IMPROVED METHODS FOR STATIC MODEL PRUNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Static model pruning is presented as a performance optimization technique for large language and vision models. The approach aims to identify and remove neurons, connections unlikely to lead to expected generation results for typical user queries. The goal is to obtain a much smaller model that can quickly return results almost as good as those of the unpruned ones. Through careful analysis of pre-trained weights, bias, activations and user queries, an initial mathematical model based on certain probabilities obtained from the environment is developed to improve on previous results for pruned model size, achieving significant improvement in most cases. This paper explores and compares to previously proposed approaches that perform pruning based on other factors.

1 INTRODUCTION

(General Intro) Large language models and large vision models are facing significant performance challenges due to the massive data size and query loads they need to support. These models, along with other related systems, crawl, analyze, and incorporate billions of web pages, videos, and multi-modal data into their network architectures such as transformer. One crucial cost factor is the query processing per user, which must scale with both data size and query load. As a result, large language models devote substantial hardware and energy resources to this 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. A large category of optimization techniques commonly referred to as static or dynamic pruning has emerged in the context of query processing for large foundational models. This paper aims to explore and contribute to the understanding and improvement of these pruning techniques to enhance the performance and efficiency of those models.

(More Specific Intro) In this paper, our attention is directed towards a particular optimization technique known as model pruning. In essence, the approach involves conducting a suitable analysis of the knowledge representation, document collections, and query distribution. The objective is to determine those entries or neurons that are highly likely to yield top user query results for typical queries. Subsequently, any other neurons that are unlikely to contribute to user inputs are ”removed” from the neural network. The aim is to obtain a significantly smaller neural network with a reduced amount of parameters. This pruned network can achieve almost the same quality of results as the unpruned one while requiring much less memory and GPU footprint. Consequently, it leads to faster query processing over a shorter neural network with optimized layers.

(Given a typical example) Consider a leading large foundation model provider today. There are around 10 billion documents incorporated into its knowledge base, with an average of 300 words per document, resulting in a total of approximately 10^{15} tokens. The leading engine receives around 5 billion queries per day, with each query represented by around 10^{11} terms to convey the user's intention for interaction with the large language model. This implies that nearly 117 billion tokens in the knowledge representation could potentially lead to expected output tokens. However, in reality, far fewer tokens actually result in an output within a month. Considering the repetition of queries and postings, more than 99.5% of all routing and neuronal activation and triggers do not yield a single result output from the decoder within a month. Although we cannot reliably identify the 0.5% of active neurons that contribute to the result precisely for the next month, we might hope to identify a large subset of the optimized neurons that contains most of the important information and knowledge representation as the full neural network on common measures of effectiveness.

(Small set of closely related previous work) Previous work on model pruning for large language models and large vision models has primarily focused on approaches such as retaining layers above a global impact threshold or keeping high-scoring neurons in each layer. For detail, we refer to [Cite Wanda paper][Cite SparseGPT paper][Cite Manitude paper]. These efforts have yielded promising results, but there is room for further improvement. The goal of this paper is to build on this existing work and develop a methodology that combines different ideas to achieve a better balance between neural network size and result quality as measured by standard retrieval or information generation quality metrics. Given the feature-rich environment, pruning is considered as a prediction problem where suitable statistical techniques or deep learning methods such as language modeling and machine learning are employed to determine which neuron, weights and layers to keep.

(Paper Organization) The remainder of this paper is organized as follows. In Section 2, we provide background information on learning representation, neural networks, and related pruning technique. We summarize our key contributions in Section 3, highlighting the novelty and significance of our approach. We delve into the technical details of our proposed approach in Section 4, providing a comprehensive explanation of the methodology and algorithms developed. We present and explan our experimental results in Section 5, including some implementation detail and performance analysis. In section 6, we offer concluding remarks, summarizing the main findings of the paper and suggesting potential directions for future research.

2 Background and Related Work

In this section, we first provide some background on neural network architectures, user inputs, pruning, and machine generation. We then discuss previous work related to model pruning in the context of large language and vision models. For additional details on general neural network architectures, we refer to [][].

2.1 Background

2.1.1 Nerual Network Architectures

(TODO:)

2.1.2 Human Input & Machine Generation

(TODO:)

2.1.3 Model Quantization & Compression

(TODO:)

2.1.4 Model Pruning & Indicators Discovery

(TODO:)

2.2 Related Work

2.2.1 Typical Pruning Algorithms

There are three typical network pruning algorithms. (1) The magnitude pruning algorithm[18, 19]: Simplest approach: Prunes weights based purely on their absolute magnitude. Threshold-based: A global threshold is determined based on the desired sparsity ratio. Weights below this threshold are set to zero. Unstructured: Can prune individual weights anywhere in the matrix, potentially leading to irregular sparsity patterns that might not be hardware-friendly. Fast but less accurate: Generally the fastest method, but might remove important connections, leading to a larger accuracy drop compared to more sophisticated methods. (2) The WANDA(Weights and Activations) pruning algorithm[3]: Importance-aware: Considers both weight magnitudes and activation statistics to estimate weight importance. Calibration phase: Requires a calibration step where the model processes a small dataset to collect activation data. Row-wise scaling: Normalizes weight magnitudes within each row based on activation statistics, making the pruning less sensitive to weight scale variations across neurons. Unstructured or structured: Can be applied in an unstructured manner (pruning individual weights) or a structured manner (pruning within blocks of weights). Improved accuracy: Often achieves better accuracy-sparsity trade-offs compared to magnitude pruning. (3) The SparseGPT pruning algorithm[5]: Gradient-based: Leverages gradient information during pruning to identify less important connections. Iterative pruning: Prunes the model iteratively, gradually increasing sparsity while minimizing accuracy loss. Block-sparse structure: Encourages a block-sparse structure, which can be more hardware-efficient for some architectures and libraries. Computationally intensive: Can be more computationally expensive than magnitude or WANDA pruning due to the iterative nature and gradient calculations. State-of-the-art results: Often achieves very high sparsity levels with minimal accuracy degradation, making it suitable for compressing large language models. In summary, Magnitude pruning is the simplest and fastest but might be less accurate. WANDA improves upon magnitude pruning by considering activation information, potentially leading to better accuracy. SparseGPT is a more advanced method that uses gradient information and iterative pruning to achieve high sparsity with minimal accuracy loss, but it comes with higher computational cost.

2.2.2 Query Traces & Model Calibration

(TODO:)

2.2.3 Model Entropy & Perplexity Estimation

(TODO:)

2.2.4 Pruning Theory & Mathmatical Induction

(TODO:)

2.2.5 Comparison to Our Work

(TODO:)

3 Our Contributions

In this paper, we study LLM & LVM model pruning that attempt to achieve a good trade-off between network size and generation quality. Our main contributions are as follows:

1. We describe an approach called Movement that can perform much better than previous approaches;

2. We describe several algorithms closely related to pruning and design a unified benchmark for model evaluation;

3. We perform a comprehensive experimental evaluation via the combinations of different datasets, models and evaluation metrics;

4. We compare human designed algorithms with AIGC generated algorithms, demonstrating the pros and cons on both sides in specific domains such as ”code generation”.

4 Our Proposed Pruning Algorithms

4.1 Algorithm Design Principles

When we are designing the algorithms, we take considerations into the following design principles:

1. Target Sparsity Level: What percentage of weights do we aim to prune? Higher sparsity can lead to greater compression and speedups but might sacrifice more accuracy.

Table 1: Perplexity on pruned model (Llama-7B) from human domain experts

Pruned Level	Wanda
0.01	NA
0.05	NA
0.10	5.696
0.20	5.817
0.30	5.999
0.40	6.387
0.50	7.257
0.60	10.691
0.70	84.905
0.80	5782.432
0.90	19676.668
0.95	28309.178
0.99	108234.484

2. Quality-Size Trade-off: Finding the right balance between model size & speed & quality is crucial. Some algorithms prioritize accuracy (SparseGPT), while others are more aggressive in pursuing sparsity (magnitude).

3. Pruning Criterion: How do you determine which connections to prune? Options may include: Weights (Magnitude), Activation statistics (WANDA), Gradient (SparseGPT) et al.

4. Structured vs. Unstructured Pruning: The formal method attempts to prune individual weights anywhere, potentially leading to irregular sparsity patterns that might not be hardware-friendly. While the latter method attempts to prune in blocks (e.g., 2:4, 4:8), which can be more efficient for some underline hardware and libs.

5. Pruning Schedule: When and how do we prune? One-shot pruning attempts to prune once at the beginning or after training. While the others attempt to incrementally prune over multiple training epochs.

6. Usage of Calibration Data: Some algorithms like WANDA require a small calibration dataset to collect activation statistics before pruning. The choice of this data can impact pruning effectiveness.

7. Hardware Awareness: Consider the target hardware (CPUs, GPUs, specialized accelerators) and design pruning strategies that align with hardware constraints for optimal efficiency.

8. Layer-Wise Sparsity: Allow different layers to have varying sparsity levels based on their sensitivity. It is well-known that NOT all layers contribute equally to a model's performance.

9. Regularization and Stability: Pruning can always lead to instability during training & prediction. An end-to-end model evaluation is needed in order for final model deployment in production system.

4.2 Human Proposed Algorithms

1. Movement Pruning. The Core Idea of this alg is: Instead of directly removing weights, movement pruning identifies unimportant weights and "moves" their values to other more significant connections. This helps preserve the overall information flow within the network. (TODO: If the result is good, we will fill in more detail)

5 Preliminary Experimental Results

5.1 Tables

Table 2: Effectiveness of the weights as a major pruning measure

Pruned Level	**Prune by Weights**
0.01	NA
0.05	NA
0.10	5.806
0.20	6.020
0.30	6.669
0.40	8.601
0.50	17.285
0.60	559.987
0.70	48414.551
0.80	132175.578
0.90	317879.250
0.95	273552.281
0.99	222543.047

Table 3: Effectiveness of the bias as a major pruning indicator

Pruned Level	**Prune by Bias**
0.01	NA
0.05	NA
0.10	NA
0.20	NA
0.30	NA
0.40	NA
0.50	NA
0.60	NA
0.70	NA
0.80	NA
0.90	NA
0.95	NA
0.99	NA

6 Conclusions

Do not change any aspects of the formatting parameters in the style files. In particular, do not modify the width or length of the rectangle the text should fit into, and do not change font sizes (except perhaps in the References section; see below). Please note that pages should be numbered.

7 Citations, figures, tables, references

These instructions apply to everyone, regardless of the formatter being used.

7.1 Citations within the text

Citations within the text should be based on the `natbib` package and include the authors' last names and year (with the "et al." construct for more than two authors). When the authors or the publication are included in the sentence, the citation should not be in parenthesis using `\citet{}` (as in "See **?** for more information."). Otherwise, the citation should be in parenthesis using `\citep{}` (as in "Deep learning shows promise to make progress towards AI (**?**).").

Table 4: One pass code generation and effectiveness evaluation

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

Table 5: Perplexity on pruned model (llama-7B) from AIGC domain expert (o1)

Pruned Level	aigc algorithm 2
0.50	193740.406
0.60	110879.422
0.70	174815.859
0.80	287734.844
0.90	157028.844
0.95	90220.781
0.99	991519.125

The corresponding references are to be listed in alphabetical order of authors, in the REFERENCES section. As to the format of the references themselves, any style is acceptable as long as it is used consistently.

7.2 FOOTNOTES

Indicate footnotes with a number[1] in the text. Place the footnotes at the bottom of the page on which they appear. Precede the footnote with a horizontal rule of 2 inches (12 picas).[2]

7.3 FIGURES

You may use color figures. However, it is best for the figure captions and the paper body to make sense if the paper is printed either in black/white or in color.

8 DEFAULT NOTATION

In an attempt to encourage standardized notation, we have included the notation file from the textbook, *Deep Learning* **?** available at `https://github.com/goodfeli/dlbook_notation/`. Use of this style is not required and can be disabled by commenting out `math_commands.tex`.

Numbers and Arrays

[1] Sample of the first footnote
[2] Sample of the second footnote

Table 6: Effect of pruned model (OPT-1.3B) applying to downstream task - text generation

Pruned Level	**Perplexity**
0.00	NA
0.50	19.191
0.60	23.205
0.70	44.246
0.80	364.304
0.90	3772.829
0.95	8892.167
0.99	22548.809

Table 7: (TODO: Running Time for each pruning algorithm)

Number	**Running Time**
01	TBA
02	TBA
03	TBA
04	TBA
05	TBA
06	TBA
07	TBA
08	TBA
09	TBA
10	TBA

a	A scalar (integer or real)
$\boldsymbol{a}$	A vector
$\boldsymbol{A}$	A matrix
$\mathsf{\mathbf{A}}$	A tensor
$\boldsymbol{I}_n$	Identity matrix with n rows and n columns
$\boldsymbol{I}$	Identity matrix with dimensionality implied by context
$\boldsymbol{e}^{(i)}$	Standard basis vector $[0, \dots, 0, 1, 0, \dots, 0]$ with a 1 at position i
$\text{diag}(\boldsymbol{a})$	A square, diagonal matrix with diagonal entries given by $\boldsymbol{a}$
a	A scalar random variable
$\mathbf{a}$	A vector-valued random variable
$\mathbf{A}$	A matrix-valued random variable

Sets and Graphs

Table 8: (TODO: End-to-end model evaluation)

Number	**Inspiration Score**
01	TBA
02	TBA
03	TBA
04	TBA
05	TBA
06	TBA
07	TBA
08	TBA
09	TBA
10	TBA

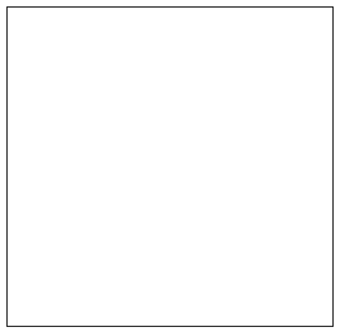

Figure 1: Sample figure caption a.

$\mathbb{A}$	A set
$\mathbb{R}$	The set of real numbers
$\{0, 1\}$	The set containing 0 and 1
$\{0, 1, \dots, n\}$	The set of all integers between 0 and n
$[a, b]$	The real interval including a and b
$(a, b]$	The real interval excluding a but including b
$\mathbb{A} \backslash \mathbb{B}$	Set subtraction, i.e., the set containing the elements of $\mathbb{A}$ that are not in $\mathbb{B}$
$\mathcal{G}$	A graph
$Pa_{\mathcal{G}}(\mathrm{x}_i)$	The parents of x_i in $\mathcal{G}$

Indexing

a_i	Element i of vector $\boldsymbol{a}$, with indexing starting at 1
a_{-i}	All elements of vector $\boldsymbol{a}$ except for element i
$A_{i,j}$	Element i, j of matrix $\boldsymbol{A}$
$\boldsymbol{A}_{i,:}$	Row i of matrix $\boldsymbol{A}$
$\boldsymbol{A}_{:,i}$	Column i of matrix $\boldsymbol{A}$
$\mathsf{A}_{i,j,k}$	Element (i, j, k) of a 3-D tensor $\mathbf{\mathsf{A}}$
$\mathbf{\mathsf{A}}_{:,:,i}$	2-D slice of a 3-D tensor
a_i	Element i of the random vector $\mathbf{a}$

Figure 2: Sample figure caption b.

Figure 3: Sample figure caption c.

Calculus

$\dfrac{dy}{dx}$	Derivative of y with respect to x
$\dfrac{\partial y}{\partial x}$	Partial derivative of y with respect to x
$\nabla_{\boldsymbol{x}} y$	Gradient of y with respect to $\boldsymbol{x}$
$\nabla_{\boldsymbol{X}} y$	Matrix derivatives of y with respect to $\boldsymbol{X}$
$\nabla_{\mathsf{X}} y$	Tensor containing derivatives of y with respect to X
$\dfrac{\partial f}{\partial \boldsymbol{x}}$	Jacobian matrix $\boldsymbol{J} \in \mathbb{R}^{m \times n}$ of $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$
$\nabla_{\boldsymbol{x}}^2 f(\boldsymbol{x})$ or $\boldsymbol{H}(f)(\boldsymbol{x})$	The Hessian matrix of f at input point $\boldsymbol{x}$
$\displaystyle\int f(\boldsymbol{x}) d\boldsymbol{x}$	Definite integral over the entire domain of $\boldsymbol{x}$
$\displaystyle\int_{\mathbb{S}} f(\boldsymbol{x}) d\boldsymbol{x}$	Definite integral with respect to $\boldsymbol{x}$ over the set $\mathbb{S}$

Probability and Information Theory

$P(\mathrm{a})$	A probability distribution over a discrete variable
$p(\mathrm{a})$	A probability distribution over a continuous variable, or over a variable whose type has not been specified
$\mathrm{a} \sim P$	Random variable a has distribution P
$\mathbb{E}_{\mathrm{x}\sim P}[f(x)]$ or $\mathbb{E}f(x)$	Expectation of $f(x)$ with respect to $P(\mathrm{x})$
$\mathrm{Var}(f(x))$	Variance of $f(x)$ under $P(\mathrm{x})$
$\mathrm{Cov}(f(x), g(x))$	Covariance of $f(x)$ and $g(x)$ under $P(\mathrm{x})$
$H(\mathrm{x})$	Shannon entropy of the random variable x
$D_{\mathrm{KL}}(P\Vert Q)$	Kullback-Leibler divergence of P and Q
$\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$	Gaussian distribution over $\boldsymbol{x}$ with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$

Functions

$f : \mathbb{A} \rightarrow \mathbb{B}$	The function f with domain $\mathbb{A}$ and range $\mathbb{B}$
$f \circ g$	Composition of the functions f and g
$f(\boldsymbol{x}; \boldsymbol{\theta})$	A function of $\boldsymbol{x}$ parametrized by $\boldsymbol{\theta}$. (Sometimes we write $f(\boldsymbol{x})$ and omit the argument $\boldsymbol{\theta}$ to lighten notation)
$\log x$	Natural logarithm of x
$\sigma(x)$	Logistic sigmoid, $\dfrac{1}{1 + \exp(-x)}$
$\zeta(x)$	Softplus, $\log(1 + \exp(x))$
$\vert\vert\boldsymbol{x}\vert\vert_p$	L^p norm of $\boldsymbol{x}$
$\vert\vert\boldsymbol{x}\vert\vert$	L^2 norm of $\boldsymbol{x}$
x^+	Positive part of x, i.e., $\max(0,x)$
$\mathbf{1}_{\mathrm{condition}}$	is 1 if the condition is true, 0 otherwise

9 Preparing PostScript or PDF files

Please prepare PostScript or PDF files with paper size "US Letter", and not, for example, "A4". The -t letter option on dvips will produce US Letter files.

Consider directly generating PDF files using `pdflatex` (especially if you are a MiKTeX user). PDF figures must be substituted for EPS figures, however.

Otherwise, please generate your PostScript and PDF files with the following commands:

```
dvips mypaper.dvi -t letter -Ppdf -G0 -o mypaper.ps
ps2pdf mypaper.ps mypaper.pdf
```

9.1 Margins in LaTeX

Most of the margin problems come from figures positioned by hand using `\special` or other commands. We suggest using the command `\includegraphics` from the graphicx package. Always specify the figure width as a multiple of the line width as in the example below using .eps graphics

```
\usepackage[dvips]{graphicx} ...
\includegraphics[width=0.8\linewidth]{myfile.eps}
```

or

```
\usepackage[pdftex]{graphicx} ...
\includegraphics[width=0.8\linewidth]{myfile.pdf}
```

for .pdf graphics. See section 4.4 in the graphics bundle documentation (`http://www.ctan.org/tex-archive/macros/latex/required/graphics/grfguide.ps`)

A number of width problems arise when LaTeX cannot properly hyphenate a line. Please give LaTeX hyphenation hints using the `\-` command.

Author Contributions

If you'd like to, you may include a section for author contributions as is done in many journals. This is optional and at the discretion of the authors.

Acknowledgments

Use unnumbered third level headings for the acknowledgments. All acknowledgments, including those to funding agencies, go at the end of the paper.

A Appendix

You may include other additional sections here.