MAMA PRUNING: IMPROVED METHODS FOR MODEL PRUNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Model pruning is a performance optimization technique for large language and vision models. However, existing pruning methods often lead to significant performance degradation or require extensive retraining and fine-tuning. This technique aims to identify and remove neurons, connections unlikely leading to the contribution during the machine generation phase. Our goal is to obtain a much smaller and faster foundational model that can quickly generate content almost as good as those of the unpruned models. We propose MAMA Pruning (short for Movement and Magnitude Analysis), an improved pruning method that effectively reduces model size and network computational complexity while maintaining performance comparable to the original unpruned model even at extreme pruned levels. The improved method is based on weights, bias, activations and proposed novel pruning indicators. Empirical results show that our method outperforms and be comparable to state-of-the-art methods across various pruning levels. All our code, models, dataset, and demo are publicly available.

1 INTRODUCTION

Large language and vision models(Brown et al., 2020a; OpenAI, 2023) face significant computational challenges due to massive model sizes and the high query loads that these systems need to support. These models, along with related large-scale production systems, are responsible for processing and integrating vast amounts of data—including web pages, videos, and multimodal content—into underlying network architectures such as Transformers and Diffusion models. One crucial cost factor is the query processing per user, which must scale with both data size and query load. As a result, large foundational models devote substantial hardware and energy resources to this kind of generation task. There has been extensive research on improving query processing performance, including work on various caching techniques, retrieval information systems, and high-performance knowledge representation. To address these challenges, a significant number of optimization techniques, commonly referred to as model pruning, have emerged to enhance the efficiency and effectiveness of AI-Generated Content (AIGC) generation processes. In this paper, we propose an improved model pruning algorithm based on novel indicators derived from an in-depth analysis of weights, biases, and activations. Our method significantly enhances the performance and efficiency of large language and vision models. We demonstrate through extensive experiments that our approach outperforms existing state-of-the-art pruning techniques across various metrics.

In this paper, we focus on model pruning, an optimization technique aimed at enhancing neural network efficiency. Our approach involves analyzing learning representations, network architectures, and performance metrics to identify neurons and connections that significantly contribute to the model's output in response to user input. By systematically removing neurons unlikely to improve the network's performance, we reduce the model's size and complexity. The resulting pruned neural network maintains nearly the same output quality as the original unpruned model while requiring substantially less CPU power, memory, and

GPU resources. This optimization leads to faster query processing and more efficient deployment of neural networks in resource-limited environments.

To motivate the problem, consider a state-of-the-art language model with around 175 billion parameters, deployed across various services and processing an estimated 100 million user queries daily. Interestingly, during typical inference tasks, only about 20-30% of the parameters are activated. This means that a majority—approximately 70-80% of the model's weights—remain unused for each specific input. For example, each query might engage roughly 52 billion parameters on average, leaving about 123 billion parameters inactive.

This imbalance raises an important question: Can we reduce the computational load by pruning these inactive weights without sacrificing the model's overall performance? Pruning 70-80% of the model's parameters could theoretically reduce the active parameter count significantly, lowering the operational costs of running the service. However, aggressive pruning might lead to unacceptable losses in quality, causing user-facing services to suffer noticeable degradation in response accuracy or fluency. The challenge, therefore, lies in developing a pruning method that effectively reduces the parameter count while maintaining high accuracy and minimizing computational overhead. Given the scale of such large models, even a 1-2% drop in accuracy could result in millions of queries per day yielding suboptimal results. This example underscores the urgent need for improved pruning methods that can balance sparsity and efficiency without compromising the quality of large-scale models.

Previous work on model pruning for large language and vision models has primarily focused on approaches such as retaining layers that exceed a global impact threshold or keeping high-scoring neurons within each layer. For details, we refer to (Han et al., 2015b; Frantar & Alistarh, 2023b; Sun et al., 2024b). While these methods have yielded promising results at certain pruning levels, there remains significant room for optimization. The goal of this paper is to build on existing work by developing a methodology that combines multiple indicators to achieve a better balance between neural network size and generation quality, as measured by standard retrieval evaluation metrics. In our approach, we consider pruning as a prediction problem within a feature-rich environment, aiming to determine which neurons, weights, or layers to retain.

Table 1 serves as a motivating example on the end-to-end effectiveness of pruned model (OPT-1.3B) for downstream task such as text generation. Below are the key observations:

1. Perplexity and Pruning Level: As the pruning level increases (from 0.00 to 0.99), the perplexity of the generated text also increases significantly. This indicates that the model's ability to generate coherent and meaningful text deteriorates as more parameters are pruned.

2. Text Generation Quality: The generated text samples clearly demonstrate the impact of pruning. At lower pruning levels, the model generates relatively fluent and coherent sentences. However, as pruning progresses, the generated text becomes increasingly repetitive, fragmented, and nonsensical.

3. Trade-off between Efficiency and Quality: Pruning can lead to significant reductions in model size and computational requirements, but it comes at the cost of reduced performance. The table illustrates this trade-off, as higher pruning levels result in smaller models but poorer text generation quality.

The remainder of this paper is organized as follows. In Section 2, we provide background information on learning representations, neural networks, and related pruning techniques. We summarize our key contributions in Section 3, highlighting the novelty and significance of our approach. In Section 4, we provide a comprehensive explanation of our proposed methodology and developed algorithms. We present our experimental results in Section 5, along with implementation details and performance analysis. Finally, in Section 6, we provide concluding remarks, summarizing our main findings and suggesting potential directions for future research.

Table 1: A motivating example for the End-to-end effectiveness of pruned model for downstream task text generation across different pruning levels.

Pruned Level	Perplexity	text generation starts with University is
0.00	5.677	University is a great place to learn about the world.
0.50	19.191	University is a great place to start a new year.
0.60	23.205	University is a great place to start.
0.70	44.246	University is a good place to get a good place to get a good place to get a good
0.80	364.304	University is a lot lot lot lot lot lot lot lot lot lot lot lot lot lot lot lot
0.90	3772.829	University is.
0.95	8892.167	University is is is is is is is is is is is is is is is is is is
0.99	22548.809	University is is is is is is is is is is is is,,,,,,

2 Background and Related Work

In this section, we provide background on neural network architectures, emphasizing components pertinent to model pruning. We discuss the role of human input and machine-generated data in training neural networks and explore quantization and compression techniques used to optimize model performance. We then review previous work related to model pruning in the context of large-scale systems, highlighting advances and identifying areas for further research. For additional details on general neural network architectures, we refer readers to (Goodfellow et al., 2016; Russell & Norvig, 2020).

2.1 Background

2.1.1 Neural Network Architectures

Neural network architectures have undergone significant evolution over the past decades, moving from simple multi-layer perceptrons (MLPs) to highly sophisticated models such as convolutional neural networks (CNNs), recurrent neural networks (RNNs), and transformers. These architectures are designed to handle different types of input data and tasks. This enables deep learning models to tackle complex challenges across various domains, including natural language processing (NLP), computer vision, and time-series prediction.

CNNs were pivotal in advancing the field of computer vision by introducing specialized layers that focus on spatial hierarchies in data. AlexNet(Krizhevsky et al., 2012), for instance, brought CNNs to prominence by demonstrating their superiority in tasks like image classification. These models use convolutional layers to capture local features from the input, pooling layers for dimensionality reduction, and fully connected layers to make final predictions. More advanced versions like ResNet(He et al., 2016) introduced the concept of residual learning, enabling the training of ultra-deep networks by allowing information to bypass layers through skip connections.

In contrast, RNNs and their variants, such as long short-term memory (LSTM) networks (Hochreiter & Schmidhuber, 1997), have excelled in sequential data processing. These models capture temporal dependencies, making them suitable for tasks like speech recognition and machine translation. However, their limitations in handling long-range dependencies led to the rise of attention mechanisms and transformers.

Transformers(Vaswani et al., 2017) revolutionized the field by discarding the need for recurrence, relying instead on self-attention mechanisms to capture relationships between tokens, regardless of their distance in

the sequence. This architectural shift led to models such as BERT (Bidirectional Encoder Representations from Transformers)(Devlin et al., 2019) and GPT (Generative Pre-trained Transformer) (Radford et al., 2018), which set new benchmarks across a range of NLP tasks.

As neural networks continue to grow in size and complexity, designing architectures that balance performance with efficiency has become a critical challenge. Techniques like depth-wise separable convolutions used in MobileNet(Howard et al., 2017) and efficient transformer variations such as DistilBERT(Sanh et al., 2019) aim to reduce the computational burden without sacrificing accuracy. These advancements pave the way for large-scale neural networks that are both performant and scalable.

2.1.2 HUMAN INPUT & MACHINE GENERATION

The interaction between human input and machine generation is a rapidly evolving area with significant implications for artificial intelligence (AI) systems designed to augment or replace human decision-making. Human input—whether in the form of annotations, feedback, or direct manipulation of model outputs—plays a critical role in training and refining AI models. This interaction is key in supervised learning, where labeled datasets curated by humans guide the learning processes of models (LeCun et al., 2015).

In domains like content generation, machine learning models utilize human inputs to produce creative outputs such as text, images, or music. Generative Adversarial Networks (GANs)(Goodfellow et al., 2014) have demonstrated remarkable success in generating high-quality images by leveraging an adversarial relationship between a generator and a discriminator. The generator creates new images, while the discriminator evaluates their authenticity compared to real images, driving iterative improvements in generation quality.

Similarly, transformer-based models like GPT-3(Brown et al., 2020b) have showcased the power of machine generation in natural language tasks. GPT-3 can generate human-like text ranging from simple responses to complex narratives or technical content. The flexibility of transformer architectures allows them to be fine-tuned for diverse applications, from automated customer support to creative writing.

However, machine generation presents challenges, particularly in controlling and guiding model outputs to align with human intentions. While human input can define high-level goals or constraints, fine-tuning models often requires extensive iterations and feedback loops. In creative fields, for instance, human designers may provide input to a generative model, but the output might still require refinement to meet aesthetic or functional criteria. Techniques like Reinforcement Learning with Human Feedback (RLHF)(Christiano et al., 2017; Ouyang et al., 2022) address this issue by using reward mechanisms to guide models toward more desirable outputs based on human evaluations.

The relationship between human input and machine generation continues to evolve, driving innovations in AI systems that not only automate tasks but also enhance creative and decision-making processes. Understanding this interplay is essential for developing efficient model pruning techniques that preserve the quality of machine-generated content while optimizing computational resources.

2.1.3 MODEL QUANTIZATION & COMPRESSION

Model quantization and compression are essential techniques aimed at reducing the computational and memory footprints of deep learning models, making them more suitable for deployment in resource-constrained environments such as mobile devices and embedded systems. These techniques allow models to maintain high accuracy while being smaller, faster, and more energy-efficient, which is critical in real-time applications and edge computing.

Quantization refers to the process of reducing the precision of the weights and activations in a neural network (Krishnamoorthi, 2018). Instead of using 32-bit floating-point numbers, which is the default precision in many models, quantized models use lower precision formats, such as 16-bit or 8-bit integers. Quantization-

Aware Training (QAT) incorporates quantization into the training process itself, allowing the model to adjust to lower precision during learning (Jacob et al., 2018). Techniques like Post-Training Quantization (PTQ) enable quantization after training, making it easier to deploy pre-trained models with minimal loss in accuracy (Simonyan & Zisserman, 2015).

Model compression involves various techniques that aim to reduce the overall size of a neural network. This can be achieved through weight pruning, where less important weights are removed (Han et al., 2016), or through knowledge distillation, where a smaller "student" model is trained to mimic the behavior of a larger "teacher" model (Hinton et al., 2015). Model compression techniques often complement quantization, as both aim to make models more efficient without substantial sacrifices in performance.

Recent advances in model quantization and compression have introduced more sophisticated strategies that exploit the trade-offs between model size and accuracy. For example, mixed-precision quantization allows certain layers of the model to retain higher precision for critical tasks while quantizing less critical layers more aggressively. Compression-aware training techniques further optimize the model architecture, ensuring that compressed models remain robust during inference (He et al., 2018).

These techniques are particularly important in real-world scenarios where latency, memory usage, and energy efficiency are crucial. From autonomous driving systems to personal digital assistants, quantization and compression enable AI to function effectively in environments where computational resources are limited.

2.1.4 Model Pruning & Indicators Discovery

Model pruning is a widely used technique aimed at reducing the complexity of deep neural networks by removing unnecessary parameters, resulting in more efficient models in terms of both size and inference speed (Han et al., 2015a). Pruning can target weights, neurons, channels, or even entire layers within the network. A crucial aspect of the pruning process is the discovery of reliable indicators that determine which parts of the network can be safely removed without significantly affecting model performance.

Traditional pruning techniques, such as magnitude-based pruning, operate under the assumption that smaller-magnitude weights contribute less to the overall model output and can be pruned with minimal loss of accuracy (Han et al., 2015a). While this method is straightforward and effective in some cases, it overlooks the complex interdependencies between weights and the hierarchical structure of neural networks, potentially leading to suboptimal pruning choices.

In contrast, structured pruning techniques operate at a higher level by removing entire channels, filters, or layers (Li et al., 2017). This approach offers more systematic reductions in model size and is particularly advantageous for hardware implementations, as it leads to predictable reductions in computational load. However, identifying reliable indicators for structured pruning—such as the importance or sensitivity of specific channels or layers—remains a challenge.

Layer-wise pruning approaches introduce the concept of adaptive sparsity, where different layers are pruned at varying rates based on their sensitivity to performance degradation (Zhu & Gupta, 2018). Techniques like variational dropout (Molchanov et al., 2017a) and dynamic sparse training enhance pruning by dynamically adjusting sparsity during training. These methods enable the model to discover optimal pruning strategies that balance performance and efficiency.

Recent advancements in pruning have focused on identifying novel indicators for pruning decisions. Gradient-based methods, such as SNIP (Wang et al., 2024), evaluate the importance of weights based on the magnitude of their gradients during backpropagation. Additionally, the Lottery Ticket Hypothesis (Frankle & Carbin, 2019) suggests that sparse sub-networks exist within dense models that can be trained from scratch to achieve comparable performance. This has opened new avenues for efficient pruning by emphasizing the selection of these "winning tickets."

The discovery of reliable pruning indicators plays a pivotal role in advancing model pruning techniques. By accurately identifying which parameters can be removed without harming performance, it is possible to develop more compact models that maintain high accuracy across various tasks. In our work, we aim to build upon these methods by developing a novel approach that combines multiple indicators to guide pruning decisions more effectively, leading to improved efficiency without compromising model performance.

2.2 Related Work

2.2.1 The Magnitude Pruning Algorithm

The magnitude pruning algorithm (Han et al., 2015a) is one of the simplest and most widely used methods for reducing the size of neural networks. It operates by pruning weights based on their absolute magnitude: weights with smaller absolute values are considered less critical to the model's performance and are pruned, while larger weights are retained. The typical approach involves setting a global threshold—determined by the desired sparsity ratio—below which weights are set to zero. Magnitude pruning is unstructured, meaning it can prune individual weights from any part of the model, leading to irregular sparsity patterns.

Advantages: Magnitude pruning is computationally inexpensive and fast. Evaluating weights based on their magnitude requires minimal computational overhead and can be applied to any layer of the network without complex calculations or additional data. This method is also versatile, as it can be used across different types of neural network architectures.

Limitations: Despite its simplicity, magnitude pruning has inherent drawbacks in terms of accuracy degradation and the resulting sparsity patterns. Since the algorithm considers only the magnitude of individual weights, it may inadvertently remove important connections, potentially leading to significant loss in model accuracy. Additionally, the unstructured nature of the resulting sparsity leads to irregular memory access patterns, making it challenging to achieve computational speed-ups on standard hardware due to poor cache utilization and inefficient parallelization.

To address these limitations, researchers have explored more structured versions of the algorithm that target specific parts of the network—such as entire filters, channels, or blocks—rather than individual weights (Li et al., 2017). Structured pruning results in more regular sparsity patterns that are better suited for hardware acceleration. Moreover, combining magnitude pruning with retraining or fine-tuning steps can help mitigate accuracy loss by allowing the model to adjust to the pruned architecture (Han et al., 2015a).

2.2.2 The SparseGPT Pruning Algorithm

The SparseGPT pruning algorithm (Frantar & Alistarh, 2023a) is an advanced method specifically designed to handle large language models like GPT. It employs a gradient-based approach, utilizing gradient information during pruning to identify and remove less important connections in the model. By calculating the gradients of the loss function with respect to network weights, SparseGPT assesses the significance of each weight, allowing for more informed pruning decisions.

SparseGPT adopts an iterative pruning strategy, gradually increasing sparsity while minimizing accuracy loss at each step. In each iteration, a subset of weights with the least impact on the model's performance is pruned based on gradient evaluations. This progressive approach enables the algorithm to carefully balance the trade-off between model size and accuracy.

An essential feature of SparseGPT is its encouragement of block-sparse structures, where entire blocks or groups of weights are pruned together. Block sparsity results in regular sparsity patterns that are more hardware-friendly and can be efficiently leveraged by modern high-performance computing architectures. This makes SparseGPT particularly suitable for hardware acceleration and allows it to take advantage of libraries optimized for such sparsity patterns.

SparseGPT achieves high levels of sparsity with minimal accuracy degradation, making it ideal for compressing large-scale models. Its ability to maintain performance while significantly reducing model size addresses the challenges associated with deploying large language models in resource-constrained environments.

Limitations: Despite its advantages, SparseGPT has limitations related to computational expense and complexity. The iterative nature of the algorithm, coupled with the necessity to compute gradients during the pruning process, increases computational overhead compared to simpler methods like magnitude pruning. SparseGPT requires careful management of hyperparameters—such as pruning rates and the number of iterations—and multiple passes through the data to achieve optimal results. This complexity can make implementation and integration more challenging.

Potential Optimizations: Optimizing SparseGPT involves reducing computational overhead during the iterative pruning process. Techniques such as approximating gradient calculations using estimations or using fewer iterations while striving to preserve sparsity patterns could enhance efficiency. Incorporating layer-wise pruning strategies may also refine performance by allowing for differential pruning rates across layers based on their importance or sensitivity.

2.2.3 THE WANDA PRUNING ALGORITHM

The WANDA (Weights and Activations) pruning algorithm(Sun et al., 2024a) introduces an importance-aware approach to pruning by considering both weight magnitudes and activation statistics. By integrating activation information, WANDA makes more informed pruning decisions based on the relative importance of weights to the network's output.

The algorithm begins with a calibration phase, where the model processes a small calibration dataset to collect activation data. This data is used to normalize the weight magnitudes within each row of the weight matrices, effectively weighting the importance of each connection based on both its inherent strength and its activation impact. This normalization allows WANDA to identify and retain weights that contribute most significantly to the model's performance.

WANDA can be applied in both unstructured and structured pruning settings. In unstructured pruning, individual weights are pruned, leading to irregular sparsity patterns. In structured pruning, entire filters, channels, or neurons are pruned, resulting in regular sparsity patterns that are more conducive to hardware acceleration. This flexibility allows WANDA to be utilized across various models and hardware configurations.

By considering activation statistics alongside weight magnitudes, WANDA achieves better accuracy-sparsity trade-offs compared to traditional magnitude-based pruning. Empirical results have demonstrated that WANDA maintains higher model accuracy at comparable levels of sparsity(Sun et al., 2024a).

Potential Optimizations: To further enhance WANDA's efficiency, the calibration phase could be streamlined by employing smaller datasets or more efficient methods for collecting activation statistics. Additionally, combining WANDA with other pruning techniques, such as gradient-based methods, may yield even better accuracy and sparsity outcomes. For example, integrating gradient information could help in identifying weights that are critical for minimizing the loss function.

2.2.4 PRUNING IN GENERAL

Beyond the aforementioned methods, several other pruning algorithms have been proposed in the literature, each with unique approaches and trade-offs. Structured pruning, for example, targets entire neurons, channels, or filters rather than individual weights, offering more predictable reductions in computation.

Techniques like Taylor-based pruning(Molchanov et al., 2017b) use first-order approximations to evaluate the impact of pruning each weight or filter, allowing for more fine-tuned control over sparsity.

2.2.5 COMPARISON TO OUR WORK

MAMA (Movement And Magnitude Analysis) pruning is fundamentally different from existing pruning methods in its approach to identifying and preserving important neural connections. Unlike magnitude-based pruning, which simply removes weights below a certain threshold, or methods like SparseGPT that use gradient information, MAMA employs a novel three-step process that considers both the magnitude and the dynamic behavior of weights during training. We will describe our proposed methods in detail in the following sections.

3 OUR CONTRIBUTIONS

In this paper, we study large model pruning that attempt to achieve a good trade-off between model size and performance. Our main contributions are as follows:

1. We describe an approach called MAMA that can perform much better over previous SOTAs;

2. We design a unified automatic evaluation framework for pruning technique evaluation;

3. We perform an comprehensive experimental evaluation over different datasets, models and metrics;

4. We compare human designed algorithms to AIGC generated algorithms by industry leading models, concluding the strengths and weaknesses from both sides in complex tasks such as ”code generation for optimization”.

4 OUR PROPOSED METHOD: MAMA PRUNING

MAMA Pruning is grounded in a systematic approach that identifies and preserves dynamically significant weights by redistributing less important weights to more critical connections within the network. This ensures the maintenance of overall information flow and network adaptability, even under high pruning ratios. The methodology encompasses three core steps:

1. **Identify the pruned weights**. The first step involves identifying weights eligible for pruning based on both their magnitude and dynamic behavior during training. This dual analysis ensures that weights contributing minimally to the network's performance are targeted for pruning.

2. **Redistribute weights to related neurons**. Following the identification of prunable weights, MAMA Pruning undertakes a redistribution phase, wherein the values of unimportant weights are ”moved” to more significant connections within the same layer. This strategic redistribution ensures the preservation of the network's information flow and compensates for the pruned weights.

3. **Execute the pruning**. The final step involves pruning the identified weights to achieve the desired sparsity level within the model. This step actualizes the reduction in model parameters, preparing the network for deployment or further optimization.

5 EXPERIMENTAL RESULTS

Table 2 and Figure 2 presents the effectiveness of the Weights as major pruning indicator measured by perplexity. Below are the key observations:

1. Low Pruning Levels (0.01 - 0.20)

- ”Prune by Weights” produces very low perplexity values at these levels, but it’s unavailable (”NA”) at the 0.01 and 0.05 pruning levels. This could mean the method is ineffective or unapplicable at extremely low pruning levels.
- ”Prune by -Weights” shows relatively higher perplexity (e.g., 24377.635 at 0.01), indicating that this method has a larger impact on performance early on, potentially making the model less effective in terms of perplexity.

2. Medium Pruning Levels (0.30 - 0.50)

- At these levels, ”Prune by Weights” continues to have low perplexity values (e.g., 6.669 at 0.30, 17.285 at 0.50), suggesting it maintains good performance even as pruning increases.
- ”Prune by -Weights” perplexity remains significantly higher (e.g., 335747.406 at 0.30, 227413.484 at 0.50), indicating a larger negative impact on model performance.

3. Higher Pruning Levels (0.60 - 0.80)

- At 0.60, both methods show a noticeable increase in perplexity, but ”Prune by Weights” sees a much steeper rise (559.987 compared to 185086.078 for ”Prune by -Weights”). This might indicate that ”Prune by Weights” starts to struggle at this point, although it still outperforms the alternative in perplexity.
- By 0.80, ”Prune by Weights” perplexity has jumped to 132175.578, while ”Prune by -Weights” starts to plateau at 188488.000. This suggests that both methods show diminishing returns in terms of perplexity improvement at these high pruning levels.

4. Very High Pruning Levels (0.90 - 0.99)

- ”Prune by Weights” still yields results (e.g., 317879.250 at 0.90), though the perplexity is extremely high. This is expected, as models pruned this heavily often perform worse.
- ”Prune by -Weights” is unavailable at the highest pruning levels (0.95 and 0.99), suggesting that the method becomes inapplicable or irrelevant as the model becomes excessively sparse.
- Interestingly, ”Prune by Weights” is still operational even at 0.99, albeit with a high perplexity of 222543.047, implying that this method retains some function even in extreme pruning cases.

5. In general, the table suggests that ”Prune by Weights” is generally more stable and effective at various pruning levels, particularly if maintaining low perplexity is critical. However, the rapid increase in perplexity at higher pruning levels indicates that careful tuning is still needed to optimize performance.

Table 3 and Figure 1 presents perplexity results for pruned model (Llama-7B) from domain human experts. Below are the key observations:

1. General Trend with Pruning

- As the pruning level increases, i.e., a higher fraction of the model’s parameters are removed, the perplexity values generally increase for all methods. This trend is expected as a greater loss of parameters typically leads to a degradation in model performance.

2. High Pruning Levels (0.50 - 0.90)

- From 0.50 onward, the differences between the methods become more pronounced. SparseGPT and Wanda maintain significantly lower perplexity scores compared to Magnitude and MAMA, which exhibit a rapid increase in perplexity.

Table 2: Effectiveness of the pruning indicator Weights.

Pruned Level	Prune by Weights	Prune by -Weights
0.00	5.677	5.677
0.10	5.806	104948.891
0.20	6.020	352772.500
0.30	6.669	335747.406
0.40	8.601	260632.641
0.50	17.285	227413.484
0.60	559.987	185086.078
0.70	48414.551	273153.688
0.80	132175.578	188488.000
0.90	317879.250	185304.016

- At 0.70 pruning, for instance, SparseGPT has a perplexity of 27.214, while Magnitude and MAMA reach over 48,000 and 51,000 respectively.

3. Extreme Pruning Levels (0.95 and 0.99)

- All methods show significantly higher perplexity values, yet SparseGPT and Wanda continue to outperform Magnitude and MAMA by a large margin.
- At 0.99 pruning, SparseGPT has a perplexity of $\sim$16,869, whereas Magnitude and MAMA exhibit perplexities of $\sim$222,543 and $\sim$214,966 respectively.

4. In general

- SparseGPT and Wanda consistently outperform Magnitude and MAMA pruning methods, especially at medium to high pruning levels (0.60 and above).
- Magnitude and MAMA pruning methods exhibit significant degradation in performance (higher perplexity) at more aggressive pruning levels.
- SparseGPT is the most resilient pruning method across varying pruning levels, maintaining the lowest perplexity even at extreme pruning levels (0.90 and 0.95).
- SparseGPT (and Wanda to some extent) seem to be the preferred methods when applying aggressive pruning to large models, as they better preserve performance as indicated by perplexity.

Table 4 presents perplexity results for pruned model (Llama-7B) from domain machine expert (o1). Specifically, Figure 3 put a direct comparizon between human and machine designed pruning technique. Below are some of the key observations:

1. The human expert algorithms are much more effective at pruning the model while maintaining lower perplexity, especially as pruning becomes more aggressive. This suggests that, at least for now, human expertise outperforms machine-generated pruning strategies.

2. Please note that AIGC-Gen Alg with number 1,4,7,8,9 and 10 can NOT pass the one-pass code generation test. We will NOT report those numbers here. Alg 5 is the the exactly the same as Alg 6 so that we will ONLY report numbers for Alg 6.

Table 6 presents one-pass code generation results for the model o1. Below are the key observations:

Table 3: Perplexity on pruned model (Llama-7B) from domain human experts.

Pruned Level	Wanda	SparseGPT	Magnitude	MAMA
0.00	5.677	5.677	5.677	5.677
0.50	7.257	7.234	17.285	17.247
0.60	10.691	10.442	559.987	554.727
0.70	84.905	27.214	48414.551	51841.121
0.80	5782.432	182.463	132175.578	135494.797
0.90	19676.668	3198.101	317879.250	301472.500
0.95	28309.178	4088.413	273552.281	273629.750
0.99	108234.484	16869.203	222543.047	214966.484

Table 4: Perplexity on pruned model (Llama-7B) from domain machine expert (o1).

Pruned Level	AIGC-Gen Alg 2	AIGC-Gen Alg 3	AIGC-Gen 6
0.00	5.677	5.677	5.677
0.50	193740.406	266826.094	294350.188
0.60	110879.422	244139.875	138577.469
0.70	174815.859	453267.031	171725.375
0.80	287734.844	570346.750	186493.797
0.90	157028.844	384411.375	298142.469
0.95	90220.781	455298.469	187259.063
0.99	991519.125	206585.391	70452.703

1. The model finds it difficult to generate effective algorithms in one go in the creative application scenario of ”core algorithm generation,” despite our clear understanding of the context and the knowledge domain involved during the experiment.

2. The model did not demonstrate the exceptional capabilities of ”slow thinking,” ”outstanding mathematical logic reasoning,” and ”programming ability” that were emphasized during its promotion and dissemination, at least in our innovation application scenario, as these traits were not significantly quantifiable by scientific metrics.

3. The result suggest some important future work: (1) Investigating the reasons why the algorithm cannot be generated and successfully run in one go. (2) A horizontal comparison of the effectiveness of AIGC-generated algorithms versus those designed by human algorithm engineers. (3) Expanding the evaluation from ”code generation” by generative AI to more comprehensive assessments such as ”text generation,” ”image generation,” and ”video generation.” (4) Adding a horizontal comparison of models such as GPT-4 and Gemini Pro in vertical domains.

6 Conclusion

In this paper, we introduced MAMA pruning algorithm for model pruning in large language and vision models. Our methods estimate the likelihood of neurons producing expected results by leveraging diverse neuron features, collections, and query statistics. By comparing our approach with the other typical methods, we demonstrated significant improvements over prior work. Further, we compare human designed

Table 5: Perplexity on pruned model (Llama-13B) from domain machine expert (o1).

Pruned Level	AIGC-Gen Alg 2	AIGC-Gen Alg 3	AIGC-Gen 6
0.00	Eearly Termination	Eearly Termination	Eearly Termination
0.50	Eearly Termination	Eearly Termination	Eearly Termination
0.60	Eearly Termination	Eearly Termination	Eearly Termination
0.70	Eearly Termination	Eearly Termination	Eearly Termination
0.80	Eearly Termination	Eearly Termination	Eearly Termination
0.90	59447.305	134509.875	128350.227
0.95	60122.586	110021.789	55667.176
0.99	472500.469	100298.625	113332.750

Table 6: One-pass code generation for downstream task.

Number	Core Idea	Status	Usage Scenario
01	Gradient Sensitive Pruning	Error	Pruning
02	L1 Norm Pruning	OK	Pruning
03	Structured Pruning	OK	Pruning
04	K-means Clustering Pruning	Error	Pruning
05	Random Pruning	OK	Pruning
06	Random Pattern Pruning	OK	Pruning
07	Variational Dropout Pruning	Error	Pruning
08	Gradient based Pruning	Error	Pruning
09	Elastic Weight Consolidation Pruning	Error	Pruning
10	Dynamic Pruning with Reinforcement Learning	Error	Pruning

algorithms to AIGC generated algorithms, concluding the strengths and weaknesses for complex tasks such as optimization code generation.

For the future, we plan several extensions. This includes conducting experiments with other language models, such as GPT-4 and BERT, which may potentially achieve even better pruning performance. We also aim to further optimize our approach by exploring hybrid methods that combine our likelihood estimation with existing pruning techniques.

Additionally, we plan to study the trade-off between model size and query cost under different cost models and actual query processing algorithms. This research holds promise for enhancing the efficiency and performance of large language and vision models through more effective pruning techniques. By reducing model sizes without sacrificing performance, our work could significantly lower computational costs and enable faster inference times in practical applications.

REFERENCES

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess,

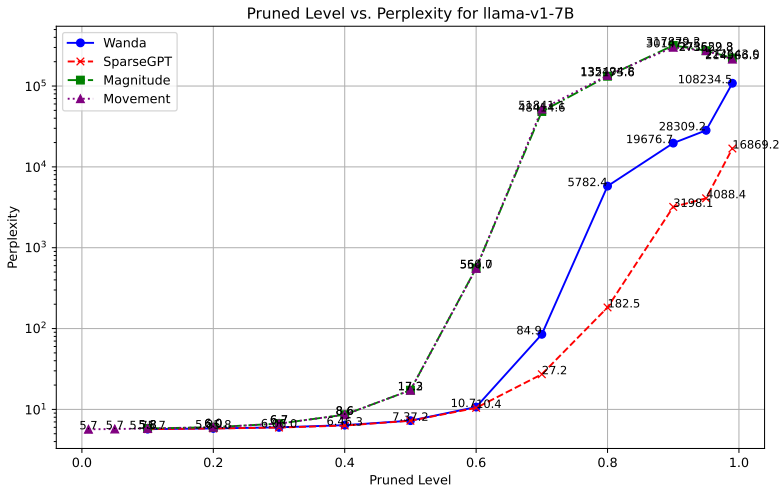

Figure 1: Pruned Level vs. Perplexity for llama-v1-7B.

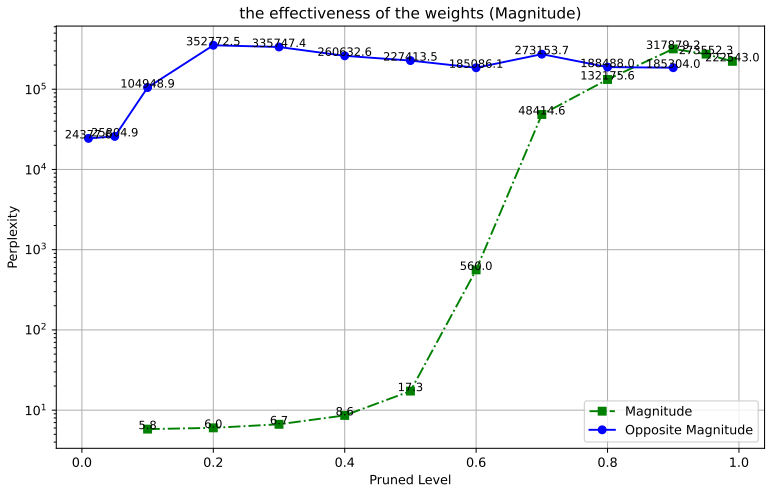

Figure 2: Effectiveness of the weights indicator.

Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020a. URL https://proceedings.neurips.cc/paper/2020/hash/

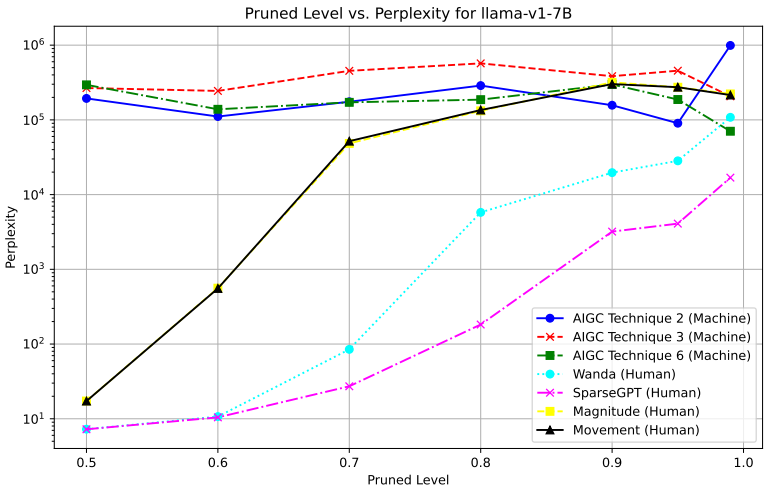

Figure 3: Performance Evaluation between Machine & Human.

1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020b. URL https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html.

Paul F. Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 4299–4307, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/d5e2c0adad503c91f91df240d0cd4e49-Abstract.html.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pp. 4171–4186. Association for Computational Linguistics, 2019. doi: 10.18653/V1/N19-1423. URL https://doi.org/10.18653/v1/n19-1423.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=rJl-b3RcF7.

Elias Frantar and Dan Alistarh. Sparsegpt: Massive language models can be accurately pruned in one-shot. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pp. 10323–10337. PMLR, 2023a. URL https://proceedings.mlr.press/v202/frantar23a.html.

Elias Frantar and Dan Alistarh. Sparsegpt: Massive language models can be accurately pruned in one-shot. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pp. 10323–10337. PMLR, 2023b. URL https://proceedings.mlr.press/v202/frantar23a.html.

Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT Press, 2016.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial networks. *CoRR*, abs/1406.2661, 2014. URL http://arxiv.org/abs/1406.2661.

Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both weights and connections for efficient neural networks. *CoRR*, abs/1506.02626, 2015a. URL http://arxiv.org/abs/1506.02626.

Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both weights and connections for efficient neural networks. *CoRR*, abs/1506.02626, 2015b. URL http://arxiv.org/abs/1506.02626.

Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. In Yoshua Bengio and Yann LeCun (eds.), *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. URL http://arxiv.org/abs/1510.00149.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pp. 770–778. IEEE Computer Society, 2016. doi: 10.1109/CVPR.2016.90. URL https://doi.org/10.1109/CVPR.2016.90.

Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, and Song Han. Amc: Automl for model compression and acceleration on mobile devices. In *European Conference on Computer Vision (ECCV)*, 2018.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531, 2015. URL http://arxiv.org/abs/1503.02531.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, 1997. doi: 10.1162/NECO.1997.9.8.1735. URL https://doi.org/10.1162/neco.1997.9.8.1735.

Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861, 2017. URL http://arxiv.org/abs/1704.04861.

Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew G. Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pp. 2704–2713. Computer Vision Foundation / IEEE Computer Society, 2018. doi: 10.1109/CVPR.2018.00286. URL http://openaccess.thecvf.com/content_cvpr_2018/html/Jacob_Quantization_and_Training_CVPR_2018_paper.html.

Raghuraman Krishnamoorthi. Quantizing deep convolutional networks for efficient inference: A whitepaper. *CoRR*, abs/1806.08342, 2018. URL http://arxiv.org/abs/1806.08342.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In Peter L. Bartlett, Fernando C. N. Pereira, Christopher J. C. Burges, Léon Bottou, and Kilian Q. Weinberger (eds.), *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*, pp. 1106–1114, 2012. URL https://proceedings.neurips.cc/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.

Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=rJqFGTslg.

Dmitry Molchanov, Arsenii Ashukha, and Dmitry P. Vetrov. Variational dropout sparsifies deep neural networks. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pp. 2498–2507. PMLR, 2017a. URL http://proceedings.mlr.press/v70/molchanov17a.html.

Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient inference. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017b. URL https://openreview.net/forum?id=SJGCiw5gl.

OpenAI. GPT-4 technical report. *CoRR*, abs/2303.08774, 2023. doi: 10.48550/ARXIV.2303.08774. URL https://doi.org/10.48550/arXiv.2303.08774.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/b1efde53be364a73914f58805a001731-Abstract-Conference.html.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018.

Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach (4th Edition)*. Pearson, 2020. ISBN 9780134610993. URL http://aima.cs.berkeley.edu/.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR*, abs/1910.01108, 2019. URL http://arxiv.org/abs/1910.01108.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In Yoshua Bengio and Yann LeCun (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1409.1556.

Mingjie Sun, Zhuang Liu, Anna Bair, and J. Zico Kolter. A simple and effective pruning approach for large language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024a. URL https://openreview.net/forum?id=PxoFut3dWW.

Mingjie Sun, Zhuang Liu, Anna Bair, and J. Zico Kolter. A simple and effective pruning approach for large language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024b. URL https://openreview.net/forum?id=PxoFut3dWW.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 5998–6008, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

Pingjie Wang, Ziqing Fan, Shengchao Hu, Zhe Chen, Yanfeng Wang, and Yu Wang. Reconstruct the pruned model without any retraining. *CoRR*, abs/2407.13331, 2024. doi: 10.48550/ARXIV.2407.13331. URL https://doi.org/10.48550/arXiv.2407.13331.

Michael Zhu and Suyog Gupta. To prune, or not to prune: Exploring the efficacy of pruning for model compression. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Workshop Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=Sy1iIDkPM.