RL-TRIM: Reinforcement Learning driven Transformer Model Structured Pruning

Milin Narayan Bhade
milinbhade@iisc.ac.in
Department of Computer Science and Automation,
IISc, Bangalore
Sr.No. - 21404

Prof. Chiranjib Bhattacharyya
chiru@iisc.ac.in
Department of Computer Science and Automation, IISc, Bangalore

MTech Project Report

Abstract

Conventional model compression techniques rely on hand-crafted heuristics and rule-based policies that require domain experts to explore a large design space, balancing trade-offs among model size, speed, and accuracy. This process is often sub-optimal and time-consuming . Previous research has explored leveraging reinforcement learning to find model compression policies for CNN-based models such as VGGNet and MobileNet. However, Transformer-based models, due to their significant model size and computational demands, can greatly benefit from advanced compression techniques that reduce memory footprint and computational requirements.

In this work, we introduce **RL-TRIM**, a novel framework that employs reinforcement learning for the structured pruning of Transformer models, specifically targeting models like BERT and LLaMA. Our approach involves pruning at different granularities, including head pruning and intermediate dimension pruning, which directly reduces memory size and computational load, facilitating acceleration on consumer GPUs. By utilizing a reinforcement learning agent to determine the optimal pruning strategy, RL-TRIM achieves a significant balance between model size reduction and performance retention, offering a scalable and efficient solution for optimizing various Transformer architectures.

1 Introduction

Transformer-based [25] models have become essential in a wide range of natural language processing tasks due to their exceptional performance in understanding and generating human language. These models like BERT and, including large language models (LLMs) like LLaMA, are widely used in both research and practical applications, powering everything from search engines to virtual assistants. However, their impressive capabilities come at the cost of significant computational and memory demands, making them challenging to deploy on consumer-grade hardware. Compressing these models is crucial to enhance their usability and accessibility for a broader audience. In this work, we explore the use of reinforcement learning, driven approach to prune transformer-based models. By reducing the memory footprint and computational requirements through structured pruning techniques such as head pruning and intermediate dimension pruning, we aim to make these powerful models more efficient and practical for everyday use.

2 Related Works

Structured pruning have been widely studied as compression techniques in computer vision and natural language processing, where task-specific models like classification and Text Generation are often overparameterized and can be pruned significantly with minimal impact on performance [11], [14], [22], [13], [26], [30]. Unstructured pruning [10], [6],[21] prunes individual neurons instead of struc-

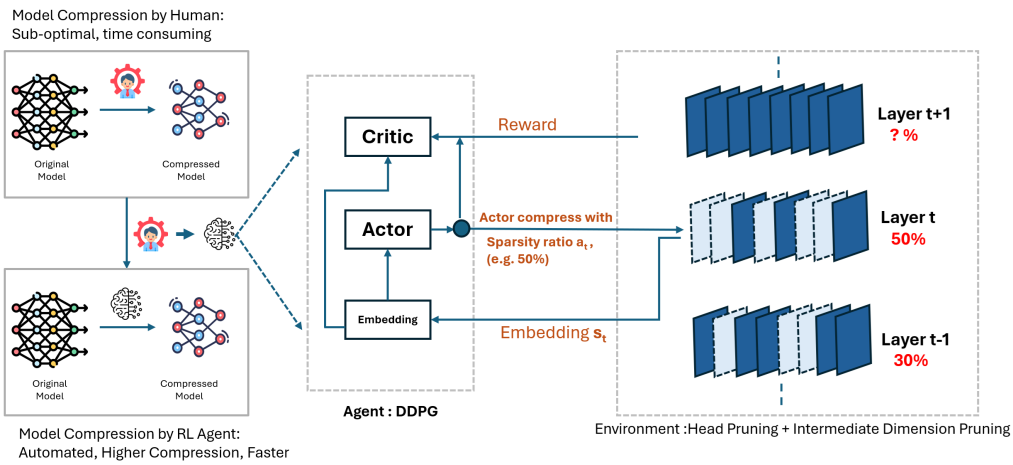

Figure 1: Example of a Transformer model pruning process.

tured blocks. Though unstructured pruning usually achieve higher compression rates, they are not practical for model speedup.

In the realm of structural pruning, researchers have explored various approaches to identify the most effective units for removal, aiming to streamline the architecture of large language models (LLMs) while preserving their functionality. While some studies, such as Laco et al. [28], advocate for treating entire layers as the minimal unit for pruning, others, like Voita et al. and Michel et al. [19], target more specific components like multi-head attention or feed-forward layers.

The granularity of the pruning unit has been a subject of investigation, with projects like CoFi [27] delving into different levels of granularity to optimize pruning techniques. Additionally, innovations like targeted pruning coupled with dynamic batch loading, as demonstrated by Shearedllama et al. [26], offer strategies to achieve high sparsity in pruned models.

Central to the effectiveness of pruning methods are the criteria used to assess the importance of model components. Recent studies, including Wanda et al. [24], Movement-pruning [21], and Blockmovement [14], have contributed valuable insights into refining these criteria, thereby facilitating the removal of non-critical components while maintaining model functionality. LLMPruner [17] takes a novel approach by leveraging dependency graphs to selectively remove coupled structures based on gradient information, thereby preserving the core functionality of LLMs.

In addition to structural pruning, alternative compression techniques such as knowledge distillation [9], quantization [8], and tensor decomposition [2] have also emerged as viable strategies for reducing the computational burden of LLMs. Furthermore, methods like data-free pruning [23], which measure the similarity between neurons to guide pruning decisions, offer promising avenues for model compression without the need for additional training data.

Overall, the landscape of model compression for LLMs is rich with diverse approaches, each offering unique insights and trade-offs in balancing model size reduction with performance preservation.

3 Methodology

We present an overview of our automated Model Compression engine RL-TRIM in Figure 1. We aim to automatically find the redundancy for each layer, characterized by sparsity. We train an reinforcement learning agent to predict the action and give the sparsity, then perform form the pruning. We quickly evaluate the accuracy/perplexity after pruning but before fine-tuning as an effective delegate of final accuracy/perplexity. Then we update the agent by encouraging smaller, faster and more accurate models. RL-TRIM learns through trial and error to find the best policy to prune the model.

3.1 Problem Defination

Model compression is achieved by reducing the number of parameters and the computation required for each layer in transformer-based models. There are two categories of structured pruning explored in this work: head pruning and intermediate layer pruning. *Head pruning* aims to prune individual heads from the model's multi-head attention layers, thereby reducing the computational burden and the number of parameters associated with these heads. *Intermediate layer pruning*, on the other hand, focuses on pruning neurons within the feed-forward layers of the transformer,

which can significantly decrease the model's size and improve computational efficiency. By applying these pruning techniques, the overall complexity of the model is reduced, leading to faster inference times and lower resource consumption while maintaining a high level of performance. Out goal is to precisely find out the effective sparsity for each layer, which used to be manually determined in previous studies [19], [21]. In each Self-Attention layer having dimension of hXh, where h is the hidden size of the model, for each of the parameter matrices Query, Key, Value and Output, consists of `h/head_dim` numbers of heads. Pruning each head will reduce the output size of this matrices by `head_dim`, giving sparsity of `removed_heads/ Total_heads`. Similarly pruning intermediate dimension removes neurons from the output of up_proj matrix and input of down_proj matrix.

We leveraged reinforcement learning to automatically sample the design space and improve the model compression quality. Accuracy of the compressed model is very sensitive to the sparsity of each layer, requiring a fine-grained action space. A DDPG [15] agent searches over a continuous compression ratio and learns through trial and error: penalizing accuracy loss while encouraging model shrinking and speedups. DDPG agent processes the network in a layer-wise manner. For each layer L_t, the agent receives a layer embedding s_t which encodes useful characteristics of this layer, and then it outputs a precise compression ratio a_t . After layer L_t is compressed with a_t, the agent moves to the next layer L_{t+1}. The validation accuracy of the pruned model with all layers compressed is evaluated without fine-tuning, which is an efficient delegate of the fine-tuned accuracy. After the policy search is done, the best-explored model is fine-tuned to achieve the best performance. The reward for the environment is setup to preserve as much accuracy as possible after pruning.

3.2 Automated Compression with Reinforcement Learning

The State Space: For each layer t, we have 10 features that characterize the state s_t

$$(t, L, H, S_h, W_Q, W_K, W_V, reduced, rest, a_{t-1})$$

where t is the layer index, H is number of heads in layer t, S_h is the size of each attention head (usually 64), W_Q, W_K, W_V represents number of weights in Q, K, V matrices of the layer t. $Reduced$ is the total number of reduced FLOPs in previous layers. $Rest$ is the number of remaining FLOPs in the following layers. Before being passed to the agent, they are scaled within [0; 1]. Such features are essential for the agent to distinguish one layer from another.

The Action Space: Agent gives continuous actions a ϵ (0, 1), which enables more fine grained pruning and accurate compression. We observed that constraining action space is essential in case of Large Language Models like LLaMA, as high pruning can damage the model and its generation capability.

Algorithm 1 Predict the sparsity ratio $\texttt{action}_t$ for layer L_t with constrained model size using structured pruning

▷ Initialize the reduced model size so far
if t is equal to 0 **then**
  $\texttt{W}_{\text{reduced}} \leftarrow 0$
end if

▷ Compute the agent's action and bound it with the maximum sparsity ratio
$\texttt{action}_t \leftarrow \mu'(s_t)$
$\texttt{action}_t \leftarrow \min(\texttt{action}_t, \texttt{action}_{\text{max}})$

▷ Compute the model size of the whole model and all the later layers
$\texttt{W}_{\text{all}} \leftarrow \sum_k \texttt{W}_k$
$\texttt{W}_{\text{rest}} \leftarrow \sum_{k=t+1} \texttt{W}_k$

▷ Compute the number of parameters we have to reduce in the current layer if all the later layers are pruned with the maximum sparsity ratio. α is the target sparsity ratio of the whole model.
$\texttt{W}_{\text{duty}} \leftarrow \alpha \cdot \texttt{W}_{\text{all}} - \texttt{action}_{\text{max}} \cdot \texttt{W}_{\text{rest}} - \texttt{W}_{\text{reduced}}$

▷ Bound $\texttt{action}_t$ if it is too small to meet the target model size reduction
$\texttt{action}_t \leftarrow \max(\texttt{action}_t, \texttt{W}_{\text{duty}}/\texttt{W}_t)$

▷ Update the accumulation of reduced model size
$\texttt{W}_{\text{reduced}} \leftarrow \texttt{W}_{\text{reduced}} + \texttt{action}_t \cdot \texttt{W}_t$

return $\texttt{action}_t$

DDPG Agent: We use the deep deterministic policy gradient (DDPG) for continuous control of the compression ratio, which is an off-policy actor-critic algorithm. For the exploration noise process, we use a truncated normal distribution:

$$\mu^{'}(s_t) \sim TN(\mu(s_t|\theta_t^{\mu}), \sigma^2, 0, 1) \quad (1)$$

During exploitation, noise σ is initialized as 0.5 and is decayed after each episode exponentially. The agent receives an embedding state s_t of the layer L_t from the environment and then outputs a sparsity ratio as action a_t. The underlying layer is compressed with a_t (rounded to the nearest

feasible fraction) using a specified compression algorithm (e.g., head pruning). Then the agent moves to the next layer Lt+1, and receives state s_{t+1}. After finishing the final layer L_T, the accuracy of the reward is evaluated in the validation set and returned to the agent.

Search Protocol: For accuracy-Guaranteed Compression, following reward function is used in case of BERT model:

$$R_{FLOPs} = -Error * \log(FLOPs) \quad (2)$$

While in case of LLaMA model, we used perplexity based reward function as it gives a measure of generation quality of the model which is essential in case of language modelling task.

$$R_{Perp} = -\log(Perplexity) * \log(FLOPs) \quad (3)$$

This reward function is sensitive to accuracy/ perplexity and also provides an incentive for reducing FLOPs or model size. Reduction in perplexity value helps to preserve the generation capability of the Large language Model.

4 Experiments

This section presents the experimental results of our framework, RL-TRIM, on two prominent Transformer-based models, BERT-Base and LLama2-7B.

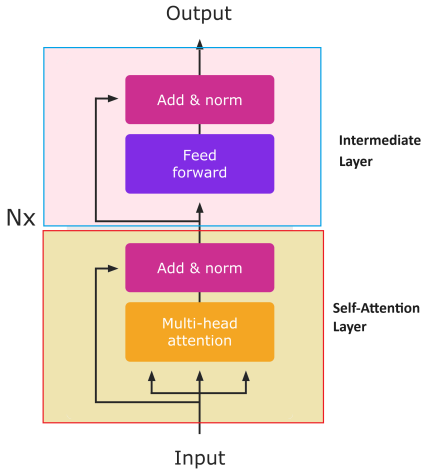

Figure 2: Operations in transformers block

4.1 Goals of the Experiment

The primary goal of this experiment is to evaluate the effectiveness of structured pruning using RL-TRIM in achieving a balance between model size reduction and performance retention. By leveraging reinforcement learning to determine the optimal pruning strategy through a trial-and-error approach, we aim to demonstrate significant model size reduction while maintaining performance. The experiments will compare RL-TRIM's performance on these models with traditional compression techniques, highlighting the benefits of our framework in terms of reduced memory footprint and computational requirements.

4.2 Comparison of BERT and LLaMA

4.2.1 Model Architecture

- **BERT**: BERT is a transformers based encoder-only model with 12 transformer layers. Each layer includes a Self-Attention mechanism with 12 heads and an Intermediate layer with 3072 hidden units.
- **LLaMA2-7B**: Decoder-only model with 32 transformer layers. Each layer includes an attention mechanism with 32 heads and an intermediate dimension size of 11008.

4.2.2 Parameter Distribution

- **BERT**: Contains a significant number of parameters i.e 52%in the Intermediate layers, and 26% in self-attention layers and remaining in embedding layer, totalling to 110 million parameters.
- **LLaMA2-7B**: Contains 66% parameters in intermediate layers and 32% parameters in self-attention layers and remaining in embedding layer, totalling to around 7 Billion parameters.

4.2.3 Applications

- **BERT**: Primarily used for tasks requiring understanding of the input text, such as question answering and text classification.
- **LLaMA2-7B**: Designed for text generation and other tasks requiring the production of human-like text, benefiting from its extensive parameter count and training on diverse data.

4.3 BERT Experiments

By considering self attention and intermediate layer in BERT model for pruning in the pruning process, model can be pruned to much lower memory size, and still maintaining the accuracy.

4.3.1 Setup

We conducted experiments on Nvidia-V100 GPUs with 32GBs of memory. The Accuracy-Guaranteed Compression was investigated using the reward function mentioned in Equation 3, aiming to minimize accuracy loss while reducing FLOPs or model size. Two FLOPs-constrained compression schemes were employed: 80% FLOPs-constrained (Low compression) and 30% FLOPs-constrained (High compression).

Layer Type	No. of Parameters
Embedding Layer	$\sim 21.41\%$
Self-Attention Layer	$\sim 25.85\%$
Intermediate Layer	$\sim 51.71\%$

Table 1: Percentage of parameters present in different layers in BERT-base-uncased

The reward function is calculated based on the accuracy of the model after pruning using the policy from the DDPG agent, acting as a proxy for the final accuracy. A small subset of the validation set for each dataset is used for reward calculation. This reward guides the actor in learning an efficient policy, aiming to maximize the reward through trial and error learning.

The `preserve_ratio` parameter determines the amount of computation preserved in the model after pruning, driving the agent to perform necessary pruning. Head parameters consist of Query (W_Q), Key (W_K), Value (W_V), and Output (W_O) matrices. Ranking heads is based on the sum of weights of these matrices, effectively removing unimportant heads. Similarly, intermediate layer parameters consist of Up projection (W_{up_proj}) and Down projection (W_{down_proj}) matrices, ranked based on L1 norm for row removal.

The results, as shown in Tables 4 and 5, demonstrate that most finetuned models recover accuracy after pruning, with some even improving, potentially due to the regularization effect of pruning.

4.3.2 Evaluation

To assess the performance of a reinforcement learning approach in compressing the BERT-base-uncased model, we conducted experiments on the GLUE benchmark, a comprehensive suite of Natural Language Processing (NLP) tasks. Each task within GLUE was individually fine-tuned using the BERT model, followed by pruning at different checkpoints. This allowed us to evaluate the impact of compression on task-specific performance across the GLUE tasks.

GLUE encompasses nine tasks covering various aspects of language understanding, including single-sentence tasks like CoLA and SST-2, similarity and paraphrasing tasks like MRPC, STS-B, and QQP, as well as natural language inference tasks like MNLI, QNLI, RTE, and WNLI. By evaluating the compressed BERT models across these tasks, we gained insights into the trade-offs between model compression and task-specific performance.

Our experiments revealed that the reinforcement learning-based compression techniques achieved notable reductions in model size while maintaining competitive performance on the GLUE benchmark tasks.

4.3.3 Analysis

Head Pruning From table 2, we can see that on average model is compressed with 36.50% of heads (around 53 heads)from the model using 80% FLOPs constrained compression. Fig 3 shows the number of heads present in model layers after layers being pruned with the compression policy found by DDPG agent. The maximum accuracy drop was found in MRPC dataset which is of 2.46%. On QNLI dataset, after pruning 11.57% improvement in accuracy was recorded after finetuning. All finetuned-models after pruning were retrained for 3 epochs.

Dataset	Heads in original model	Heads in pruned model	Compression
MNLI	144	97	32.64%
SST2	144	100	30.55%
QNLI	144	80	44.45%
RTE	144	95	34.02%
MRPC	144	98	31.94%
QQP	144	94	34.72%
WNLI	144	76	47.22%

Table 2: Number of heads remaining in model after pruning as policy given by AMC for 80% FLOPs constraint compression

Dataset	Heads in original model	Heads in pruned model	Compression
MNLI	144	52	63.88%
SST2	144	52	63.88%
QNLI	144	51	64.58%
RTE	144	51	64.58%
MRPC	144	50	65.27%
QQP	144	51	64.58%
WNLI	144	46	68.05%

Table 3: Number of heads remaining in model after pruning as policy given by AMC for 30% FLOPs constraint compression

Dataset	Ratio	Val. Acc./ F1 before Pruning	Val. Acc./ F1 after Pruning	Val. Acc./ F1 after FT	ΔAcc
MNLI	80% FLOPs	0.8459	0.8114	0.8415	-0.0044
SST2	80% FLOPs	0.9266	0.9231	0.9231	-0.0035
QNLI	80% FLOPs	0.7916	0.6591	0.9073	+0.1157
RTE	80% FLOPs	0.664	0.6281	0.7111	+0.0471
MRPC	80% FLOPs	0.8603/ 0.9042	0.801/0.852	0.8480/ 0.8934	-0.0123
QQP	80% FLOPs	0.909/ 0.8781	0.8731/ 0.83	0.9113/ 0.8805	+0.0023
WNLI	80% FLOPs	0.563	0.561	0.5633	+0

Table 4: Pruning policy RL-TRIM results of BERT-base-uncased finetuned on GLUE benchmark datasets with 80%FLOPs-constrained compression with head pruning

Similar results have been found with 30% FLOPs constrained compression. The DDPG agents found policies were able to prune on average 64.97% of heads (around 93 heads) from the finetuned models without degradation to model quality. The maximum drop in accuracy was found in MRPC dataset of 2.4% while QNLI dataset saw increase of 9% in accuracy.

Table 3 shows the number of heads remaining after pruning of models. Table 5 shows the performance of pruned models using the compression policy given by DDPG agent for 30% FLOPs constrained compression scheme. Fig 4 shows per layer number of heads remaining after pruning models for different GLUE benchmark datasets.

Head + Intermediate Layer Pruning
Pruning heads along with intermediate layers gives model more flexibility and scope for maintaining performance. Much more redundancy and parameters can be found in intermediate layers. DDPG agent learn automatically that pruning heads leads to more loss in performance and thus needs to be preserved. As heads learn semantic meaning of the inputs, they are important for achieving good accuracy on the tasks in GLUE benchmarks. Table 6, shows the performance of model with 75% FLOPs constrained compression. After pruning using the policy given by DDPG agent, maximum drop was found in RTE dataset around 12% and QNLI has improvement of around 3%. Results indicate that while pruning significantly reduces FLOPs, it often leads to a decrease in accuracy, which can be partially recovered through fine-tuning. However, the extent of accuracy recovery varies across datasets, with some experiencing minimal accuracy loss (e.g., SST2) and others seeing more significant declines (e.g., RTE).

4.3.4 Observation:

1. QNLI and RTE are pruned heavily, but QNLI is able to recover but RTE doesn't. This may be due to the fact that size of RTE dataset is very small and thus previous observation suggest that such datasets should not be used for pruning experiments

2. Heads are preserved in almost all experiments of head + intermediate dim pruning, suggesting low redundancy in heads compared to intermediate dimention

3. Intemediate dimentions in later layers are pruned more than in the starting layers, meaning starting layers are required to maintain performance of the model and later layers are more redundant.

Fig 3, and Fig 4 shows how if intermediate layer is pruned with head layer, the model favours to prune intermediate layer and preserve most of the heads across all datasets, except in some like QNLI and RTE.

4.3.5 Conclusion

The results demonstrate that the BERT model can be effectively pruned using a reinforcement learning agent RL-TRIM, with a policy that achieves substantial FLOPs reduction while maintaining good performance across various GLUE benchmark datasets. Despite the initial drop in validation accuracy after pruning, fine-tuning the pruned model allows for significant recovery of performance, as evidenced by the minimal changes in accuracy for most datasets. This indicates that the pruning policy derived from the RL agent is robust and capable of balancing the trade-off between computational efficiency and model accuracy.

Dataset	Ratio	Val. Acc./ F1 before Pruning	Val. Acc./ F1 after Pruning	Val. Acc./ F1 after FT	ΔAcc
MNLI	30% FLOPs	0.8459	0.5721	0.8313	-0.0146
SST2	30% FLOPs	0.9266	0.7958	0.9059	-0.0207
QNLI	30% FLOPs	0.7916	0.7166	0.8848	+0.0932
RTE	30% FLOPs	0.664	0.5270	0.6606	-0.0034
MRPC	30% FLOPs	0.8603/ 0.9042	0.7034/ 0.8051	0.8357/ 0.887	-0.0246
QQP	30% FLOPs	0.909/ 0.8781	0.7677/ 0.68	0.9088/ 0.8756	-0.0002
WNLI	30% FLOPs	0.563	0.6056	0.6056	+0.0426

Table 5: Pruning policy by RL-TRIM results of BERT-base-uncased finetuned on GLUE benchmark datasets with 30% FLOPs-constrained compression with head pruning

Dataset	Ratio	Val. Acc./ F1 before Pruning	Val. Acc./ F1 after Pruning	Val. Acc./ F1 after FT	ΔAcc
MNLI	75% FLOPs	0.8459	0.7211	0.8160	-0.0299
SST2	75% FLOPs	0.9266	0.9254	0.9254	-0.0012
QNLI	75% FLOPs	0.7916	0.4803	0.8213	+0.0297
RTE	75% FLOPs	0.664	0.550	0.5403	-0.1237
MRPC	75% FLOPs	0.8603/ 0.9042	0.75	0.8284	-0.0319
QQP	75% FLOPs	0.909/ 0.8781	0.8128/ 0.7204	0.8813/ 0.8431	-0.0277

Table 6: Pruning policy bertAMC results of bert-base-uncased finetuned on GLUE benchmark datasets with 75% FLOPs-constrained compression with head + Intermediate dim pruning

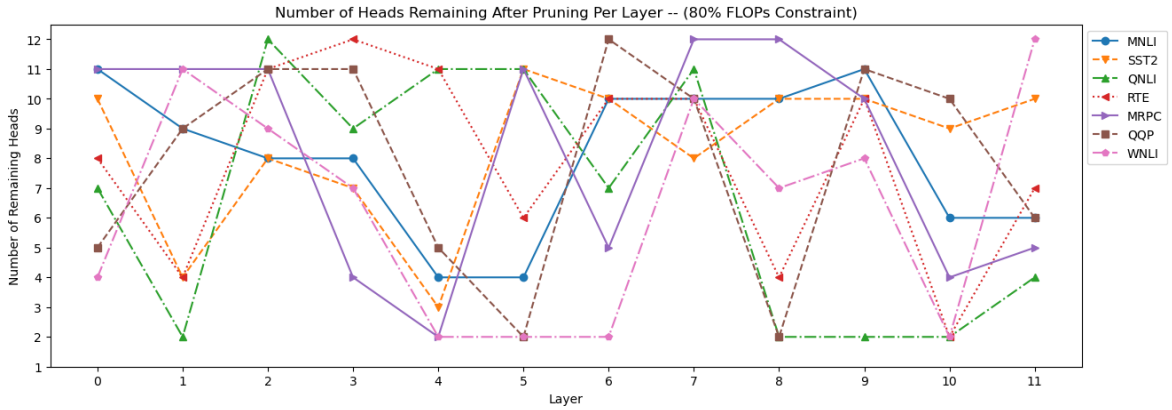

Figure 3: 80% FLOPs Constrained Compression: Number of heads remaining after pruning heads with the policy found by DDPG Agent in each layer for BERT-base-uncased finetuned on GLUE datasets

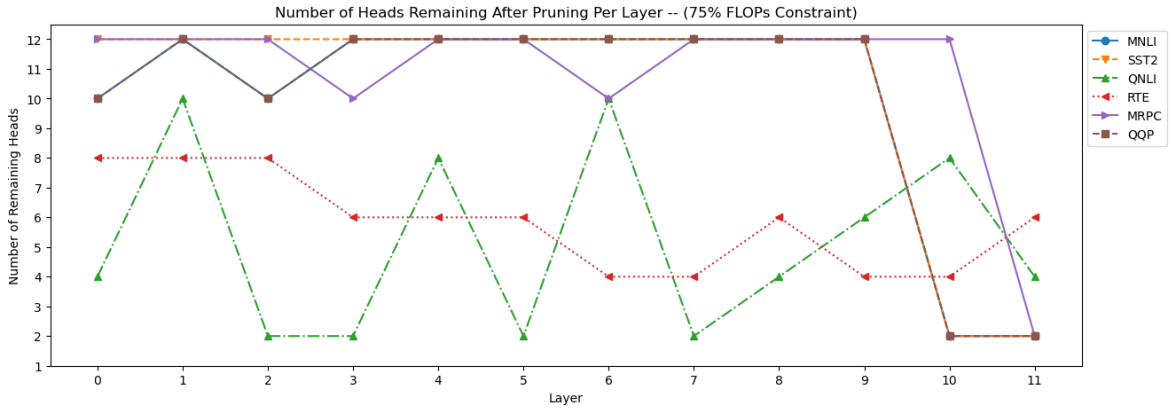

Figure 4: 75% FLOPs Constrained Compression: Number of heads remaining after pruning (head + intermediate dim) with the policy found by DDPG Agent in each layer for BERT-base-uncased finetuned on GLUE datasets

4.4 LLaMA Experiments

We utilized the LLaMA2-7B model for pruning. For pruning, both the attention heads and the intermediate dimensions were considered to reduce the model size and improve efficiency without significantly compromising performance. For pruning, the first and last four layers were not considered, as it was observed in the [17], that pruning these layers had a significant impact on performance. Also giving choice to prune both heads and intermediate dim allows RL agent to explore more effectively as more choice of pruning is available.

In the LLaMA2 model, the distribution of parameters across different layers is as follows given in 7. This distribution highlights the significant parameter allocation in the Head and Intermediate layers, which are crucial for the model's performance.

Table 7: Percentage of parameters present in different layers in layers in LLaMA-2-7B

Layer Type	Percentage of Parameters
Embedding Layer	$\sim 2\%$
Self-Attention Layers	$\sim 32\%$
Intermediate Layers	$\sim 66\%$

Additionally, it is important to note that approximately 1% of the parameters are present per layer in the Head (QKVO), around 2% of the parameters are in each MLP layer (GUD), and about 2% of the parameters are allocated to the LM-Head after last layer.

4.4.1 Setup

The experimental setup involved using a Nvidia V100 GPU for running the experiments. During the reward calculation for the DDPG agent, a small subset of the Wikitext-2 dataset was utilized for perplexity calculation after one episode. Once the training of the DDPG agent is completed, the agent returned a pruning policy. This policy is then applied to prune the LLaMA model. Here the wikitext2 perplexity after each episode as a proxy metric for how good each pruned model is. The pruned model is then subsequently evaluated on a wide variety of tasks to assess its capabilities. The evaluation was conducted using the lm-eval-harness package [7], focusing on tasks such as HellaSwag, OBQA, RTE, ARC-Easy (ARC-e), ARC-Challenge (ARC-C), Winogrande, and BoolQ, all of which were evaluated in a zero-shot setting.

Table 8 provides details about the evaluation criteria for each dataset, specifying what aspects of model performance are assessed.

4.4.2 Evaluation

To assess the performance of the model in the task-agnostic setting, we follow LLaMa's evaluation to perform zero-shot task classification on common sense reasoning datasets: BoolQ [3], PIQA [1], HellaSwag [29], WinoGrande [20], ARC-easy [4], ARC-challenge [4] and OpenbookQA [16]. Following [7], the model ranks the choices in the multiple choice tasks or generates the answer in the open-ended generation. Additionally, we complement our evaluation with a zero-shot perplexity (PPL) analysis on WikiText2 [18] which measures how good is the generation capability of the model. 8 give details about what each dataset evaluated model on and also the metric used.
Evaluating LLMs on a such diverse set of datasets, is crucial for assessing their comprehensive language understanding and reasoning capabilities. It can help to measure its generalization ability and robustness across different types of questions and contexts. his holistic evaluation approach

Table 8: Evaluation Datasets Information

Name of Dataset	What the Dataset Tests	Metric Used
BoolQ (Boolean QA)	Involves answering boolean questions based on a provided passage	Accuracy
OBQA (OpenBookQA)	Involves answering open-domain factual questions	Accuracy
RTE (Recognizing Textual Entailment)	Involves determining if one text entails another, contradicts it, or neither	Accuracy
ARC-Easy (AI2 Reasoning Challenge Easy)	Involves answering grade-school level science questions	Accuracy
ARC-Challenge (AI2 Reasoning Challenge Challenge)	Involves answering more challenging science questions than ARC-e	Acc Norm
Winogrande	Involves resolving coreference and selecting the correct referent in a sentence	Accuracy
HellaSwag	Involves choosing the most plausible continuation to a sentence, requiring commonsense reasoning	Acc Norm
MMLU (Massive Multitask Language Understanding)	Multitask language understanding dataset for evaluating overall language understanding	Accuracy
PIQA (Physical Interaction QA)	Focuses on physical commonsense reasoning, on cause and effect relationships and object interactions	Accuracy

ensures that pruning the LLaMA model using an RL agent does not compromise its performance on a broad spectrum of tasks, thereby confirming its retained or improved capability post-pruning.

4.4.3 Hyperparameter Tuning and Action Space Constraints

In the experimental results, it is crucial to consider the dependence on hyperparameters and the importance of constraining the action space while pruning the LLaMA model. The effective training and convergence of a DDPG agent are highly dependent on the appropriate selection of hyperparameters. Three key parameters play a significant role in this process: `preserve_ratio`, `left_bound`, and `right_bound`.

1. **preserve_ratio (pr)**: This parameter is essential for guiding the model on how much computation to preserve. It directly influences the final FLOPs, steering the model towards an optimal balance between efficiency and performance by using the reward function provided. By adjusting the `preserve_ratio`, we can control the degree of pruning, ensuring that the model retains sufficient computational resources to maintain its capabilities while reducing unnecessary complexity.

2. **left_bound (lb)**: The `left_bound` parameter sets the minimum action that the model can take. This constraint is vital because, without a well-defined `left_bound`, extreme pruning can cause the DDPG agent to collect suboptimal samples during exploration, hindering the agent's convergence. Thus, a carefully chosen `left_bound` ensures that the agent's actions remain within a reasonable range, facilitating effective learning and robust sample collection. The `left_bound` should not be significantly smaller than the preserve ratio. For example, a preserve ratio (`pr`) of 0.9 and a `left_bound` (`lb`) of 0.5 is a bad combination and will result in suboptimal samples during the exploration phase. An `lb` value of less than 0.6 collects samples that prune the model heavily, resulting in poor performance.

3. **right_bound (rb)**: The `right_bound` parameter enforces the minimum pruning required in the model. It dictates what portion of parameters should always be pruned from the given layer, ensuring that the model does not retain too many redundant parameters. Along with the `preserve_ratio`, `right_bound` can also help in constraining pruning in each layer. However, during experiments, it was found that a high value of `right_bound` gives good results, and the `preserve_ratio` is sufficient for maintaining the required pruning. Thus, `right_bound` was set to 1.0 in most cases.

These hyperparameters and their constraints are pivotal in shaping the pruning strategy, ensuring that the DDPG agent operates within a balanced action space. This balance helps in achieving a pruned model that is both efficient and capable, demonstrating the importance of carefully tuning these parameters for optimal results.

Replay Buffer : All standard algorithms for training a deep neural network to approximate the optimal action-value function, $Q^*(s, a)$ make use of an experience replay buffer. This is the set $\mathcal{D}$ of previous experiences $(s_t, a_t, r_t, s_{t+1}, d_t)$. In order for the algorithm to have sta-

ble behavior, the replay buffer should be large enough to contain a wide range of experiences, but it may not always be good to keep everything. If it only uses the very-most recent data, agent will overfit to that and things will break; if it use too much experience, it may slow down your learning.

The following graphs 5, 6 shows how the lb and rb parameters affect experiences $(s_t, a_t, r_t, s_{t+1}, d_t)$ from the DDPG agent to be added to the replay buffer.

The choice of lower bound (`lb`) during the exploration phase significantly impacts the performance of the DDPG agent. As illustrated in Figure 5, optimal settings demonstrate that when the lower bound is set appropriately (`pr`=0.95, `lb`=0.8, `rb`=1.0), the agent primarily collects samples with lower perplexity values, with most actions resulting in perplexities between 0 and 1250 (Figure 5(a)). This careful calibration ensures that the agent's exploration is concentrated on actions leading to lower perplexity values, stabilizing the training process.

Even with a moderate pruning (`pr`=0.7, `lb`=0.5, `rb`=1.0), the agent manages to avoid excessively high perplexities, as shown in Figure 5(b), where the perplexity values are more spread out but still manageable. However, when the lower bound is set too low (`pr`=0.6, `lb`=0.5, `rb`=1.0), the agent collects more high-perplexity samples, as evidenced by the higher frequency of perplexity values up to 5000 in Figure 5(c), indicating less stable training after exploration.

Conversely, during poor runs, the inadequacy of a proper lower bound setting becomes more pronounced, leading to the collection of detrimental high-perplexity samples. Figure 6 highlights this issue. Even with a slightly low `lb` setting (`pr`=0.95, `lb`=0.7, `rb`=0.9), the agent collects higher perplexity values samples (Figure 6(a)), but still within a manageable range (0-3000). However, with a moderate pruning of `pr`=0.7, `lb`=0.65, `rb`=0.8 (Figure 6(b)), the perplexity values are more widely spread and include higher frequencies in the upper range, indicating less effective exploration mostly concentrated near 2500 perplexity.

The situation worsens when the lower bound is set too low (`pr`=0.6, `lb`=0.5, `rb`=0.7), leading to a significant concentration of actions resulting in the maximum perplexity value of 5000, as shown in Figure 6(c). This demonstrates that improper `left_bound` and `right_bound` settings cause the agent to explore suboptimal actions, leading to unstable training due to the collection of high-perplexity samples. Therefore, carefully selecting the this values is crucial to ensure the DDPG agent's exploration phase effectively supports stable and efficient training during exploitation.

The figure illustrates the impact of different hyperparameter settings on the training runs of a DDPG agent, comparing good and bad selections of (`preserve_ratio`, `left_bound`,`right_bound`). In Figure 7 (a), with parameters set to `pr`=0.7, `lb`=0.5, and `rb`=1.0, the agent demonstrates a steady improvement in episode rewards over time and maintains low perplexity values, indicating stable and effective training. Conversely, Figure 7 (b), with parameters `pr`=0.7, `lb`=0.65, and `rb`=0.8, shows less consistent reward progression and higher, more variable perplexity values, suggesting that the agent's exploration phase includes more suboptimal actions. This comparison highlights the critical role of carefully selecting hyperparameters to ensure efficient training and optimal performance of the DDPG agent.

4.4.4 Zero-shot Performance

Table 10, presents the zero-shot performance metrics of the compressed LLaMA2-7B model, detailing how the model performs across various tasks after pruning, and highlighting its capabilities in different evaluation scenarios without continued pretraining

The Table 10 presented showcases the zero-shot performance metrics of the compressed LLaMA2-7B model across various tasks, highlighting how the model's performance changes as pruning ratios increase. The pruning ratios range from 0% to 40%, with corresponding hyperparameters `preserve_ratio` (pr), `left_bound` (lb), `right_bound` (rb) . As pruning increases, the model generally shows a decline in performance, but the extent of this decline varies across different datasets.

Different datasets exhibit varied drops in performance after pruning. For instance, WikiText2 shows a dramatic increase in perplexity, jumping from 5.69 at 0% pruning to 35.08 at 40% pruning, indicating high sensitivity to pruning. Drop in perplexity greatly impacts the generation quality of the model as can be seen in Table 16. BoolQ and ARC-C see a slower, more gradual decline, with BoolQ dropping from 0.76 to 0.45 and ARC-C from 0.42 to 0.31. Hellaswag drops from 0.69 to 0.49, and Winogrande from 0.70 to 0.55, showing moderate sensitivity. RTE decreases from 0.65 to 0.57, indicating moderate sensitivity, while OBQA shows a sharper decline from 0.28 to 0.14, similar to WikiText2.

To mitigate these performance drops, continued pretraining with publicly available high-quality datasets such as RedPajama-1T, SlimPajama, and Cosmopedia could be beneficial. These datasets provide diverse and comprehensive language data that can help the pruned model regain some of its lost capabilities and improve its performance across tasks.

Pruning the LLaMA-2 7B chat version has a detrimental impact on the generation quality of the model. Chat models are trained with instruction fine-tuning to provide coherent answers to prompts given in a question format. Pruning such chat-optimized models greatly impacts the instruction-following ability of the model and shows degradation across

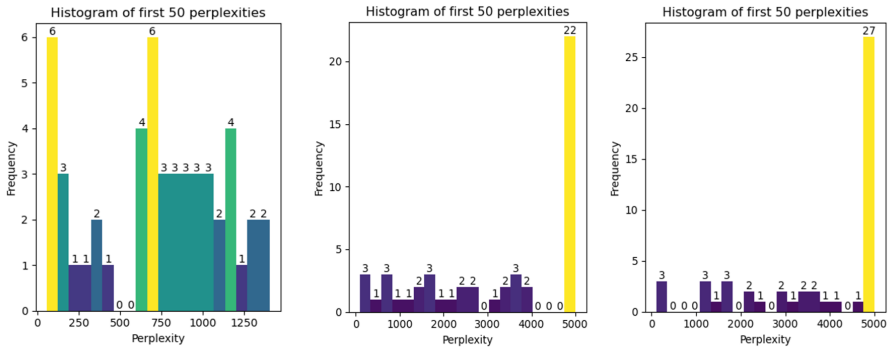

Figure 5: Model perplexity based on DDPG agent actions during exploration for good runs for different combination (a) `pr`=0.95, `lb`=0.8, `rb`=1.0 (b) `pr`=0.7, `lb`=0.5, `rb`=1.0 and (c) `pr`=0.6, `lb`=0.5, `rb`=1.0

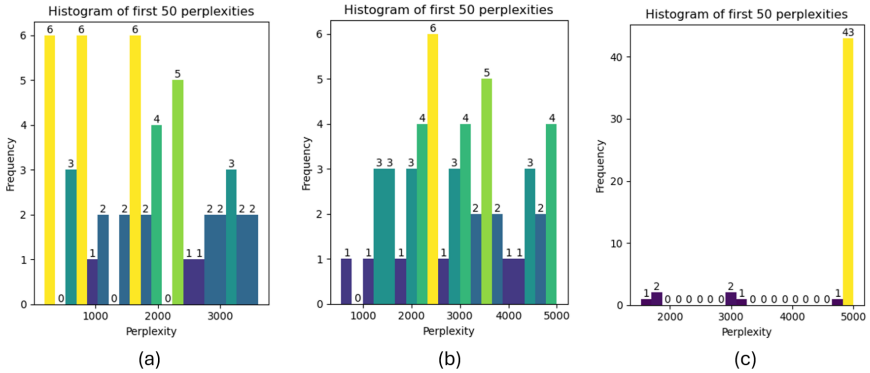

Figure 6: Model perplexity based on DDPG agent actions during exploration for poor runs for different combination (a) `pr`=0.95, `lb`=0.7, `rb`=0.9 (b) `pr`=0.7, `lb`=0.65, `rb`=0.8 and (c) `pr`=0.6, `lb`=0.5, `rb`=0.7

the aforementioned datasets, which test varied capabilities of the model such as factual knowledge, reasoning, and creativity.

To regain chat capability, supervised fine-tuning with datasets like the Stanford Alpaca dataset and similar datasets should be utilized. These datasets are designed to enhance the model's ability to follow instructions and generate coherent, contextually appropriate responses, thereby restoring the performance of the pruned chat models in various conversational tasks. Table 11 shows the reduction of parameters and model size as obtained by different pruning ratios using RL-TRIM.

Table 16 shows generation examples of models pruned using RL-TRIM. We present the generation results for both the pruned and unpruned models. Post-training to regain performance has not been conducted, which sometimes results in incoherent prompts. (Output text have been cleaned to remove repeated sentences).

Pruning Ratio	**Model Size (GBs)**	**Number of Params.**
0	13.51	6738415616
0.1	12.59	6277746688
0.2	11.62	5795127296
0.3	10.67	5318332416
0.4	9.64	4801880064

Table 11: Statistics of pruned models

4.4.5 Comparative Results

For comparing the performance of RL-TrIM, we compare it with LLM-Pruner [17], which is a task-agnostic, data-free method based on dependency graph and uses Taylor-based importance criteria. We also compare with the latest unstructured pruning method called Wanda [24] (weight and activation pruning) in a structured pruning setting. Additionally, we compare with uniform pruning using L1, L2, and Wanda as importance criteria.

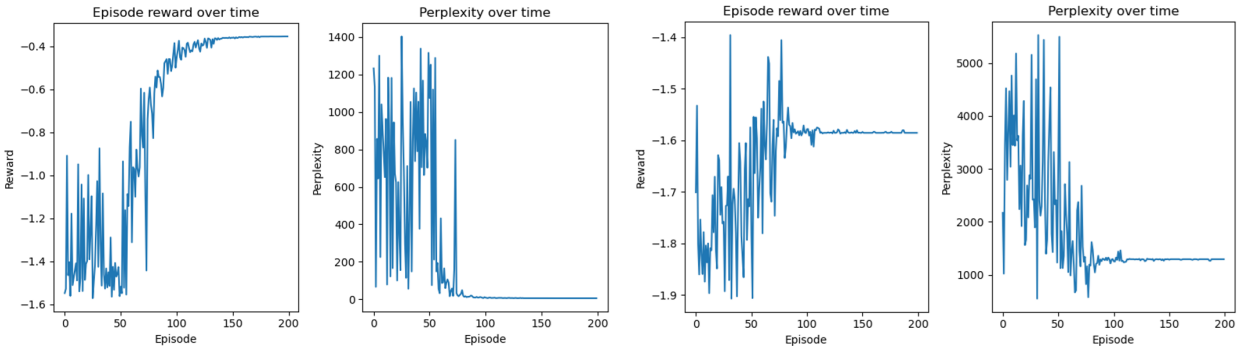

Figure 7: Training runs for good and bad selection of (`preserve_ratio`, `left_bound`, `right_bound`)
(a) `pr`=0.7, `lb`=0.5, `rb`=1.0 (b) `pr`=0.7, `lb`=0.65, `rb`=0.8

Table 9: Zero-shot performance of the compressed LLaMA2-7B using wanda as importance criteria

Pruned Model	Hyperparams (pr, lb, rb)	WikiText2 ↓	BoolQ	ARC-E	ARC-C	Hellaswag	Winogrande	RTE	OBQA	Average
Ratio = 0%		5.69	0.76	0.72	0.42	0.69	0.70	0.65	0.28	0.602
Ratio = 10%	(0.9, 0.8, 1.0)	9.71	0.71	0.64	0.35	0.65	0.65	0.53	0.16	0.521
Ratio = 20%	(0.8, 0.7, 1.0)	8.154	0.77	0.69	0.44	0.63	0.7	0.68	0.28	0.598
Ratio = 30%	(0.7, 0.5, 1.0)	11.051	0.72	0.58	0.38	0.65	0.62	0.49	0.18	0.517
Ratio = 40%	(0.6, 0.5, 1.0)	17.79	0.62	0.49	0.27	0.58	0.66	0.64	0.21	0.495

The Table 12 compare the performance of LLaMA-2-7B pruned using various methods and importance criteria. The results indicate that RL-TrIM with Wanda importance criteria generally outperforms Uniform pruning methods (L1, L2, Wanda) and LLM-Pruner using similar criteria. Even simple L2-based pruning shows competitive performance. Specifically, in the case of 50% pruning, RL-TrIM with Wanda criteria demonstrates superior performance compared to LLM-Pruner, highlighting its effectiveness in maintaining model accuracy despite significant pruning.

4.4.6 Additional Analysis

Observation on generated text by pruned models

When considering the output of pruned models, it becomes evident that without retraining, these models face considerable challenges in generating coherent text. Key issues include:

- **Repeated tokens:** Excessive pruning of parameters leads to the recurrence of repeated tokens. As the model's capacity is reduced, its ability to produce

Table 10: Zero-shot performance of the compressed LLaMA2-7B using L1 Norm as importance criteria

Pruned Model	Hyperparams (pr, lb, rb)	WikiText2 ↓	BoolQ	ARC-E	ARC-C	Hellaswag	Winogrande	RTE	OBQA	Average
Ratio = 0%		5.69	0.76	0.72	0.42	0.69	0.70	0.65	0.28	0.602
Ratio = 5%	(0.95,0.8, 1.0)	6.05	0.74	0.7	0.43	0.68	0.69	0.72	0.31	0.612
Ratio = 10%	(0.9, 0.8, 1.0)	14.53	0.69	0.59	0.42	0.64	0.6	0.58	0.22	0.534
Ratio = 20%	(0.8, 0.7, 1.0)	15.14	0.67	0.51	0.36	0.65	0.69	0.63	0.21	0.531
Ratio = 30%	(0.7, 0.5, 1.0)	19.43	0.71	0.53	0.32	0.56	0.66	0.64	0.23	0.521
Ratio = 40%	(0.6, 0.5, 1.0)	35.08	0.45	0.41	0.31	0.49	0.55	0.57	0.14	0.417

diverse and contextually relevant text diminishes, resulting in the unnatural repetition of phrases or words.

- **Loss of reasoning abilities:** Pruning can cause the model to lose its reasoning abilities. With fewer parameters, the model struggles to maintain logical coherence, failing to string together ideas meaningfully.

- **Generation of garbage text:** Excessive pruning can make the model generate nonsensical text, full of disconnected sentences and random ideas. This makes the output practically useless for many tasks.

- **Domain-specific degradation:** Pruning affects different subjects differently. For example, if a model was trained on a lot of science data but very little history data, it might struggle to generate accurate historical information after pruning. Understanding these differences is crucial for fixing any issues and making sure the model works well across all topics.

- **Loss of context and generation of irrelevant information:** Pruning can disrupt the model's understanding of context, leading to the generation of irrelevant information. Without access to sufficient parameters, the model may fail to grasp the context of the input text, resulting in the insertion of unrelated or erroneous content. Pruning can confuse the model, causing it to switch topics abruptly.

Importance of Pruning for Large Language Models (LLMs) and Challenges

Pruning is crucial for LLMs due to the high costs of the prune-retrain paradigm, which is infeasible for LLMs due to their massive scale. Effective pruning strategies that maintain model capabilities without extensive retraining are essential.

LLMs are trained on massive corpora for next-token prediction, a process that is computationally and financially demanding. For instance, the pretraining dataset for LLaMA2-7B, such as *RedPajama-1T*, is about 5TB in size and requires extensive processing resources. Thus, pruning should aim to reduce model size with minimal performance impact.

Pruning LLMs presents unique challenges compared to CNNs. LLMs, with billions of parameters, are highly sensitive to pruning, and identifying redundant parameters is complex. Moreover, the diverse and vast training data of LLMs makes it harder to determine redundancy, and retraining on this scale is often impractical.

Conclusion

Careful choice of importance criteria and pruning ratios across layers in large language models (LLMs) can hinder the pruning process as it requires significant human expertise, making it time-consuming and potentially suboptimal. The RL-TRIM framework we have devised addresses these challenges by eliminating human heuristics and introducing a learning-based approach to determine the optimal pruning policy for the model. Through this framework, the RL agent efficiently learns which layers and components are crucial for maintaining the model's performance. By adjusting the reward function, different objectives can be prioritized. In our case, we used perplexity as the criterion to preserve the model's generation capabilities, but other criteria can be employed to achieve various goals, further enhancing the flexibility and effectiveness of the pruning process.

Future Works

Future work should focus on incorporating inference time information into the reward function to enhance the practical usability of pruned models, ensuring that reductions in model size also translate to faster inference times. Developing better importance criteria for pruning will be essential to more accurately identify and remove less critical components, thereby improving the efficiency of the pruning process. Additionally, continued pretraining of the model post-pruning will be crucial for regaining and potentially enhancing performance. Additionally, continued pretraining of the model post-pruning will be crucial for regaining and potentially enhancing performance. Comparative studies with state-of-the-art methods such as ShearedLLama [26], LoRAPrune [30], and LLMPruner [17] will be conducted to validate the effectiveness of our proposed techniques. Exploring mechanisms to prune entire layers, similar to those used in the LACO [28] approach, and employing techniques like Low-Rank Adaptation (LoRA) [12] for parameter-efficient fine-tuning will further advance this research area. Utilizing smaller subsets of pretraining data like RedPajama [5] to recover performance post-pruning and developing more effective reward functions tailored to specific constraints will also be pivotal in advancing this field.

References

[1] Y. Bisk, R. Zellers, R. L. Bras, J. Gao, and Y. Choi. Piqa: Reasoning about physical commonsense in natural language, 2019.

[2] Y. Cheng, D. Wang, P. Zhou, and T. Zhang. A survey of model compression and acceleration for deep neural networks, 2020.

[3] C. Clark, K. Lee, M.-W. Chang, T. Kwiatkowski, M. Collins, and K. Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions, 2019.

[4] P. Clark, I. Cowhey, O. Etzioni, T. Khot, A. Sabharwal, C. Schoenick, and O. Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge, 2018.

[5] T. Computer. Redpajama: an open dataset for training large language models, 2023.

[6] E. Frantar and D. Alistarh. Sparsegpt: Massive language models can be accurately pruned in one-shot, 2023.

[7] L. Gao, J. Tow, B. Abbasi, S. Biderman, S. Black, A. DiPofi, C. Foster, L. Golding, J. Hsu, A. Le Noac'h, H. Li, K. McDonell, N. Muennighoff, C. Ociepa, J. Phang, L. Reynolds, H. Schoelkopf, A. Skowron, L. Sutawika, E. Tang, A. Thite, B. Wang, K. Wang, and A. Zou. A framework for few-shot language model evaluation, 12 2023.

[8] A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney, and K. Keutzer. A survey of quantization methods for efficient neural network inference, 2021.

[9] J. Gou, B. Yu, S. J. Maybank, and D. Tao. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129(6):1789–1819, 2021.

[10] S. Han, H. Mao, and W. J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding, 2016.

[11] Y. He, J. Lin, Z. Liu, H. Wang, L.-J. Li, and S. Han. Amc: Automl for model compression and acceleration on mobile devices, 2019.

[12] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. Lora: Low-rank adaptation of large language models, 2021.

[13] B.-K. Kim, G. Kim, T.-H. Kim, T. Castells, S. Choi, J. Shin, and H.-K. Song. Shortened llama: A simple depth pruning for large language models, 2024.

[14] F. Lagunas, E. Charlaix, V. Sanh, and A. M. Rush. Block pruning for faster transformers, 2021.

[15] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning, 2019.

[16] M. Luo, S. Chen, and C. Baral. A simple approach to jointly rank passages and select relevant sentences in the obqa context, 2022.

[17] X. Ma, G. Fang, and X. Wang. Llm-pruner: On the structural pruning of large language models, 2023.

[18] S. Merity, C. Xiong, J. Bradbury, and R. Socher. Pointer sentinel mixture models, 2016.

[19] P. Michel, O. Levy, and G. Neubig. Are sixteen heads really better than one?, 2019.

[20] K. Sakaguchi, R. L. Bras, C. Bhagavatula, and Y. Choi. Winogrande: An adversarial winograd schema challenge at scale, 2019.

[21] V. Sanh, T. Wolf, and A. M. Rush. Movement pruning: Adaptive sparsity by fine-tuning, 2020.

[22] M. Santacroce, Z. Wen, Y. Shen, and Y. Li. What matters in the structured pruning of generative language models?, 2023.

[23] S. Srinivas and R. V. Babu. Data-free parameter pruning for deep neural networks, 2015.

[24] M. Sun, Z. Liu, A. Bair, and J. Z. Kolter. A simple and effective pruning approach for large language models, 2024.

[25] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need, 2023.

[26] M. Xia, T. Gao, Z. Zeng, and D. Chen. Sheared llama: Accelerating language model pre-training via structured pruning, 2024.

[27] M. Xia, Z. Zhong, and D. Chen. Structured pruning learns compact and accurate models, 2022.

[28] Y. Yang, Z. Cao, and H. Zhao. Laco: Large language model pruning via layer collapse, 2024.

[29] R. Zellers, A. Holtzman, Y. Bisk, A. Farhadi, and Y. Choi. Hellaswag: Can a machine really finish your sentence?, 2019.

[30] M. Zhang, H. Chen, C. Shen, Z. Yang, L. Ou, X. Yu, and B. Zhuang. Loraprune: Pruning meets low-rank parameter-efficient fine-tuning, 2023.

Table 12: Comparision between different methods with L1, L2 and Wanda as importance criteria

Method	Pruning Ratio	WikiText2 ↓	ARC-E	ARC-C	OBQA	BoolQ	RTE	Hellaswag	Winogrande	Average
Base Model	0	5.69	0.71	0.42	0.28	0.76	0.63	0.7	0.71	0.602
RL-TrIM - L1	0.2	19.99	0.56	0.37	0.22	0.71	0.55	0.61	0.64	0.52
RL-TrIM - L2	0.2	18.92	0.52	0.43	0.2	0.72	0.61	0.62	0.66	0.5385
RL-TrIM - Wanda	0.2	**8.154**	**0.69**	**0.44**	**0.28**	**0.77**	**0.68**	**0.63**	**0.7**	**0.5985**
LLM-Pruner - L1	0.2	242.78	0.32	0.29	0.16	0.58	0.52	0.39	0.50	0.399
LLM-Pruner - L2	0.2	895.04	0.28	0.298	0.134	0.533	0.541	0.325	0.51	0.374
Uniform - L1	0.2	2516.98	0.23	0.24	0.14	0.45	0.49	0.33	0.55	0.3471
Uniform - L2	0.2	234.81	0.41	0.30	0.16	0.59	0.46	0.39	0.57	0.4114
Uniform - Wanda	0.2	9.26	0.66	0.34	0.23	0.67	0.66	0.62	0.63	0.544

Table 13: Zero shot performance of LLaMA-2-7B pruned with LLM-Pruner using Taylor Based importance criteria for 20% pruning (Directly used from Paper)

Method	Pruning Ratio	WikiText2 ↓	ARC-E	ARC-C	OBQA	BoolQ	PIQA	Hellaswag	Winogrande	Average
LLM-Pruner (Base)	0	12.61	0.67	0.4138	0.424	0.731	0.78	0.72	0.67	0.6325
LLM-Pruner (Taylor Based)	0.2	19.77	0.59	0.37	0.39	0.59	0.75	0.65	0.61	0.5682

Table 14: Zero shot performance of LLaMA-2-7B pruned with RL-TrIM using Wanda importance criteria

Method	Pruning Ratio	WikiText2 ↓	ARC-E	ARC-C	OBQA	BoolQ	Hellaswag	Winogrande	Average
Base model	0	5.69	0.71	0.42	0.28	0.76	0.7	0.71	0.602
RL-TrIM - Wanda	0.5	48.38	0.40	0.29	0.19	0.61	0.45	0.59	0.4216

Table 15: Zero shot performance of LLaMA-2-7B pruned with LLM-Pruner using Taylor Based importance criteria for 50% pruning (Directly used from Paper)

Method	Pruning Ratio	WikiText2 ↓	ARC-E	ARC-C	OBQA	BoolQ	Hellaswag	Winogrande	Average
LLM-Pruner (Base)	0	12.61	0.67	0.41	0.42	0.73	0.73	0.67	0.6073
LLM-Pruner (Taylor Based)	0.5	106.07	0.32	0.25	0.34	0.52	0.35	0.49	0.384

Table 16: Generated Examples from the Compressed LLaMA model using RL DDPG agent

Model	Generated Sentences
LLaMA (unpruned)	Traveling to new countries allows you to experience diverse cultures and traditions. Walking through the bustling markets, tasting local delicacies, and interacting with people can provide a deeper understanding of the world. Traveling can also help you develop a sense of independence and self-reliance. When you travel alone, you are responsible for your own safety, transportation, and accommodation. This can help you build confidence and resilience, which can be beneficial in both personal and professional life. Traveling can also help you develop a sense of adventure and curiosity. When you travel to new places, you are constantly exposed to new experiences and challenges. This can help you develop a sense of adventure and curiosity, which can be beneficial in both personal and professional life.
	Explainable AI (XAI) is becoming increasingly important as AI systems are deployed in critical areas. By providing insights into how models make decisions, XAI aims to increase trust and transparency in AI systems. One of the key challenges in XAI is the need to balance explainability with model performance. Explaining a model's decision-making process can be computationally expensive, which can negatively impact model performance. Additionally, the explanations provided by XAI methods may not always be accurate or complete, which can lead to misinterpretations of the model's behavior. ...
	Reinforcement learning algorithms learn optimal behaviors through trial and error. The most common reinforcement learning algorithms are: Q-learning: This algorithm uses a table to store the values of the Q-function for each state-action pair. The algorithm then updates the values of the Q-function based on the current state and action. Policy gradient: This algorithm uses a gradient descent algorithm to update the policy parameters. The policy gradient is the gradient of the expected reward with respect to the policy parameters. Deep Q-network: This algorithm uses a deep neural network to learn the Q-function. ...
	Artificial intelligence is transforming the healthcare industry by enabling healthcare providers to improve patient care, reduce costs, and increase efficiency. Artificial intelligence is transforming the healthcare industry by enabling healthcare providers to improve patient care, reduce costs, and increase efficiency. AI is being used to improve patient outcomes, reduce the risk of medical errors, and improve the efficiency of healthcare operations.
LLaMA Pruned 10% (pr=0.9)	Traveling to new countries allows you to experience diverse cultures and traditions. Walking through the bustling markets, tasting local delicacies, and interacting with people can provide a deeper understanding of the world. Traveling to new countries allows you to experience diverse cultures and traditions. Traveling to new countries allows you to experience different landscapes and environments. Traveling to new countries allows you to explore and discover new places. Traveling to new countries allows you to experience different cultures and traditions.
	Explainable AI (XAI) is becoming increasingly important as AI systems are deployed in critical areas. By providing insights into how models make decisions, XAI aims to help users understand and trust the decisions being made by the system. AI is a powerful tool that can be used to improve a wide range of industries, from healthcare to finance.
	Reinforcement learning algorithms learn optimal behaviors through the use of reinforcement signals. The most common reinforcement signal is a reward, which is a number or other scalar value that is assigned to the outcome of an action. For example, a reward of 1 is often given to a person who completes a task correctly. A reward of 0 is usually given to a person who fails a task. The more positive the reward, the more likely the person will be to repeat the task. The less positive the reward, the less likely the person will be to try the task again. The more negative the reward, the less likely the person will not try the task again.
	Artificial intelligence is transforming the healthcare industry by enabling hospitals to provide more efficient and effective care. The healthcare industry is one of the largest and most important industries in the world. It is a $1 trillion dollar industry that employs over 1.1 million people. The healthcare industry is also one of the most important industries in the world because it provides health care to people.
LLaMA Pruned 30% (pr=0.7)	Traveling to new countries allows you to experience diverse cultures and traditions. Walking through the bustling markets, tasting local delicacies, and interacting with people can provide a deeper understanding of the world. Traveling to new countries allows you to learn a foreign language. Learning a new language can be a great way to improve your skills and skills. Traveling to new countries allows you to see the world from a different perspective....
	Explainable AI (XAI) is becoming increasingly important as AI systems are deployed in critical areas. By providing insights into how models make decisions, XAI aims to help users understand and control the power of AI. The goal of this project is to develop a framework for implementing the concept of transparency in the context of the AI field.
	Reinforcement learning algorithms learn optimal behaviors through repeated exposure to a problem. Their goal is to find the optimal policy that will allow the algorithm to achieve the highest level of performance. The goal of a neural network is to learn the optimal parameters of the algorithm. The goal of a neural network is to learn the parameters of the algorithm.
	Artificial intelligence is transforming the healthcare industry by enabling hospitals to identify and treat patients with a wide range of diseases and disabilities. The healthcare industry is being transformed by the introduction of new technology, in particular, by the introduction of new technology, in particular...