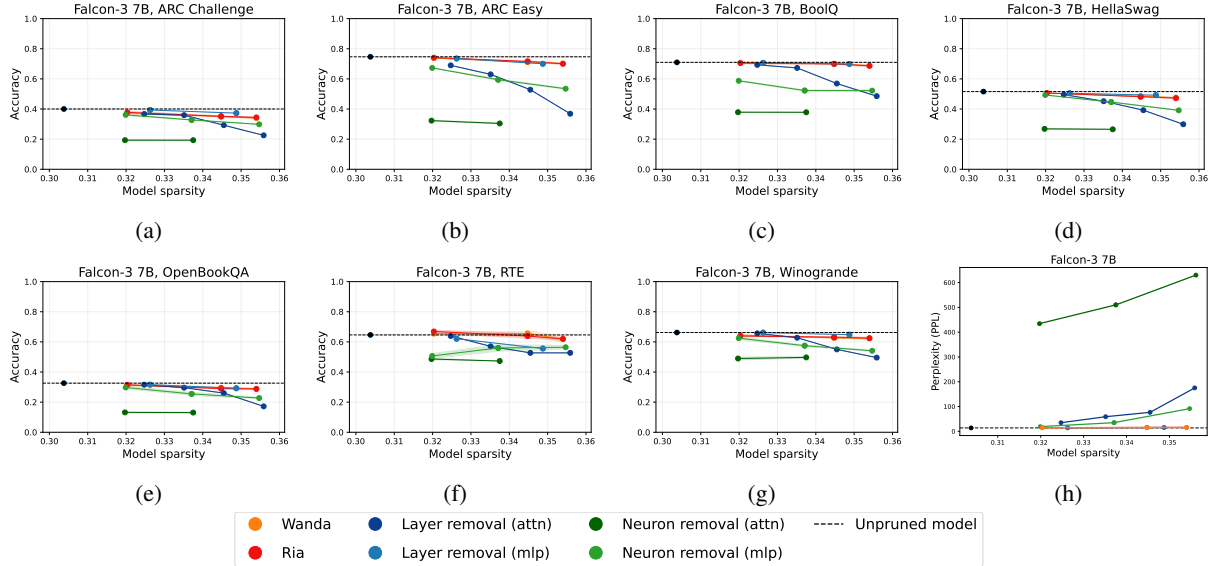


Figure 10: Panels (a)-(g): **Accuracy** of pruned and unpruned **Falcon3 7B** on seven datasets (y-axis), plotted against the **relative sparsity** of the pruned models compared to the original model (x-axis). Panel (h): **Perplexity** of pruned and unpruned BitNet on the Wikitext-2 dataset (y-axis) versus relative sparsity (x-axis). Unpruned models are shown with dashed lines. All shaded regions indicate variability across three runs ($\pm$ standard deviation).

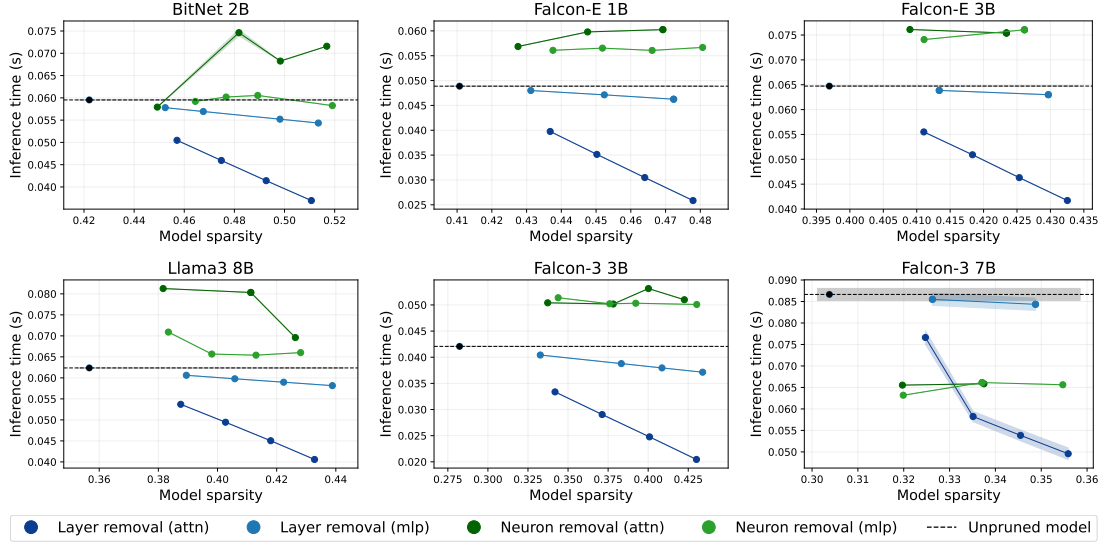


Figure 11: Inference time of pruned models as a function of sparsity (x-axis). Coloured lines represent different pruning methods, while the black dotted line indicates the inference time for the unpruned model. The black dot marks the sparsity of the original unpruned model. All shaded regions indicate variability across 128 samples ($\pm$ standard deviation).

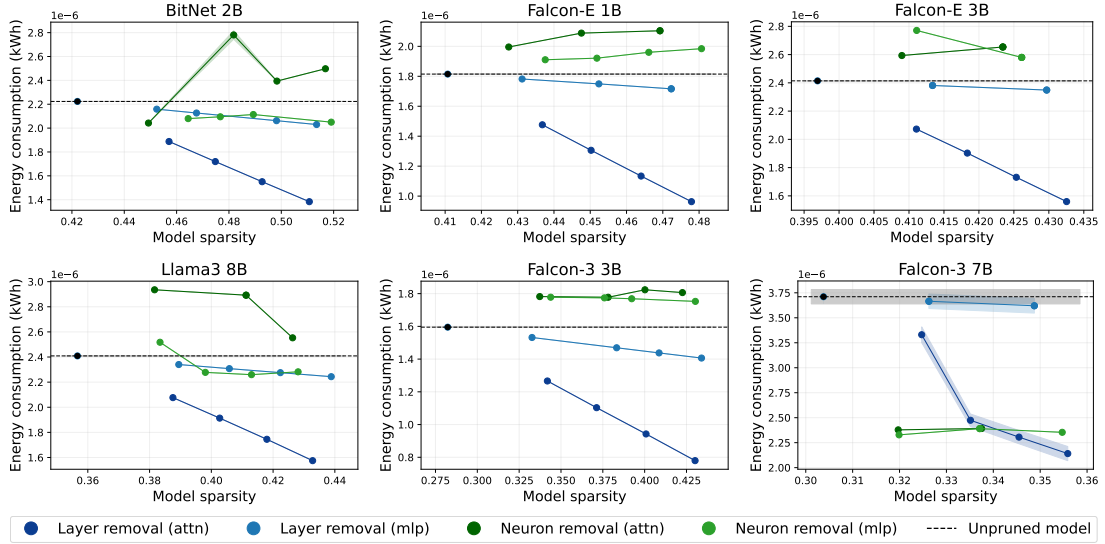


Figure 12: Energy consumption of pruned models as a function of sparsity (x-axis). Coloured lines represent different pruning methods, while the black dotted line indicates the energy consumption for the unpruned model. The black dot marks the sparsity of the original unpruned model. All shaded regions indicate variability across 128 samples ($\pm$ standard deviation).

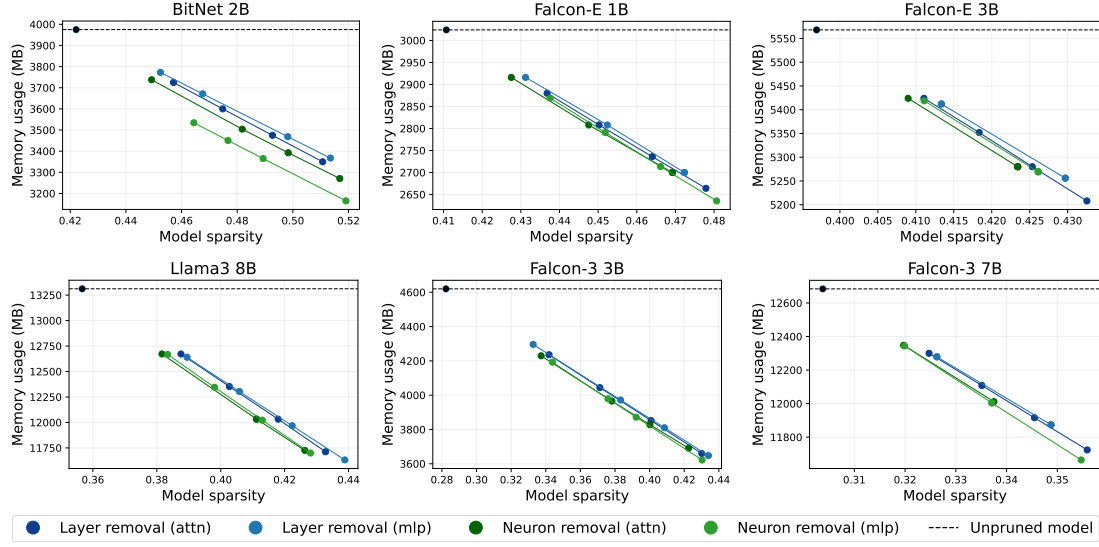


Figure 13: Memory requirements of pruned models as a function of sparsity (x-axis). Coloured lines represent different pruning methods, while the black dotted line indicates the memory requirements for the unpruned model. The black dot marks the sparsity of the original unpruned model.

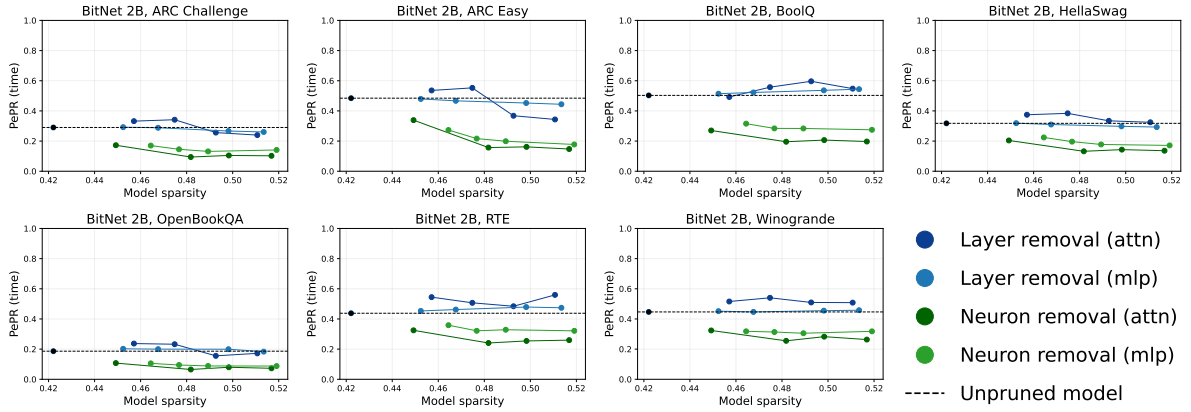


Figure 14: PePR time score of pruned models as a function of sparsity (x-axis). Coloured lines represent different pruning methods, while the black dotted line indicates the PePR time score for the unpruned model. The black dot marks the sparsity of the original unpruned model.

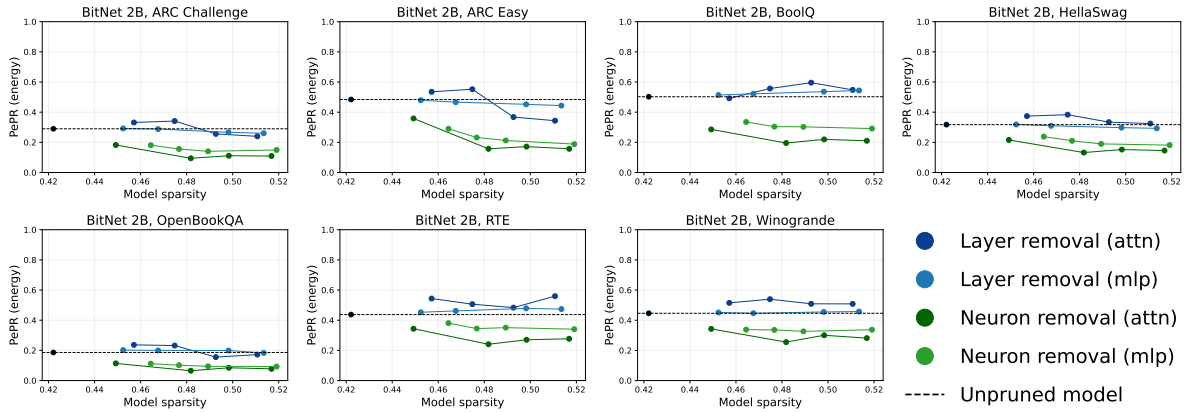


Figure 15: PePR energy score of pruned models as a function of sparsity (x-axis). Coloured lines represent different pruning methods, while the black dotted line indicates the PePR energy score for the unpruned model. The black dot marks the sparsity of the original unpruned model.

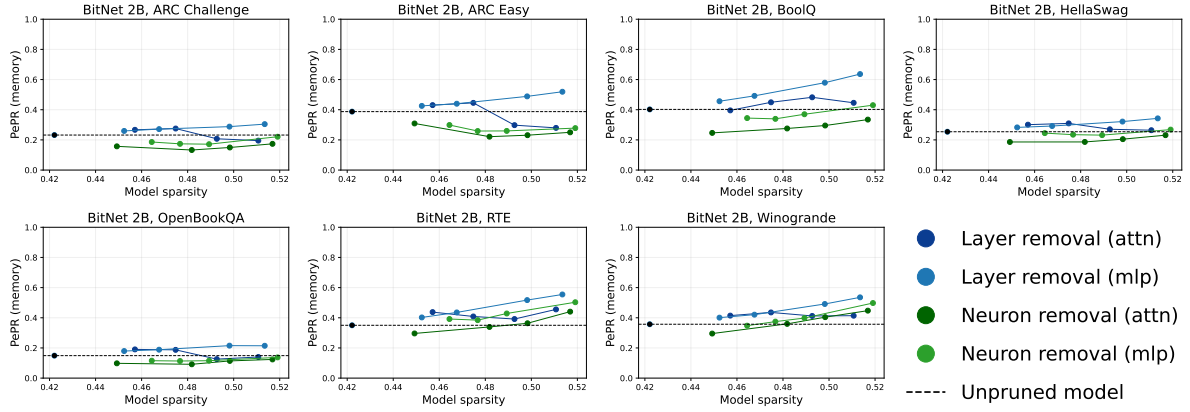


Figure 16: PePR memory score of pruned models as a function of sparsity (x-axis). Coloured lines represent different pruning methods, while the black dotted line indicates the PePR memory score for the unpruned model. The black dot marks the sparsity of the original unpruned model.

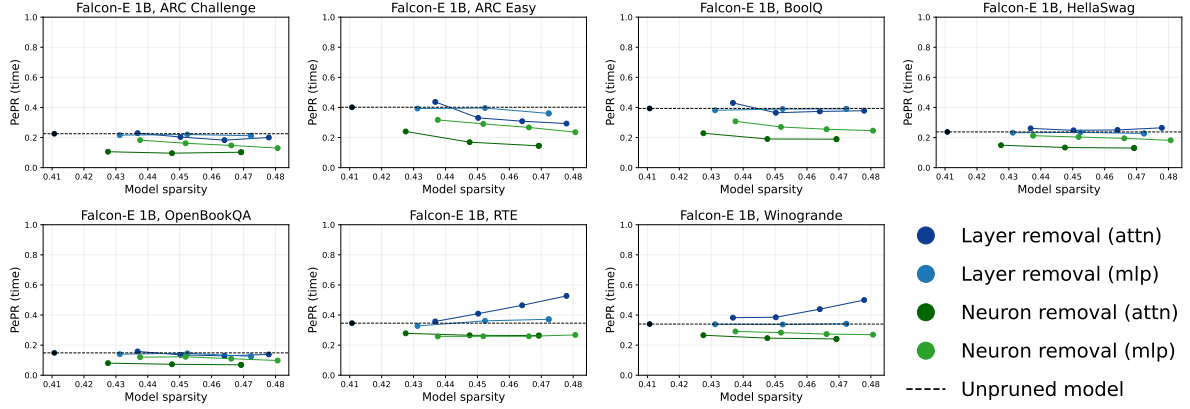


Figure 17: PePR time score of pruned models as a function of sparsity (x-axis). Coloured lines represent different pruning methods, while the black dotted line indicates the PePR time score for the unpruned model. The black dot marks the sparsity of the original unpruned model.

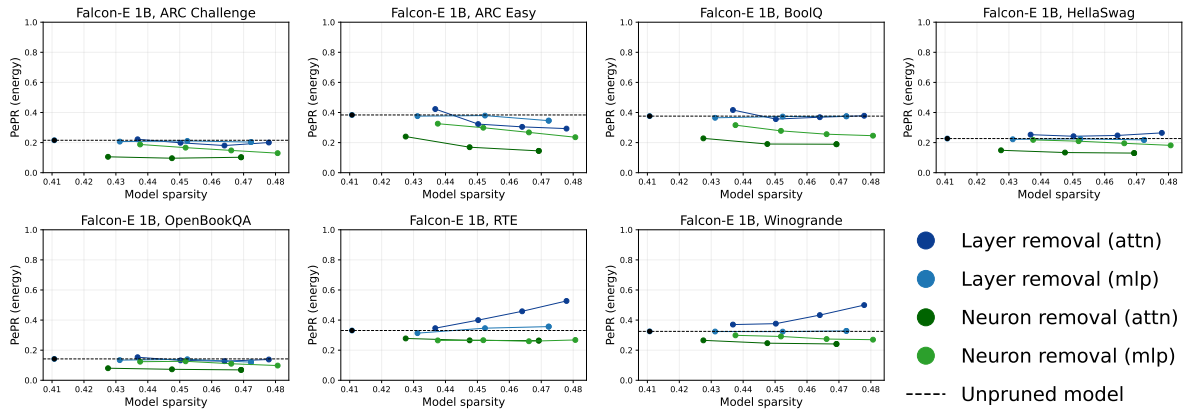


Figure 18: PePR energy score of pruned models as a function of sparsity (x-axis). Coloured lines represent different pruning methods, while the black dotted line indicates the PePR energy score for the unpruned model. The black dot marks the sparsity of the original unpruned model.

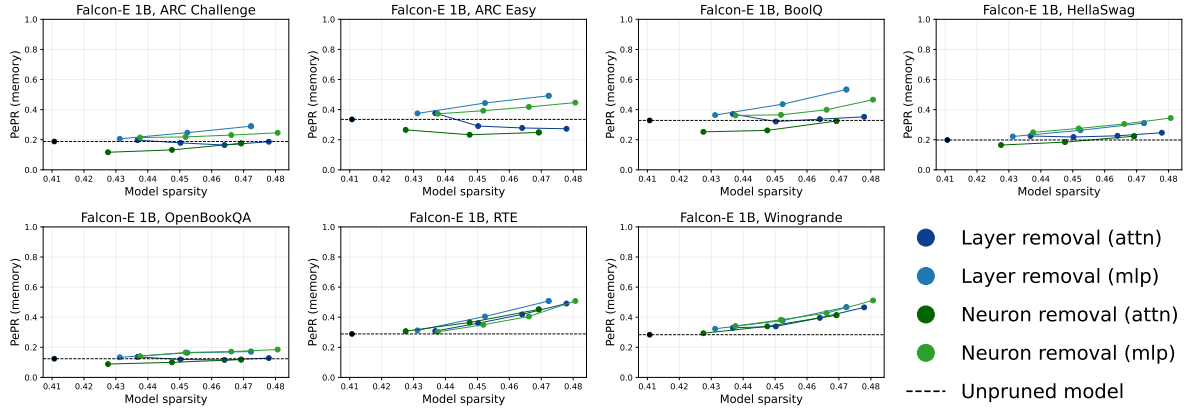


Figure 19: PePR memory score of pruned models as a function of sparsity (x-axis). Coloured lines represent different pruning methods, while the black dotted line indicates the PePR memory score for the unpruned model. The black dot marks the sparsity of the original unpruned model.

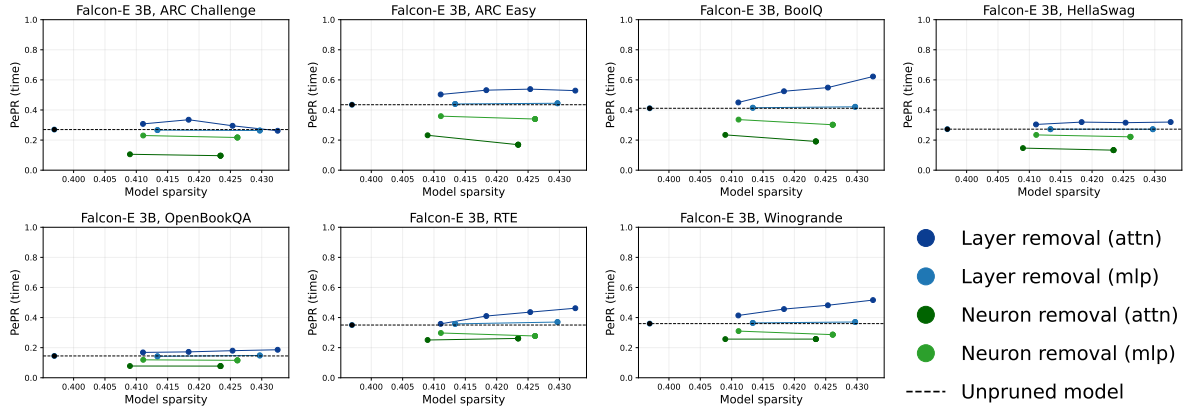


Figure 20: PePR time score of pruned models as a function of sparsity (x-axis). Coloured lines represent different pruning methods, while the black dotted line indicates the PePR time score for the unpruned model. The black dot marks the sparsity of the original unpruned model.

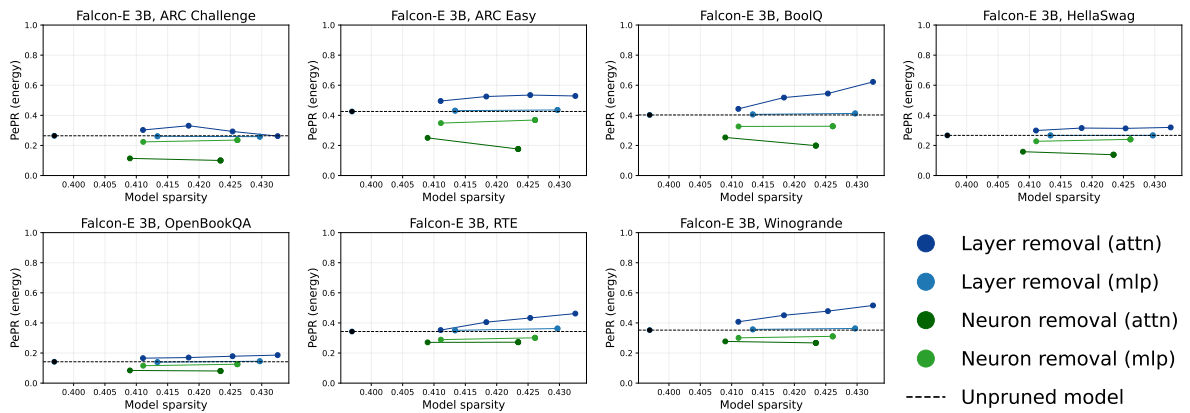


Figure 21: PePR energy score of pruned models as a function of sparsity (x-axis). Coloured lines represent different pruning methods, while the black dotted line indicates the PePR energy score for the unpruned model. The black dot marks the sparsity of the original unpruned model.

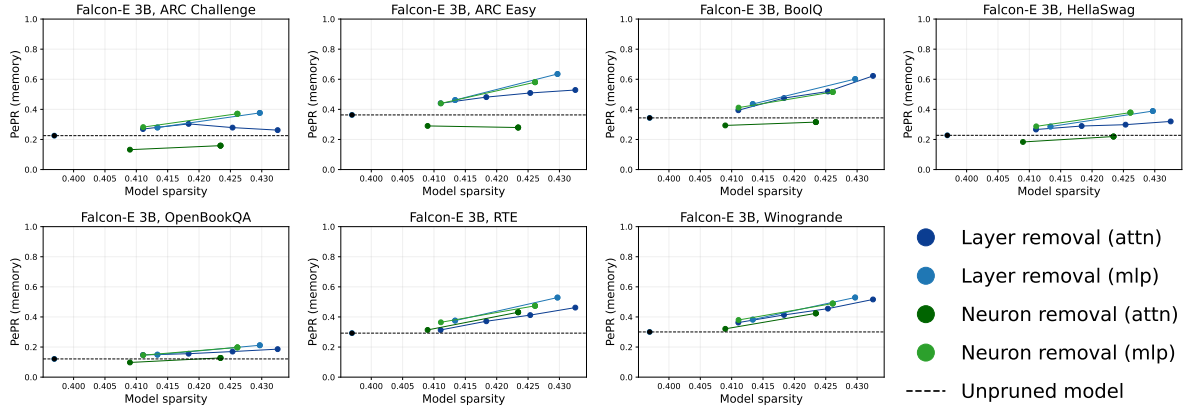


Figure 22: PePR memory score of pruned models as a function of sparsity (x-axis). Coloured lines represent different pruning methods, while the black dotted line indicates the PePR memory score for the unpruned model. The black dot marks the sparsity of the original unpruned model.

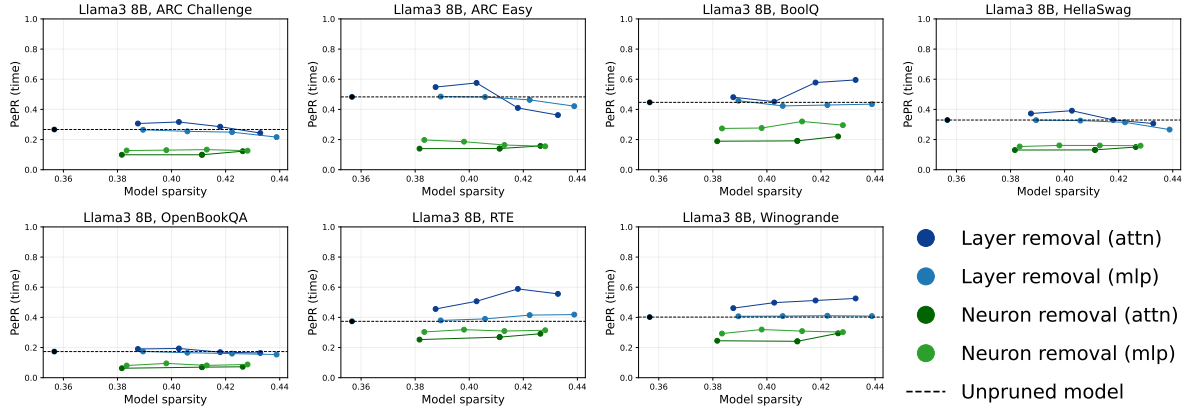


Figure 23: PePR time score of pruned models as a function of sparsity (x-axis). Coloured lines represent different pruning methods, while the black dotted line indicates the PePR time score for the unpruned model. The black dot marks the sparsity of the original unpruned model.

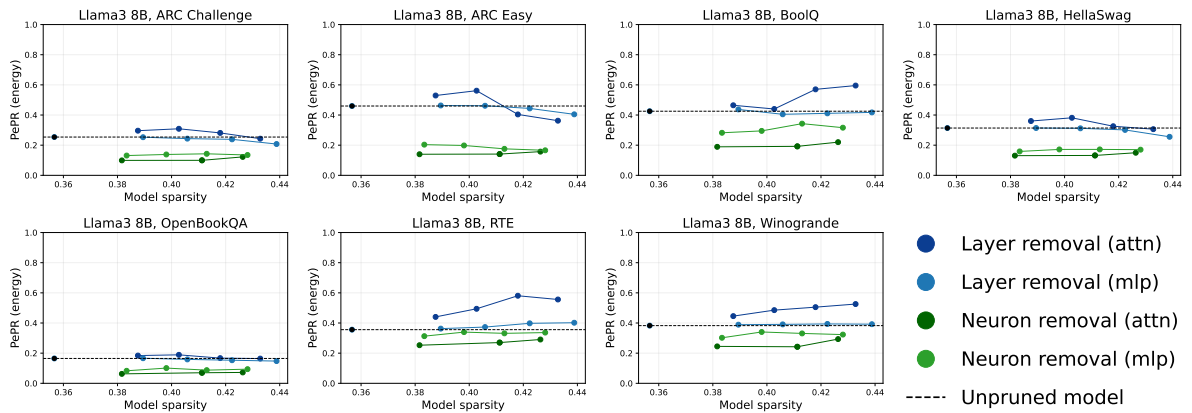


Figure 24: PePR energy score of pruned models as a function of sparsity (x-axis). Coloured lines represent different pruning methods, while the black dotted line indicates the PePR energy score for the unpruned model. The black dot marks the sparsity of the original unpruned model.

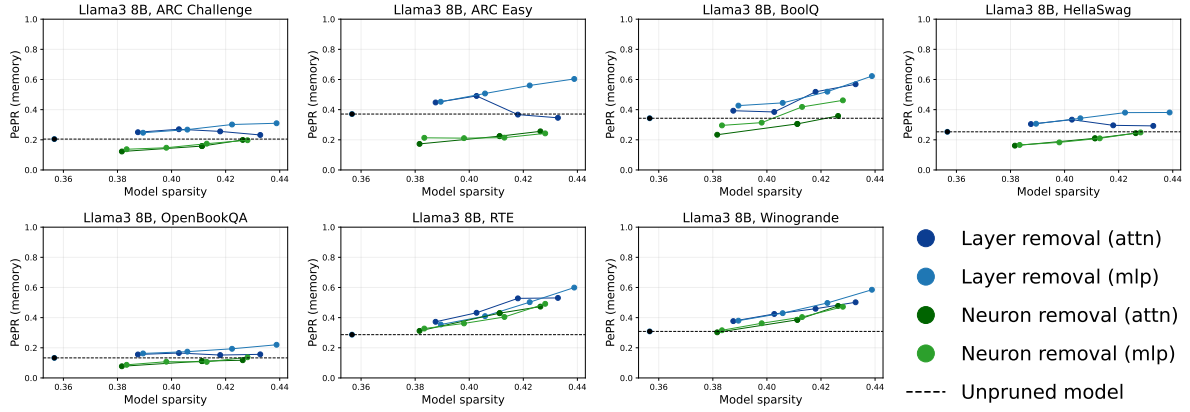


Figure 25: PePR memory score of pruned models as a function of sparsity (x-axis). Coloured lines represent different pruning methods, while the black dotted line indicates the PePR memory score for the unpruned model. The black dot marks the sparsity of the original unpruned model.

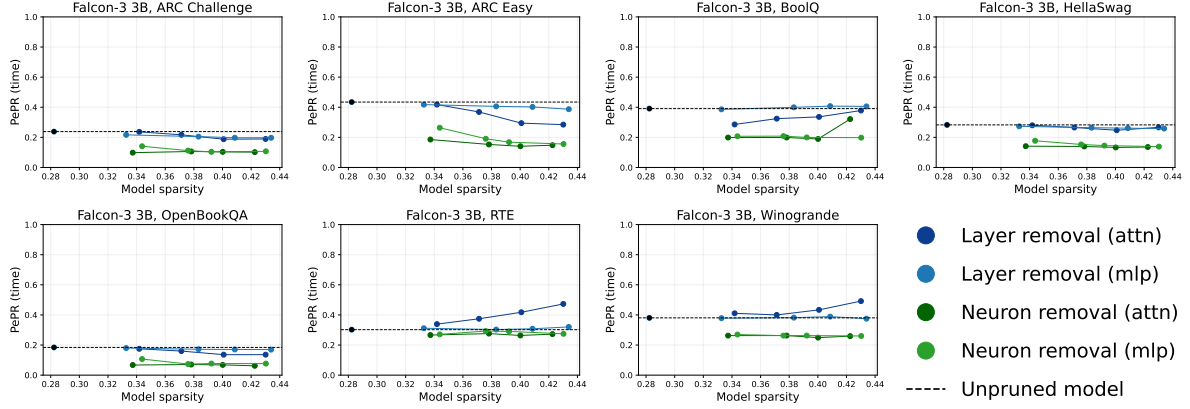


Figure 26: PePR time score of pruned models as a function of sparsity (x-axis). Coloured lines represent different pruning methods, while the black dotted line indicates the PePR time score for the unpruned model. The black dot marks the sparsity of the original unpruned model.

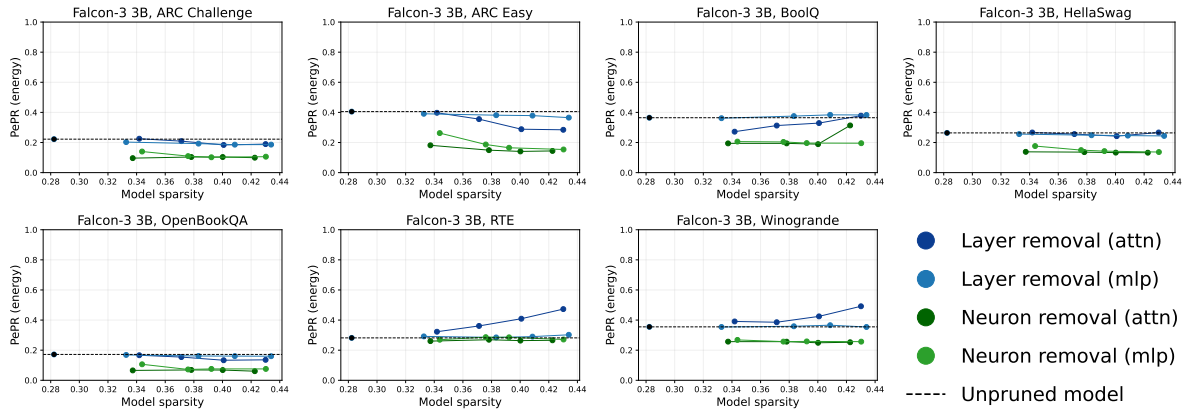


Figure 27: PePR energy score of pruned models as a function of sparsity (x-axis). Coloured lines represent different pruning methods, while the black dotted line indicates the PePR energy score for the unpruned model. The black dot marks the sparsity of the original unpruned model.

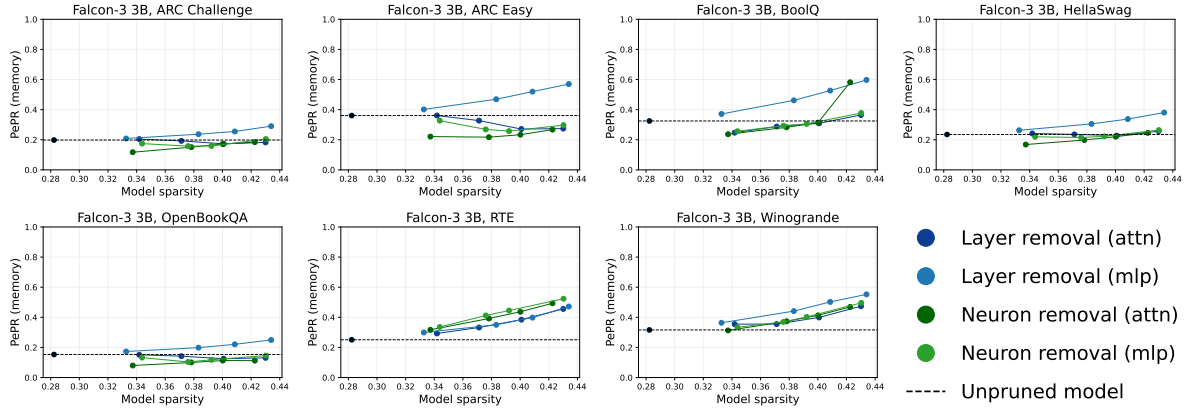


Figure 28: PePR memory score of pruned models as a function of sparsity (x-axis). Coloured lines represent different pruning methods, while the black dotted line indicates the PePR memory score for the unpruned model. The black dot marks the sparsity of the original unpruned model.

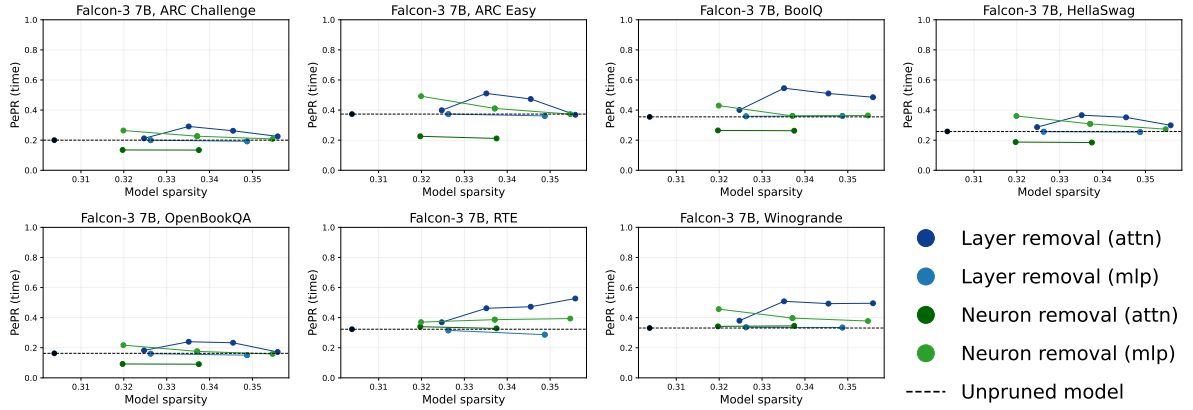


Figure 29: PePR time score of pruned models as a function of sparsity (x-axis). Coloured lines represent different pruning methods, while the black dotted line indicates the PePR time score for the unpruned model. The black dot marks the sparsity of the original unpruned model.

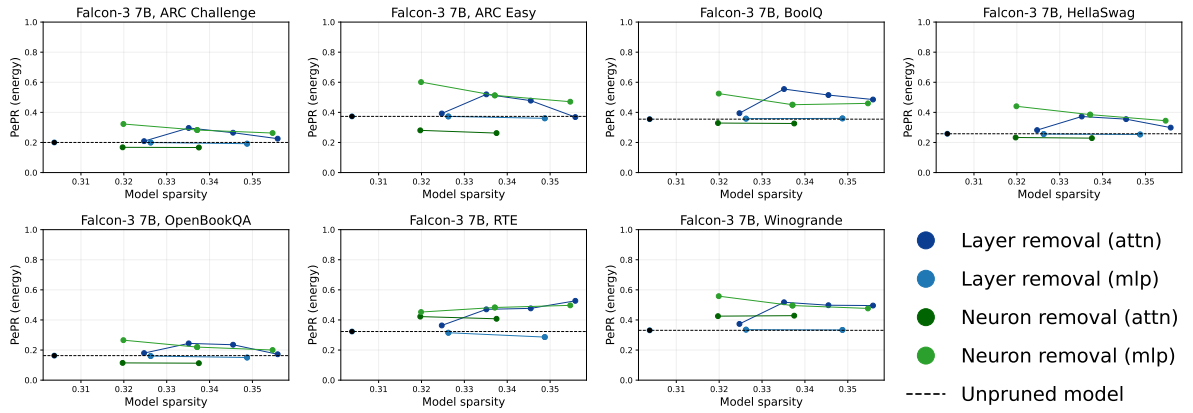


Figure 30: PePR energy score of pruned models as a function of sparsity (x-axis). Coloured lines represent different pruning methods, while the black dotted line indicates the PePR energy score for the unpruned model. The black dot marks the sparsity of the original unpruned model.

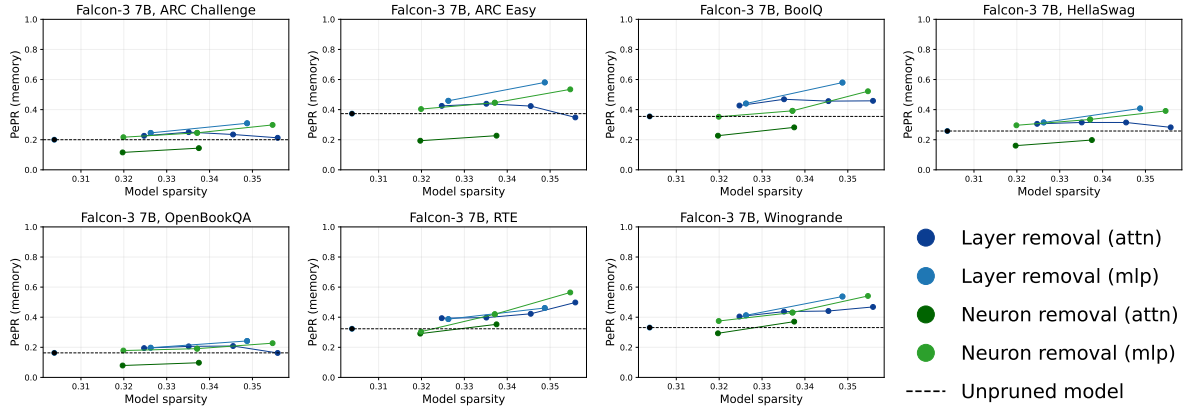


Figure 31: PePR memory score of pruned models as a function of sparsity (x-axis). Coloured lines represent different pruning methods, while the black dotted line indicates the PePR memory score for the unpruned model. The black dot marks the sparsity of the original unpruned model.

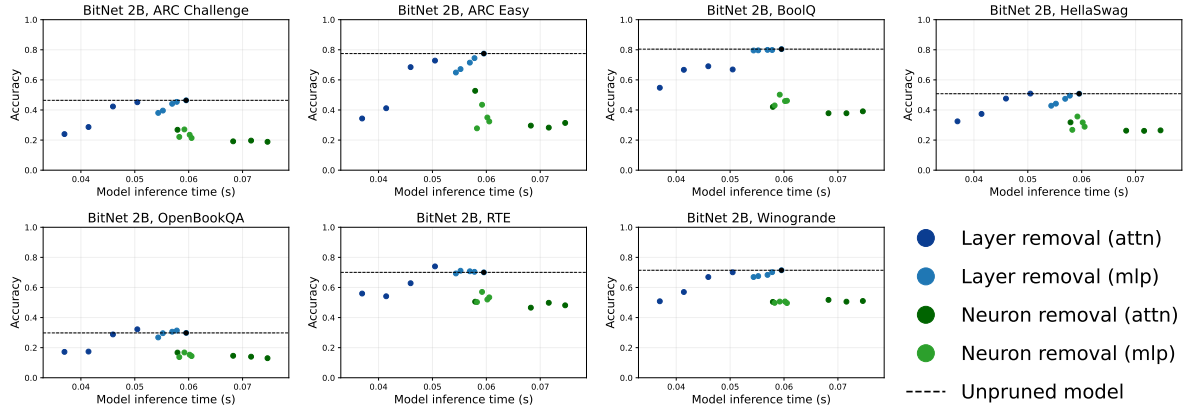


Figure 32: Inference time of pruned models against model accuracy (y-axis). Coloured dots represent different pruning methods, while the black line indicates the unpruned model accuracy. The black dot marks the inference time required by the unpruned model.

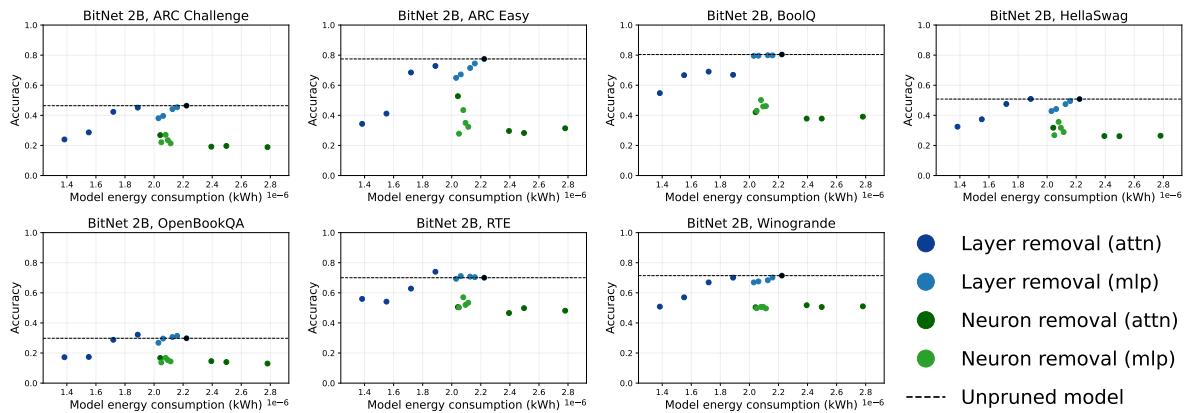


Figure 33: Energy consumption of pruned models against model accuracy (y-axis). Coloured dots represent different pruning methods, while the black line indicates the unpruned model accuracy. The black dot marks the energy consumption of the unpruned model.

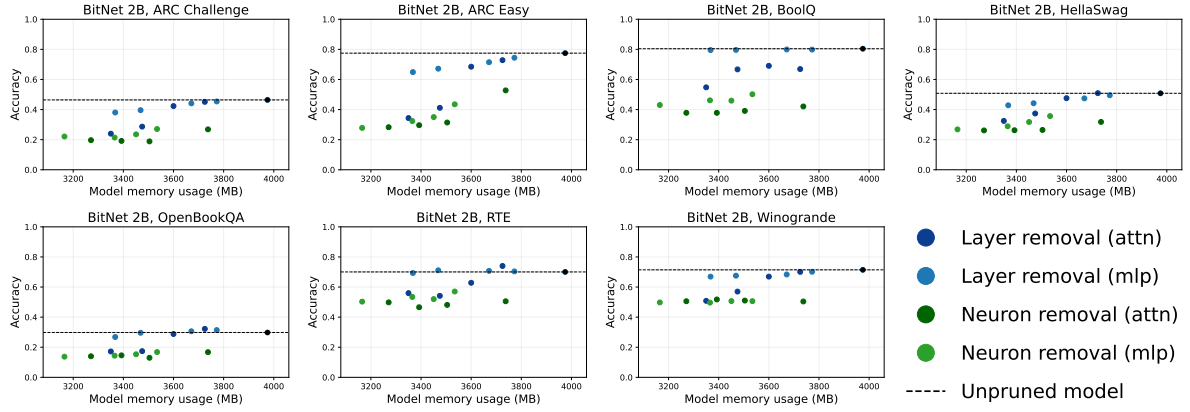


Figure 34: Memory requirements of pruned models against model accuracy (y-axis). Coloured dots represent different pruning methods, while the black line indicates the unpruned model accuracy. The black dot marks the memory requirements of the unpruned model.

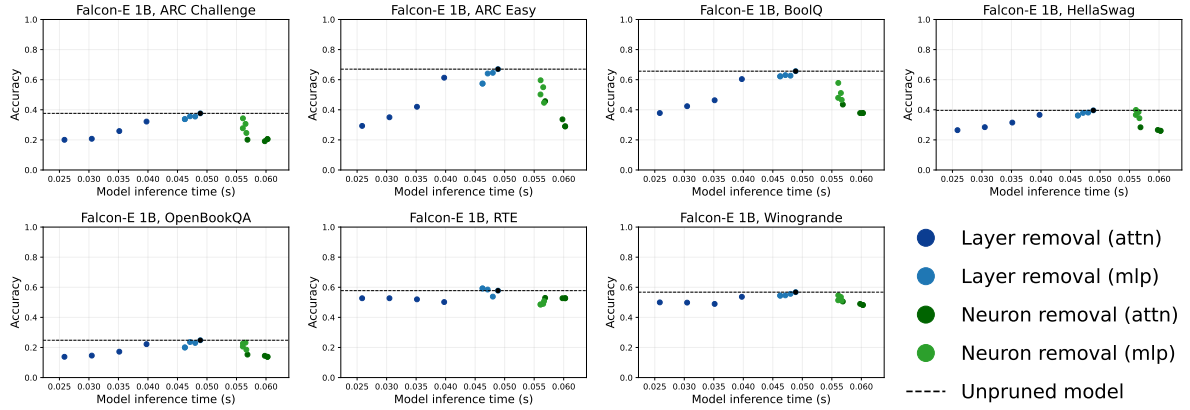


Figure 35: Inference time of pruned models against model accuracy (y-axis). Coloured dots represent different pruning methods, while the black line indicates the unpruned model accuracy. The black dot marks the inference time required by the unpruned model.

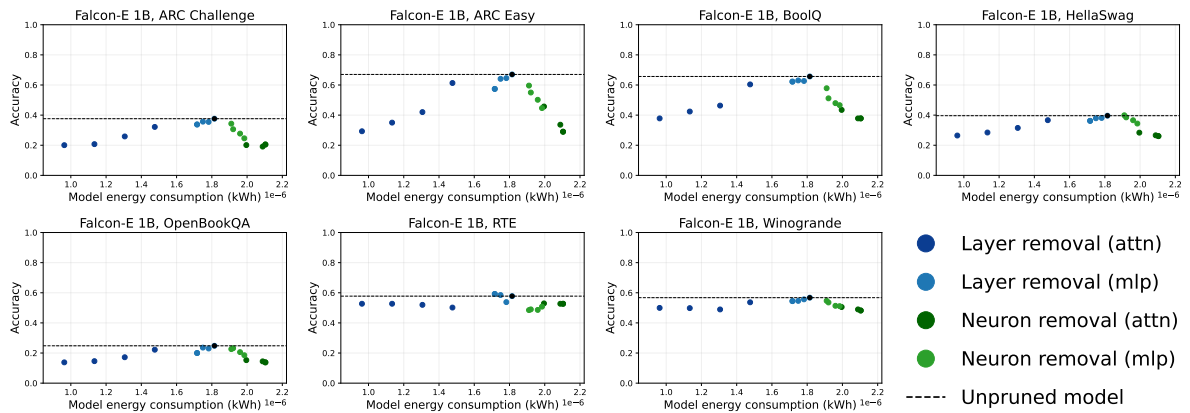


Figure 36: Energy consumption of pruned models against model accuracy (y-axis). Coloured dots represent different pruning methods, while the black line indicates the unpruned model accuracy. The black dot marks the energy consumption of the unpruned model.

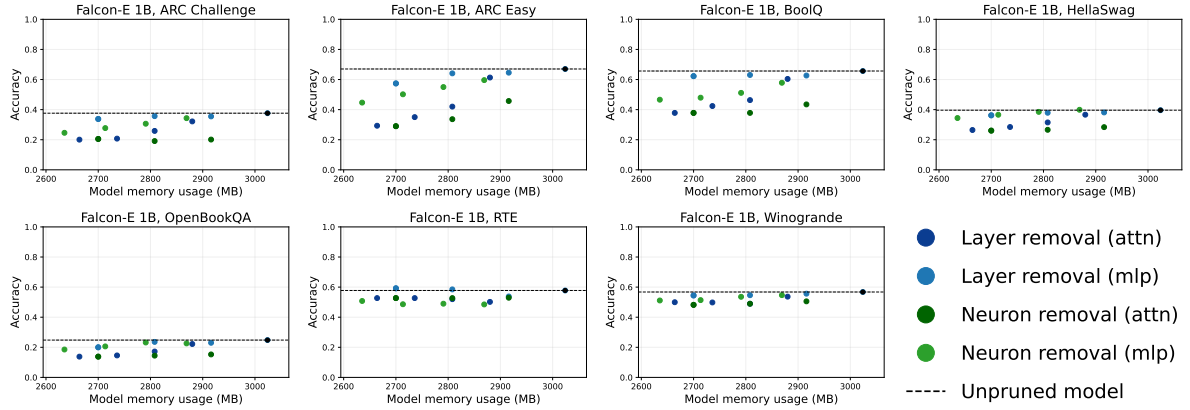


Figure 37: Memory requirements of pruned models against model accuracy (y-axis). Coloured dots represent different pruning methods, while the black line indicates the unpruned model accuracy. The black dot marks the memory requirements of the unpruned model.

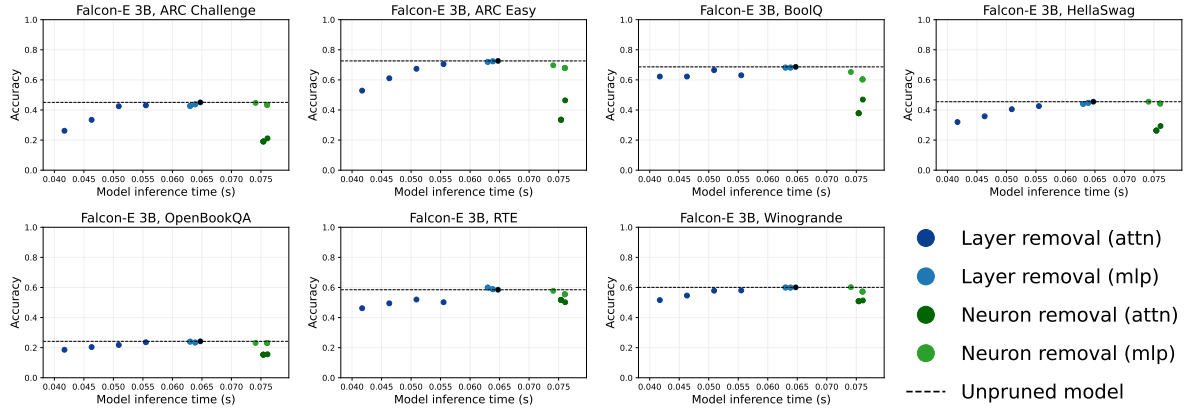


Figure 38: Inference time of pruned models against model accuracy (y-axis). Coloured dots represent different pruning methods, while the black line indicates the unpruned model accuracy. The black dot marks the inference time required by the unpruned model.

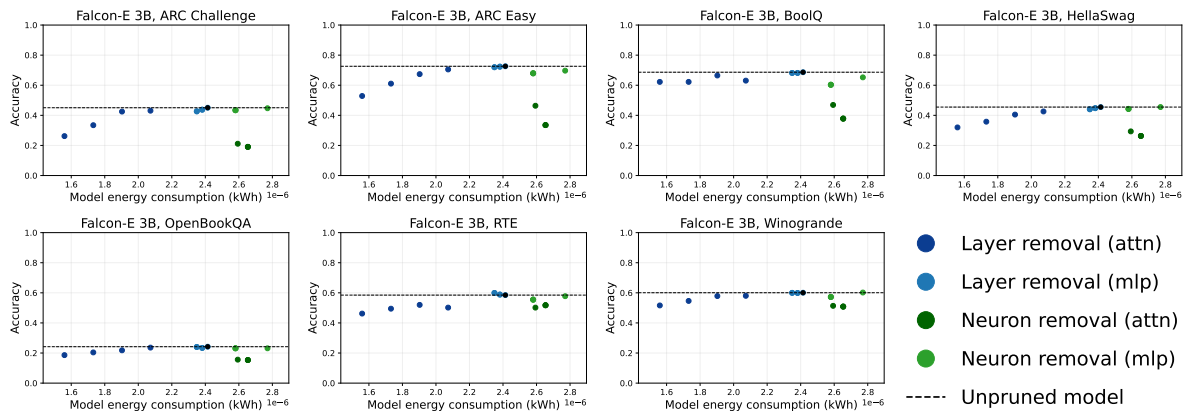


Figure 39: Energy consumption of pruned models against model accuracy (y-axis). Coloured dots represent different pruning methods, while the black line indicates the unpruned model accuracy. The black dot marks the energy consumption of the unpruned model.

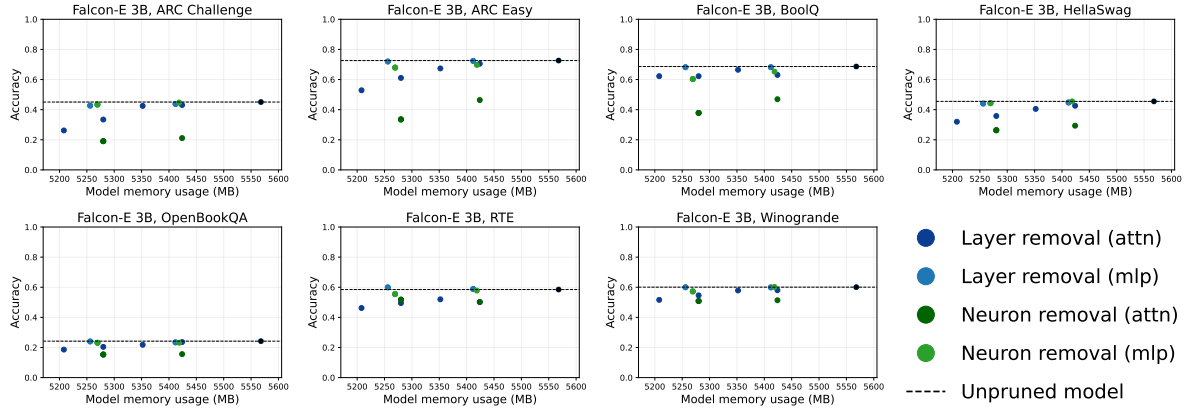


Figure 40: Memory requirements of pruned models against model accuracy (y-axis). Coloured dots represent different pruning methods, while the black line indicates the unpruned model accuracy. The black dot marks the memory requirements of the unpruned model.

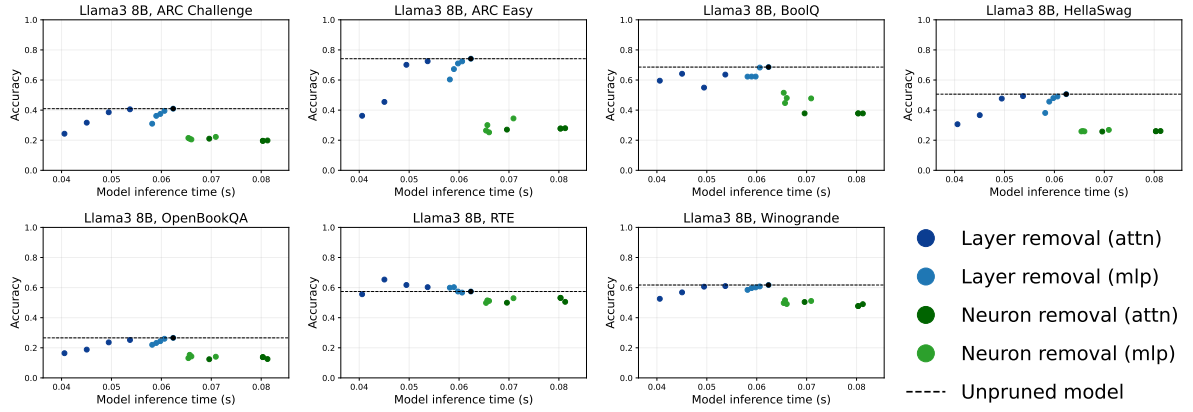


Figure 41: Inference time of pruned models against model accuracy (y-axis). Coloured dots represent different pruning methods, while the black line indicates the unpruned model accuracy. The black dot marks the inference time required by the unpruned model.

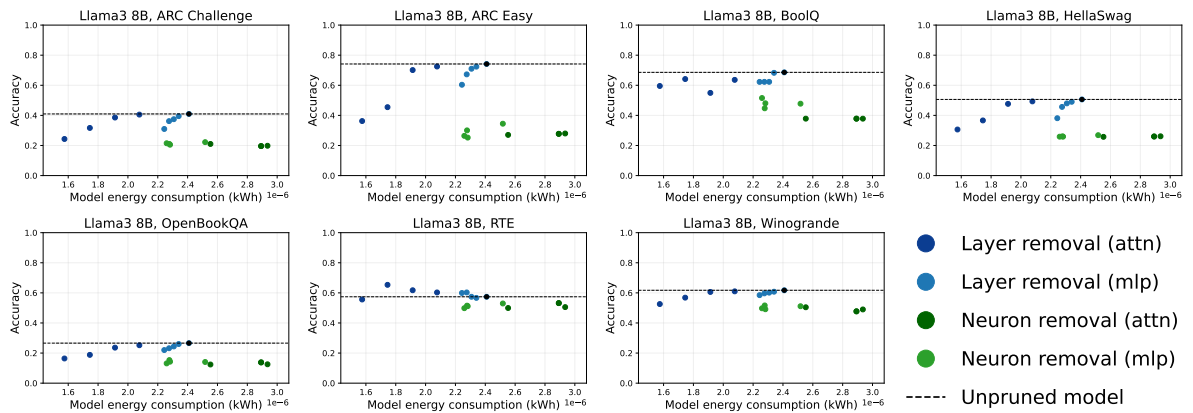


Figure 42: Energy consumption of pruned models against model accuracy (y-axis). Coloured dots represent different pruning methods, while the black line indicates the unpruned model accuracy. The black dot marks the energy consumption of the unpruned model.

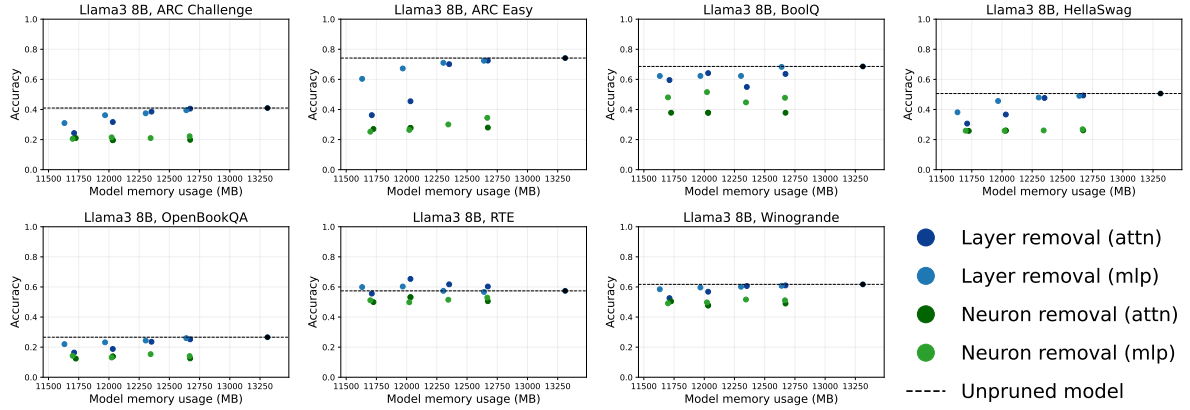


Figure 43: Memory requirements of pruned models against model accuracy (y-axis). Coloured dots represent different pruning methods, while the black line indicates the unpruned model accuracy. The black dot marks the memory requirements of the unpruned model.

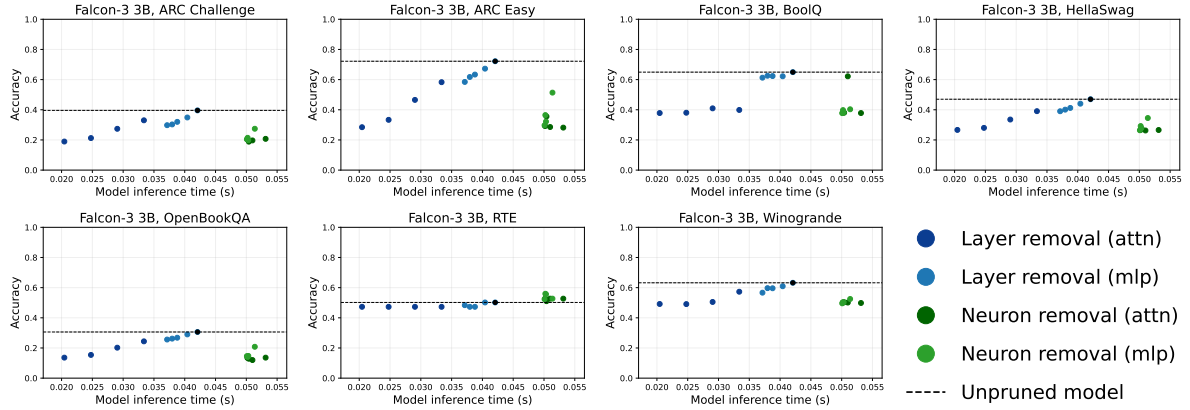


Figure 44: Inference time of pruned models against model accuracy (y-axis). Coloured dots represent different pruning methods, while the black line indicates the unpruned model accuracy. The black dot marks the inference time required by the unpruned model.

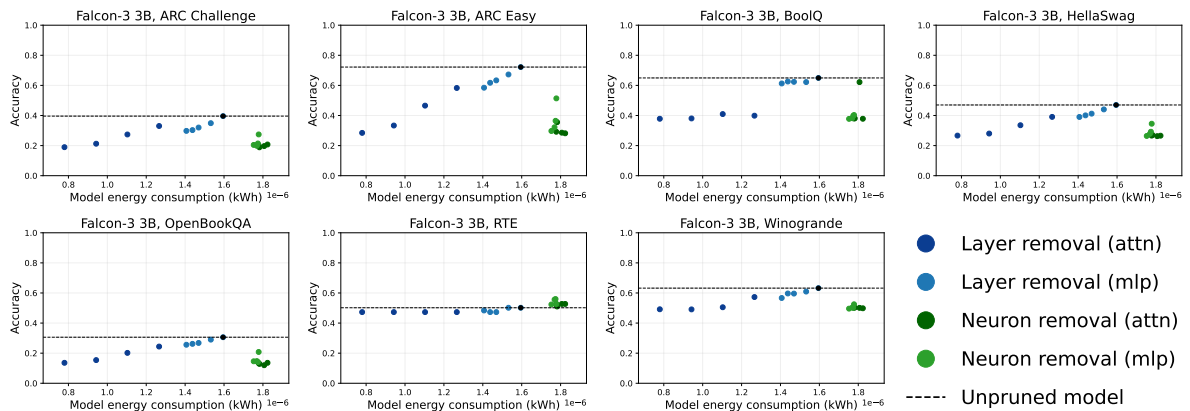


Figure 45: Energy consumption of pruned models against model accuracy (y-axis). Coloured dots represent different pruning methods, while the black line indicates the unpruned model accuracy. The black dot marks the energy consumption of the unpruned model.

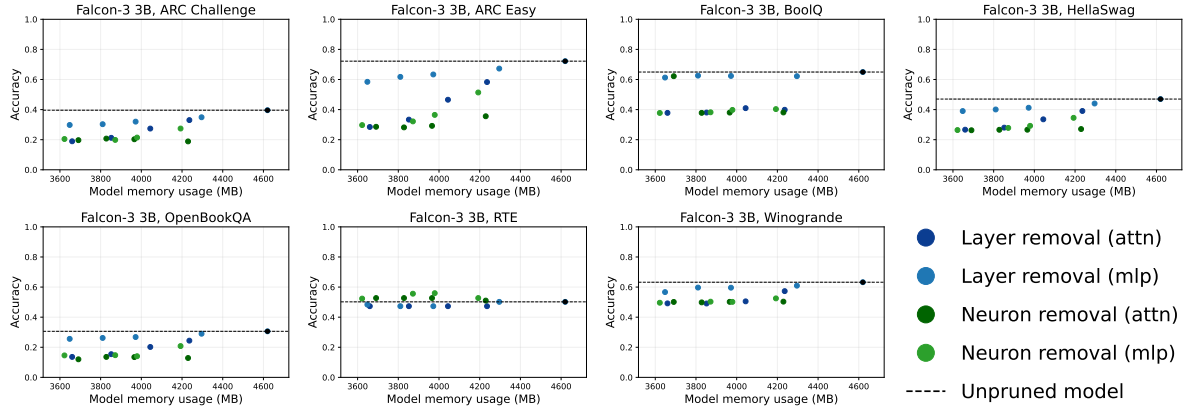


Figure 46: Memory requirements of pruned models against model accuracy (y-axis). Coloured dots represent different pruning methods, while the black line indicates the unpruned model accuracy. The black dot marks the memory requirements of the unpruned model.

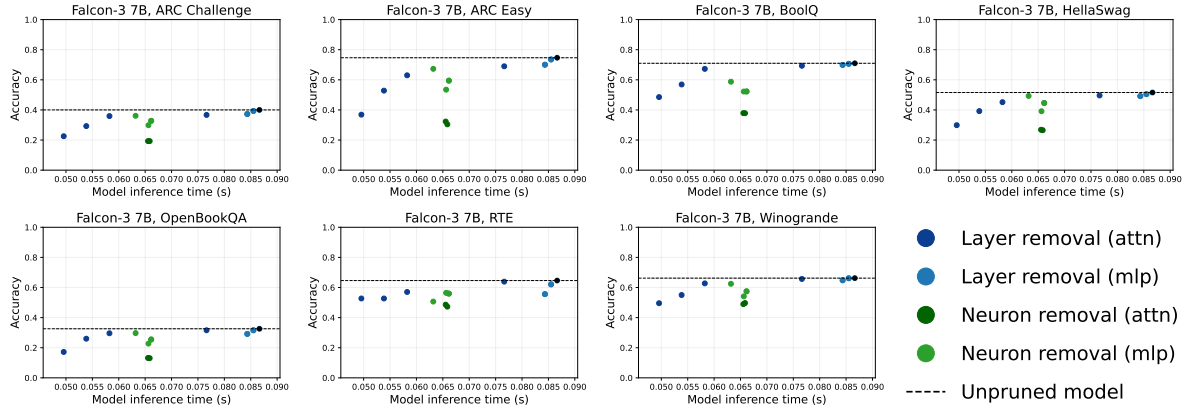


Figure 47: Inference time of pruned models against model accuracy (y-axis). Coloured dots represent different pruning methods, while the black line indicates the unpruned model accuracy. The black dot marks the inference time required by the unpruned model.

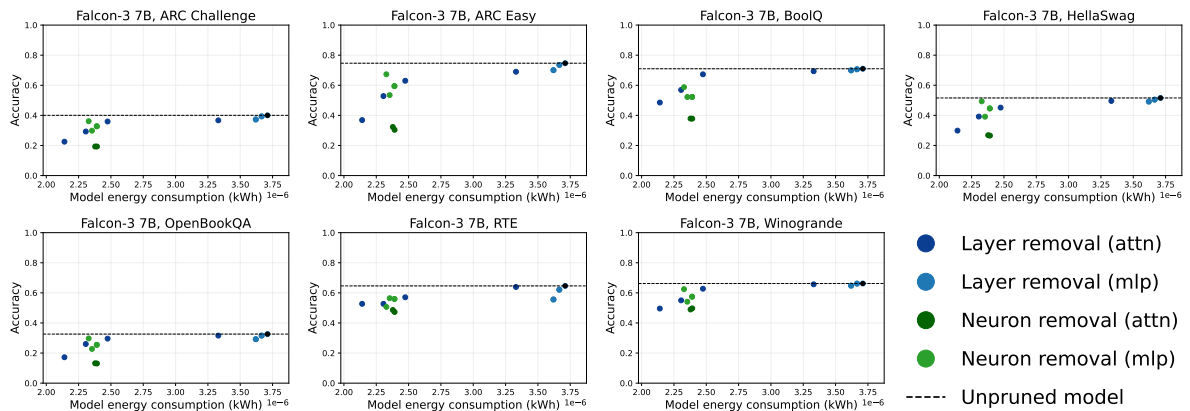


Figure 48: Energy consumption of pruned models against model accuracy (y-axis). Coloured dots represent different pruning methods, while the black line indicates the unpruned model accuracy. The black dot marks the energy consumption of the unpruned model.

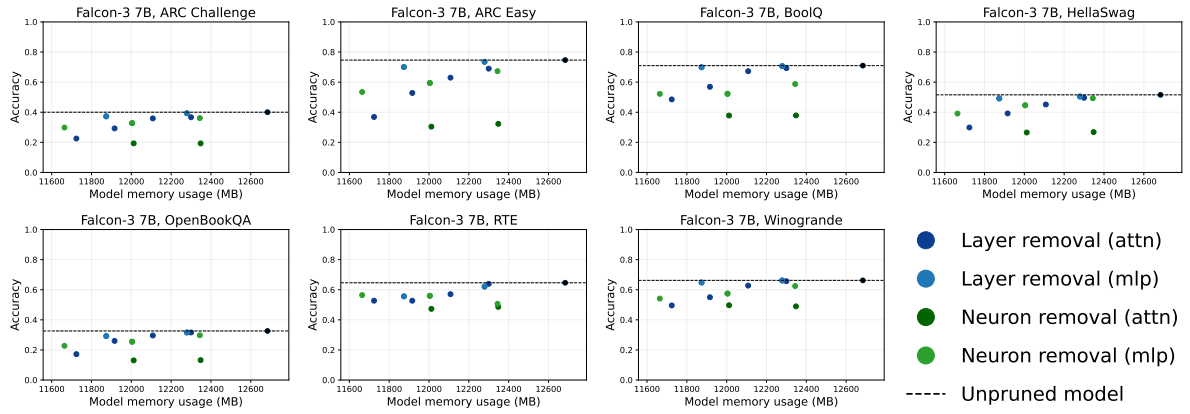


Figure 49: Memory requirements of pruned models against model accuracy (y-axis). Coloured dots represent different pruning methods, while the black line indicates the unpruned model accuracy. The black dot marks the memory requirements of the unpruned model.

Pruning Algorithm	Threshold	Accuracy	Accuracy Drop	Energy	Time	Memory
Unpruned	-	0.609	-	0.222	0.06	3975
Layer removal (attn)	8	0.589	0.02	0.189	0.05	3725
Layer removal (attn)	12	0.551	0.058	0.172	0.046	3600
Layer removal (attn)	16	0.432	0.177	0.155	0.041	3475
Layer removal (attn)	20	0.385	0.224	0.138	0.037	3350
Layer removal (mlp)	2	0.602	0.007	0.216	0.058	3773
Layer removal (mlp)	3	0.590	0.02	0.213	0.057	3671
Layer removal (mlp)	5	0.570	0.039	0.206	0.055	3469
Layer removal (mlp)	6	0.555	0.054	0.203	0.054	3368
Neuron removal (attn)	25	0.387	0.222	0.204	0.058	3738
Neuron removal (attn)	50	0.326	0.283	0.278	0.075	3504
Neuron removal (attn)	62	0.323	0.287	0.239	0.068	3393
Neuron removal (attn)	75	0.323	0.286	0.25	0.072	3271
Neuron removal (mlp)	12	0.401	0.208	0.208	0.059	3534
Neuron removal (mlp)	15	0.363	0.246	0.209	0.06	3451
Neuron removal (mlp)	18	0.352	0.257	0.211	0.061	3365
Neuron removal (mlp)	25	0.334	0.276	0.205	0.058	3165
Wanda	43	0.542	0.067	-	-	-
Wanda	46	0.497	0.112	-	-	-
Wanda	48	0.463	0.146	-	-	-
Wanda	50	0.430	0.179	-	-	-
Ria	43	0.446	0.163	-	-	-
Ria	46	0.386	0.223	-	-	-
Ria	48	0.362	0.247	-	-	-
Ria	50	0.351	0.258	-	-	-

Table 9: Aggregated BitNet 2B results for each pruning configuration. Results are averaged across datasets and, for pruned configurations, across random seeds. No information is reported for Energy, Time, and Memory for Wanda and Ria as unstructured pruning algorithms do not lead to direct reductions in those measures. Energy is reported in Kwh and exponentiated by 1e5 to appreciate changes. Time is reported in seconds and memory is reported in MB.

Pruning Algorithm	Threshold	Accuracy	Accuracy Drop	Energy	Time	Memory
Unpruned	-	0.499	-	0.181	0.049	3024
Layer removal (attn)	8	0.452	0.047	0.148	0.04	2880
Layer removal (attn)	12	0.377	0.122	0.131	0.035	2808
Layer removal (attn)	16	0.348	0.151	0.113	0.03	2736
Layer removal (attn)	20	0.329	0.17	0.096	0.026	2664
Layer removal (mlp)	1	0.476	0.023	0.178	0.048	2916
Layer removal (mlp)	2	0.482	0.017	0.175	0.047	2808
Layer removal (mlp)	3	0.462	0.037	0.172	0.046	2700
Neuron removal (attn)	25	0.366	0.133	0.2	0.057	2916
Neuron removal (attn)	50	0.333	0.166	0.209	0.06	2808
Neuron removal (attn)	75	0.326	0.173	0.21	0.06	2700
Neuron removal (mlp)	6	0.454	0.045	0.191	0.056	2869
Neuron removal (mlp)	9	0.430	0.069	0.192	0.057	2791
Neuron removal (mlp)	12	0.404	0.095	0.196	0.056	2713
Neuron removal (mlp)	15	0.387	0.112	0.198	0.057	2635
Wanda	43	0.448	0.051	-	-	-
Wanda	46	0.412	0.087	-	-	-
Wanda	48	0.380	0.119	-	-	-
Ria	43	0.451	0.048	-	-	-
Ria	46	0.413	0.086	-	-	-
Ria	48	0.381	0.118	-	-	-

Table 10: Aggregated Falcon-E 1B results for each pruning configuration. Results are averaged across datasets and, for pruned configurations, across random seeds. No information is reported for Energy, Time, and Memory for Wanda and Ria as unstructured pruning algorithms do not lead to direct reductions in those measures. Energy is reported in Kwh and exponentiated by $1e5$ to appreciate changes. Time is reported in seconds and memory is reported in MB.

Pruning Algorithm	Threshold	Accuracy	Accuracy Drop	Energy	Time	Memory
Unpruned	-	0.535	-	0.241	0.065	5568
Layer removal (attn)	8	0.501	0.034	0.207	0.056	5424
Layer removal (attn)	12	0.498	0.037	0.19	0.051	5352
Layer removal (attn)	16	0.453	0.082	0.173	0.046	5280
Layer removal (attn)	20	0.414	0.121	0.156	0.042	5208
Layer removal (mlp)	1	0.530	0.005	0.238	0.064	5412
Layer removal (mlp)	2	0.530	0.005	0.235	0.063	5256
Neuron removal (attn)	25	0.373	0.162	0.259	0.076	5424
Neuron removal (attn)	50	0.335	0.2	0.265	0.075	5280
Neuron removal (mlp)	3	0.523	0.012	0.277	0.074	5419
Neuron removal (mlp)	6	0.503	0.032	0.258	0.076	5269
Wanda	37	0.518	0.017	-	-	-
Wanda	40	0.498	0.037	-	-	-
Wanda	41	0.490	0.045	-	-	-
Wanda	43	0.470	0.065	-	-	-
Ria	37	0.520	0.015	-	-	-
Ria	40	0.503	0.032	-	-	-
Ria	41	0.494	0.041	-	-	-
Ria	43	0.469	0.066	-	-	-

Table 11: Aggregated Falcon-E 3B results for each pruning configuration. Results are averaged across datasets and, for pruned configurations, across random seeds. No information is reported for Energy, Time, and Memory for Wanda and Ria as unstructured pruning algorithms do not lead to direct reductions in those measures. Energy is reported in Kwh and exponentiated by $1e5$ to appreciate changes. Time is reported in seconds and memory is reported in MB.

Pruning Algorithm	Threshold	Accuracy	Accuracy Drop	Energy	Time	Memory
Unpruned	-	0.543	-	0.241	0.062	13312
Layer removal (attn)	8	0.532	0.011	0.208	0.054	12672
Layer removal (attn)	12	0.510	0.033	0.191	0.049	12352
Layer removal (attn)	16	0.456	0.087	0.174	0.045	12032
Layer removal (attn)	20	0.393	0.15	0.157	0.041	11712
Layer removal (mlp)	2	0.532	0.01	0.234	0.061	12640
Layer removal (mlp)	3	0.515	0.028	0.231	0.06	12304
Layer removal (mlp)	4	0.506	0.036	0.228	0.059	11968
Layer removal (mlp)	5	0.474	0.068	0.224	0.058	11632
Neuron removal (attn)	25	0.320	0.223	0.294	0.081	12672
Neuron removal (attn)	50	0.323	0.22	0.289	0.08	12032
Neuron removal (attn)	62	0.321	0.222	0.255	0.07	11725
Neuron removal (mlp)	6	0.356	0.187	0.252	0.071	12668
Neuron removal (mlp)	9	0.343	0.2	0.228	0.066	12345
Neuron removal (mlp)	12	0.340	0.203	0.226	0.065	12022
Neuron removal (mlp)	15	0.335	0.208	0.228	0.066	11700
Wanda	37	0.493	0.05	-	-	-
Wanda	40	0.456	0.087	-	-	-
Wanda	43	0.398	0.145	-	-	-
Ria	37	0.489	0.054	-	-	-
Ria	40	0.449	0.094	-	-	-
Ria	43	0.390	0.153	-	-	-

Table 12: Aggregated Llama3 8B results for each pruning configuration. Results are averaged across datasets and, for pruned configurations, across random seeds. No information is reported for Energy, Time, and Memory for Wanda and Ria as unstructured pruning algorithms do not lead to direct reductions in those measures. Energy is reported in Kwh and exponentiated by $1e5$ to appreciate changes. Time is reported in seconds and memory is reported in MB.

Pruning Algorithm	Threshold	Accuracy	Accuracy Drop	Energy	Time	Memory
Unpruned	-	0.525	-	0.16	0.042	4620
Layer removal (attn)	8	0.428	0.098	0.127	0.033	4236
Layer removal (attn)	12	0.381	0.145	0.11	0.029	4044
Layer removal (attn)	16	0.332	0.193	0.094	0.025	3852
Layer removal (attn)	20	0.317	0.208	0.078	0.02	3660
Layer removal (mlp)	2	0.498	0.027	0.153	0.04	4296
Layer removal (mlp)	4	0.475	0.05	0.147	0.039	3972
Layer removal (mlp)	5	0.468	0.057	0.144	0.038	3810
Layer removal (mlp)	6	0.456	0.069	0.141	0.037	3648
Neuron removal (attn)	37	0.334	0.191	0.178	0.05	4230
Neuron removal (attn)	62	0.329	0.196	0.178	0.05	3966
Neuron removal (attn)	75	0.328	0.198	0.182	0.053	3828
Neuron removal (attn)	88	0.359	0.166	0.181	0.051	3691
Neuron removal (mlp)	12	0.399	0.126	0.178	0.051	4193
Neuron removal (mlp)	18	0.353	0.172	0.177	0.05	3979
Neuron removal (mlp)	21	0.341	0.184	0.177	0.05	3872
Neuron removal (mlp)	28	0.330	0.196	0.175	0.05	3622
Wanda	34	0.490	0.035	-	-	-
Wanda	37	0.460	0.065	-	-	-
Wanda	40	0.439	0.086	-	-	-
Wanda	43	0.409	0.116	-	-	-
Ria	34	0.491	0.034	-	-	-
Ria	37	0.473	0.052	-	-	-
Ria	40	0.443	0.082	-	-	-
Ria	43	0.408	0.117	-	-	-

Table 13: Aggregated Falcon-3 3B results for each pruning configuration. Results are averaged across datasets and, for pruned configurations, across random seeds. No information is reported for Energy, Time, and Memory for Wanda and Ria as unstructured pruning algorithms do not lead to direct reductions in those measures. Energy is reported in Kwh and exponentiated by $1e5$ to appreciate changes. Time is reported in seconds and memory is reported in MB.

Pruning Algorithm	Threshold	Accuracy	Accuracy Drop	Energy	Time	Memory
Unpruned	-	0.572	-	0.371	0.087	12684
Layer removal (attn)	8	0.551	0.021	0.333	0.077	12300
Layer removal (attn)	12	0.515	0.057	0.247	0.058	12108
Layer removal (attn)	16	0.446	0.127	0.231	0.054	11916
Layer removal (attn)	20	0.368	0.205	0.214	0.05	11724
Layer removal (mlp)	1	0.563	0.01	0.367	0.086	12279
Layer removal (mlp)	2	0.537	0.035	0.362	0.084	11874
Neuron removal (attn)	25	0.324	0.248	0.238	0.066	12348
Neuron removal (attn)	50	0.320	0.252	0.239	0.066	12012
Neuron removal (mlp)	3	0.506	0.066	0.233	0.063	12344
Neuron removal (mlp)	6	0.469	0.104	0.239	0.066	12004
Neuron removal (mlp)	9	0.440	0.132	0.235	0.066	11664
Wanda	31	0.563	0.01	-	-	-
Wanda	34	0.547	0.026	-	-	-
Wanda	35	0.533	0.039	-	-	-
Ria	31	0.565	0.008	-	-	-
Ria	34	0.544	0.028	-	-	-
Ria	35	0.534	0.039	-	-	-

Table 14: Aggregated Falcon-3 7B results for each pruning configuration. Results are averaged across datasets and, for pruned configurations, across random seeds. No information is reported for Energy, Time, and Memory for Wanda and Ria as unstructured pruning algorithms do not lead to direct reductions in those measures. Energy is reported in Kwh and exponentiated by $1e5$ to appreciate changes. Time is reported in seconds and memory is reported in MB.