COPAL: Continual Pruning in Large Language Generative Models

Srikanth Malla [1]   **Joon Hee Choi** [1]   **Chiho Choi** [1]

Abstract

Adapting pre-trained large language models to different domains in natural language processing requires two key considerations: high computational demands and model's inability to continual adaptation. To simultaneously address both issues, this paper presents COPAL (**CO**ntinual **P**runing in **A**daptive **L**anguage settings), an algorithm developed for pruning large language generative models under a continual model adaptation setting. While avoiding resource-heavy finetuning or retraining, our pruning process is guided by the proposed sensitivity analysis. The sensitivity effectively measures model's ability to withstand perturbations introduced by the new dataset and finds model's weights that are relevant for all encountered datasets. As a result, COPAL allows seamless model adaptation to new domains while enhancing the resource efficiency. Our empirical evaluation on a various size of LLMs show that COPAL outperforms baseline models, demonstrating its efficacy in efficiency and adaptability.

1. Introduction

The advent of Large Language Models (LLMs) such as GPT-3 (Brown et al., 2020) and LLaMA (Touvron et al., 2023) has been a landmark in natural language processing (NLP). Adapting these pre-trained LLMs to diverse domains has offered unprecedented capabilities in various NLP tasks including language understanding and generation (Gururangan et al., 2020). However, there are two primary challenges associated with this approach: (i) extensive computational requirements; and (ii) limited model adaptability. Considering the large-scale models and datasets, the retraining procedure necessitates considerable computational effort, which often limits its feasibility in resource-constrained environments. Moreover, once the models are updated for
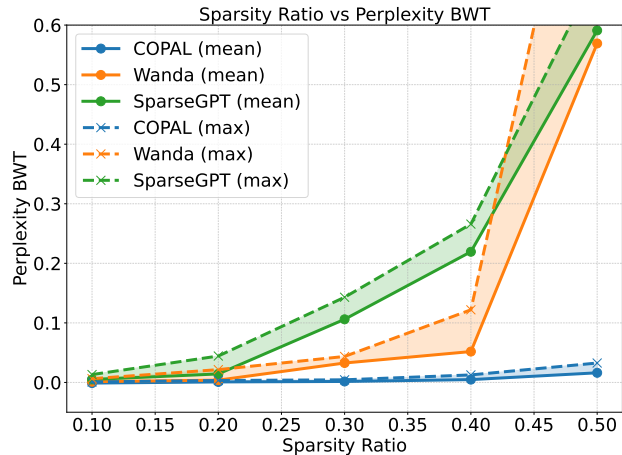

Figure 1: Motivation: average (mean) and worst (max) case scenarios of backward transfer (BWT) in perplexity with increase in sparsity ratio in unstructured continual pruning of LLaMA-7B

certain tasks or domains, they may not exhibit the same level of performance when confronted with data that deviate significantly from what they were trained on (known as "catastrophic forgetting").

Traditional methods have tackled these challenges separately from different standpoints, either utilizing neural network pruning (Frantar & Alistarh, 2023) or adopting continual learning techniques (Kirkpatrick et al., 2017). The former (*i.e.*, pruning) suggests to eliminate less critical connections (Frankle & Carbin, 2018) or structural elements (He et al., 2017) of the neural network, being beneficial in enhancing the efficiency of model inference. However, it inherently lacks a flexibility in continually adapting to new information over time, as pruning involves domain-specific training (Liu et al., 2020) or calibration (Frantar & Alistarh, 2023) procedure. On the other hand, the latter (*i.e.*, continual learning) aims to adapt models to evolving data streams, accommodating new tasks and knowledge without catastrophic forgetting. With its need for finetuning on a series of datasets, this approach does not alleviate the ongoing concern of substantial computational costs for model execution. Contrary to this trend, few research efforts (Dekhovich et al., 2023) have been recently made by applying pruning meth-

---

[1]Samsung Semiconductor, San Jose, USA. Correspondence to: Joon Hee Choi <jh4.choi@samsung.com>, Chiho Choi <chiho1.choi@samsung.com>.

ods within a continual learning context. However, they still require re-training of the base pre-trained models on the new datasets, with a potential reduction of their generalizability originially set toward various tasks.

Our framework, COPAL (COntinual Pruning in Adaptive Language settings), is designed to simultaneously address both issues (computational inefficiency and model inadaptability), inspired by recent studies in (Sun et al., 2023; Frantar & Alistarh, 2023) on post-training pruning. While optimizing LLMs without involving any further training or finetuning, we find model's weights that are relevant for all previously encountered datasets. This process involves reducing the complexity and size of the model, which naturally enables the model pruning. Additionally, our solution does not require storing past datasets or reusing them for pruning. Instead, previously pruned network with remaining parameters already contains sufficient meta information to retain the performance for past datasets. Upon transitioning to the new dataset, COPAL employs a small set of calibration data to guide the pruning process based on the sensitivity analysis we propose. Since the sensitivity measures model's robustness to perturbation caused by the new dataset, this strategy enables the model to seamlessly adapt to new information while preserving previous knowledge. To the best of our knowledge, we are the first to introduce the concept of *continual pruning* that addresses pruning under a continual model adaptation setting, bypassing the requirement for model re-training. This marks a substantial advancement in the field of LLM optimization.

To illustrate the effectiveness of our approach, Figure 1 demonstrates the impact of COPAL in both average and worst-case scenarios, showcasing how continual pruning can maintain performance even with increased sparsity. This highlights COPAL's ability to adeptly navigate the balance between model complexity and performance, a crucial factor in real-world applications.

Our contributions are threefold:

- We explore the inherent challenges in finetuning pre-trained LLMs and provide a strategic solution to address both computational inefficiency and limited model adaptability.

- We propose a mathematical formulation for the concept of continual pruning, utilizing sensitivity analysis to enable the pruning process under a continual model adaptation setting.

- Through empirical evaluations on large-scale language models, including LLaMA-7B, 13B, 30B, 65B, we show that COPAL outperforms baseline methods, setting a new standard in LLM optimization for both efficiency and adaptability.

2. Prior Works

Pre-trained LLMs have shown generizability across various NLP tasks. Despite their advancements, finetuning of LLMs face challenges in computational efficiency, requiring optimization strategies like pruning for practical deployment.

Pruning. Pruning plays an important role in optimizing neural network architectures, especially in large models. *Types of Pruning:* Structured Pruning involves removing entire structural elements such as neurons, filters, or layers. Key contributions in this area include network slimming Liu et al. (2017), channel pruning (He et al., 2017), and optimizing network architectures with minimal performance trade-offs (Luo et al., 2017; Yu et al., 2018). Structured pruning is particularly relevant for simplifying large language models and enhancing their efficiency. Whereas, unstructured pruning focuses on the selective removal of individual weights. It aims to eliminate less critical connections within the network. Pioneering techniques like "Optimal Brain Damage" (LeCun et al., 1989) and "Deep Compression" (Han et al., 2015) have significantly contributed to reducing neural network size. The "Lottery Ticket Hypothesis" (Frankle & Carbin, 2018) suggests the existence of smaller, effective sub-networks, which is a critical concept for large model optimization.
Stages of Pruning: Pruning can be applied before training, during training, or post-training. Each stage offers unique advantages and challenges. For instance, SNIP (Lee et al., 2018) and GraSP (Wang et al., 2020) focus on identifying critical connections before training starts. On the other hand, dynamic sparse training (Liu et al., 2020), soft filter pruning (He et al., 2018) or with knowledge distillation(Xia et al., 2022; Yang et al., 2022) integrate pruning into the training process. Post-training techniques like those developed by Sun et al. (2023) and Frantar & Alistarh (2023) are particularly relevant for large language models, enabling efficient pruning after the model has been fully trained and avoid re-training burden.

Continual Learning. Continual learning focuses on adapting large language models in dynamic environments, enabling continuous learning without losing prior knowledge. *Catastrophic Forgetting (McCloskey & Cohen, 1989):* Methods like rehearsal techniques (e.g., Gradient Episodic Memory (Lopez-Paz & Ranzato, 2017), Experience Replay (Rolnick et al., 2019)), regularization methods (e.g., Elastic Weight Consolidation (Kirkpatrick et al., 2017), and Synaptic Intelligence (Zenke et al., 2017)) have been developed to mitigate this issue.
Architectural Methods: Adapting the network architecture for continual learning is another approach. Techniques like Progressive Neural Networks (Rusu et al., 2016) and Dynamically Expandable Networks (Yoon et al., 2017) focus on evolving the model's structure to accommodate new tasks.
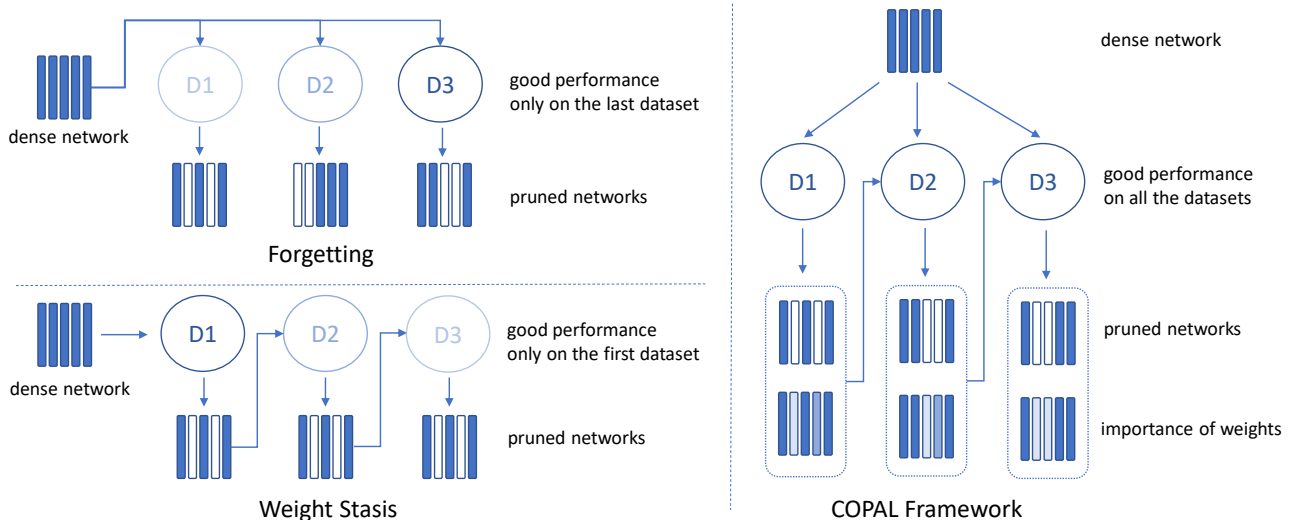
Figure 2: Overview of Forgetting, Weight Stasis and COPAL Framework in Continual Pruning. D represents calibration data used for pruning, and D1→D2→D3 is the dataset incremental order here.

Continual learning, with its need for finetuning on new datasets, not only risks diminishing the base model's generalizability but also faces heightened challenges due to rising training costs of LLMs and the evolving nature of real-world NLP applications.

Pruning in Continual Setting. In the field of continual learning, the role of pruning is useful for maintaining and evolving neural network structures effectively. The research by Yan et al. (2022) on Bayesian sparse networks in continual learning settings has been impactful. Similarly, the work of Dekhovich et al. (2023) highlights the importance of preserving efficiency in class-incremental learning. Although these studies emphasize the importance of pruning to keep neural networks efficient and adaptable to new tasks, the potential reduction in a base model's generalizability and increasing training costs of LLMs training are inevitable. Unlike all existing techniques, we tackle these challenges with a new method for optimizing LLMs, which bypasses the need for re-training as a notion of *continual pruning*.

3. Problem Formulation

Continual Pruning involves the pruning of model's weights for continually evolving tasks or datasets without losing the original capability of pre-trained LLMs, such as performance on previously encountered datasets. In this section, we introduce continual pruning and explain two challenges that may arise in this new domain.

3.1. Continual Pruning

Continual pruning clearly differs from pruning-enabled continual learning in its focus and methodology. While con-

ducting pruning under the continual learning perspective, as explored in Section 2, existing methods simply adopt pruning techniques to manage network complexity during training the model in a continually evolving environment. In contrast, continual pruning specifically refers to the training-free process of pruning that occurs throughout the lifespan of the model. Continual pruning dynamically adjusts which weights are relevant for all the encountered datasets, without the need to save all the past data or reuse them for pruning. Therefore, it ensures enhanced resource efficiency with preserved model's performance over time, regardless of the task or data evolution.

3.2. Weight Stasis in Continual Pruning

Motivated by the conceptual background of calibration-guided pruning strategies (also known as post-training pruning) (Sun et al., 2023; Frantar & Alistarh, 2023), continual pruning takes an advantage of their training-free process to eliminate weight parameters using a particular metric that is often derived from a calibration dataset. However, directly adopting their techniques is not desirable under continual model adaptation. The pruned weights already zeroed-out using their criteria (or threshold) remain consistently unchanged, which results in becoming unresponsive when transitioning the model from one dataset to another, as illustrated in Figure 2. We call this concept as "weight stasis" (WS). In the following, we provide a mathematical insight into weight stasis observed from calibration-guided pruning strategies.

We consider the importance of weights, $\mathbf{W}_i^* = |\mathbf{W}_i \cdot \mathcal{R}_i|$ as the basis for the pruning process, where $\mathcal{R}_i$ is scaling or ranking the weights $\mathbf{W}_i$ with some criteria. For a given

Table 1: Weight Stasis and Forgetting problems on pruning techniques in a continual setting, tested with unstructured pruning with 50% sparsity ratio on LLaMA-7B. A-BWT is average of backward transfer with all incrementa dataset permutations. S-INIT is sequential initialization of weights.

METHOD	S-INIT	PROBLEM	A-BWT
MAGNITUDE	-	WEIGHT STASIS	WS
SPARSE GPT	✗	FORGETTING	0.591
	✓	WEIGHT STASIS	WS
WANDA	✗	FORGETTING	0.569
	✓	WEIGHT STASIS	WS

dataset i, mask $\mathcal{M}_i$, and $\mathcal{I}$ as indicator function, the pruned weight matrix $\mathbf{W}_i^p$ is obtained as follows:

$$\mathcal{M}_i = \mathcal{I}(\mathbf{W}_i^* < \mathcal{T}_s) = \begin{cases} 0 & \text{if } \mathbf{w}_i^* < \mathcal{T}_s, \mathbf{w}_i^* \in \mathbf{W}_i^* \\ 1 & \text{otherwise.} \end{cases},$$

$$\mathbf{W}_i^p = \mathbf{W}_i \cdot \mathcal{M}_i. \tag{1}$$

To achieve a specific sparsity ratio s, the threshold $\mathcal{T}_s$ value is found automatically as detailed in Appendix D.

Upon transitioning from dataset i to dataset $i + 1$, the pruning mechanism starts with $\mathbf{W}_{i+1} = \mathbf{W}_i^p$ as the initial weight matrix. This initial matrix already has a set of pruned (zeroed-out) weights. Following Equation 1, mask $\mathcal{M}_{i+1}$ is derived as

$$\begin{aligned} \mathcal{M}_{i+1} &= \mathcal{I}\left(\mathbf{W}_{i+1}^* < \mathcal{T}_s\right) \\ &= \mathcal{I}(|\mathbf{W}_{i+1} \cdot \mathcal{R}_{i+1}| < \mathcal{T}_s) \\ &= \mathcal{I}(|(\mathbf{W}_i \cdot \mathcal{M}_i) \cdot \mathcal{R}_{i+1}| < \mathcal{T}_s) \\ &= \mathcal{I}(|\mathbf{W}_i \cdot \mathcal{R}_{i+1}| \cdot |\mathcal{M}_i| < \mathcal{T}_s) \\ &= \begin{cases} 0 & \text{if } m < \mathcal{T}_s, m \in |\mathbf{W}_i \cdot \mathcal{R}_{i+1}| \cdot |\mathcal{M}_i| \\ 1 & \text{otherwise.} \end{cases}, \\ &= \mathcal{M}_i, \end{aligned} \tag{2}$$

$$\mathbf{W}_{i+2} = \mathbf{W}_{i+1} = \mathbf{W}_i \cdot \mathcal{M}_i.$$

Since the masked weights in w_{i+1}^* are zero for the previously pruned weights ($\mathbf{W}_i$), these weights will again fall below the new threshold and remain pruned, regardless of the new dataset $i + 1$. Therefore, $\mathbf{W}_{i+1}^p$ is being equal to $\mathbf{W}_i^p$ freezing these weights when transitioning to a new dataset. leading to the phenomenon of "weight stasis". This limits the model's ability to adapt to new datasets while adopting calibration-guided pruning strategies to this domain. This is also observed from the experimental results in Table 1.

3.3. Forgetting in Continual Pruning

Easing weight stasis by non-sequential weight initialization, aforementioned calibration-based post-training pruning methods can be directly applied to continual settings. However, they pose a significant challenge known as catastrophic forgetting as depicted in Figure 2. This phenomenon is observed when pruned model weights are updated using a new calibration dataset, which deteriorates the performance on previously encountered datasets or tasks.

Table 1 demonstrates previously discussed challenges in continual pruning, evaluating exiting methods using LLaMA-7B with 50% sparsity. With sequential initialization, all baselines encounter WS, which means pruned weights remain the same. In addition, higher A-BWT of these methods implies that they have a forgetting issue, where the performance of the on the previously encountered dataset is increasing when pruning on the current dataset. Note that COPAL reports A-BWT as 0.016 for the same evaluation as shown in Table 2.

4. Methodology

We propose a framework for continual neural network pruning based on sensitivity analysis to overcome two challenges of continual pruning: weight stasis and forgetting. The overview of these problems and our proposed COPAL framework is shown in Figure 2.

4.1. Theoretical Framework for Continual Sensitivity Analysis

In a standard layer of a neural network, the output $\mathbf{y}_j^i$ is computed as $\mathbf{y}_j^i = f(\mathbf{x}_j^i, \mathbf{W})$, where f is the layer's function, $\mathbf{W}$ is the weight matrix of the base model, and $\mathbf{x}_j^i$ is the input feature vector based on j-th input in dataset i. Given layer operations, we derive the sensitivity of $\mathbf{y}$ to infinitesimal changes $d\mathbf{x}_j^i$ and $d\mathbf{W}$ in $\mathbf{x}_j^i$ and $\mathbf{W}$, yielding,

$$d\mathbf{y}_j^i = \frac{\partial f}{\partial \mathbf{x}_j^i} d\mathbf{x}_j^i + \frac{\partial f}{\partial \mathbf{W}} d\mathbf{W}. \tag{3}$$

The consideration of infinitesimal variations of $d\mathbf{x}_j^i$ and $d\mathbf{W}$ is analytically intractable. Therefore, we consider their approximate as $\Delta \mathbf{W}$ and $\Delta \mathbf{x}_j^i$, both of which are assumed to be sufficiently small. Equation 3 is now converted to the measure of sensitivity terms

$$\begin{aligned} S_{\mathbf{W}}^{ij} &= \frac{\partial f}{\partial \mathbf{W}} \times \Delta \mathbf{W}, \\ S_{\mathbf{x}}^{ij} &= \frac{\partial f}{\partial \mathbf{x}_j^i} \times \Delta \mathbf{x}_j^i. \end{aligned} \tag{4}$$

$S_{\mathbf{W}}^{ij}$ serves as the theoretical measure of sensitivity. Similarly, $S_{\mathbf{x}}^{ij}$ is the sensitivity measure of $\mathbf{y}_j^i$ with respect to

x. In practice, directly calculating these derivative is often challenging or infeasible, so we use the finite difference approximation of $\frac{\partial f}{\partial \mathbf{W}}$ and $\frac{\partial f}{\partial \mathbf{x}_j^i}$. With these, Equation 4 can be revised as

$$
\begin{aligned}
S_\mathbf{W}^{ij} &= f(\mathbf{W} + \Delta\mathbf{W}, \mathbf{x}_j^i) - \mathbf{y}_j^i, \\
S_\mathbf{x}^{ij} &= f(\mathbf{W}, \mathbf{x}_j^i + \Delta\mathbf{x}_j^i) - \mathbf{y}_j^i.
\end{aligned}
\tag{5}
$$

These terms respectively form the impact of perturbations in $\mathbf{W}$ with $\Delta\mathbf{W}$ and similarly those in $\mathbf{x}$ with $\Delta\mathbf{x}$ on $\mathbf{y}$, which are effectively converted from a theoretical concept to a practical metric. With the combined sensitivity measures, $d\mathbf{y}_j^i$ is reformulated as

$$
d\mathbf{y}_j^i = S_\mathbf{W}^{ij} + S_\mathbf{x}^{ij}.
\tag{6}
$$

More detailed proof is available in Appendix A.

4.2. Identification of Crucial Weights

The loss function on the j-th input vector in dataset i is defined as square of Euclidean norm of $d\mathbf{y}_j^i$ as follows:

$$
\mathcal{L}_j^i = \left\| d\mathbf{y}_j^i \right\|_2^2.
\tag{7}
$$

Employing Equation 3, $\mathcal{L}_j^i$ can be rewritten as

$$
\mathcal{L}_j^i = \left\| \frac{\partial f}{\partial \mathbf{x}_j^i} d\mathbf{x}_j^i + \frac{\partial f}{\partial \mathbf{W}} d\mathbf{W} \right\|_2^2.
\tag{8}
$$

Then, to determine the optimal perturbations in $\mathbf{W}$ (weights to be pruned) that would minimize $d\mathbf{y}$, we compute the gradient of $\mathcal{L}_j^i$ with respect to $d\mathbf{W}$ as follows:

$$
\nabla_{d\mathbf{W}} \mathcal{L}_j^i = 2 d\mathbf{y}_j^i \frac{\partial f}{\partial \mathbf{W}}.
\tag{9}
$$

Equation 9 is proved in Appendix B.

By finding $\nabla_{d\mathbf{W}} \mathcal{L}_j^i$, we can identify the output sensitivity's ($d\mathbf{y}_j^i$) loss function variation with respect to weight sensitivity ($d\mathbf{W}$).[1]

We further introduce $\nabla'_{d\mathbf{W}} \mathcal{L}^k$ to capture the sum of the absolute values of the individual gradients until dataset k.

$$
\begin{aligned}
\nabla'_{d\mathbf{W}} \mathcal{L}^k &= \sum_{i=0}^{k} \sum_j \left| \nabla_{d\mathbf{W}} \mathcal{L}_j^i \right| \\
&= \sum_j 2 \left| d\mathbf{y}_j^k \frac{\partial f}{\partial \mathbf{W}} \right| + \sum_{i=0:k-1} \sum_j 2 \left| d\mathbf{y}_j^i \frac{\partial f}{\partial \mathbf{W}} \right| \\
&= \sum_j 2 \left| d\mathbf{y}_j^k \right| \left| \frac{\partial f}{\partial \mathbf{W}} \right| + \nabla'_{d\mathbf{W}} \mathcal{L}^{k-1} \\
&= \nabla'_{d\mathbf{W}} \tilde{\mathcal{L}}^k + \nabla'_{d\mathbf{W}} \mathcal{L}^{k-1},
\end{aligned}
\tag{10}
$$

---

[1] To find the partial differential $\frac{\partial f}{\partial \mathbf{W}}$, for a linear layer where $\mathbf{y}_j^i = \mathbf{W} x_j^i$, $\frac{\partial f}{\partial \mathbf{W}} = \mathbf{x}_j^i$ and for generic purpose, it can be approximated as $S_\mathbf{W}^{ij} \Delta\mathbf{W}^+$. Please check Appendix C.

where $\tilde{\mathcal{L}}^k$ is loss function for only dataset k. This metric is particularly insightful for understanding how sensitive the loss is to each individual sample, without regard to the direction of that sensitivity (positive or negative). By focusing on the magnitude rather than the direction, we aim to offer a more nuanced view of the model's robustness to perturbations in $\mathbf{W}$ and $\mathbf{x}$.

Finally, we use magnitude of directional derivative D of $\mathcal{L}_j^i$ along $\mathbf{W}$ as our measure of importance weights denoted by $\mathbf{W}_k^*$.

$$
\begin{aligned}
\mathbf{W}_k^* &= \sum_{i=0:k} \sum_j \left| D_\mathbf{W} \mathcal{L}_j^i \right| \\
&= \sum_{i=0:k} \sum_j \left| \mathbf{W} \cdot \nabla_{d\mathbf{W}} \mathcal{L}_j^i \right| \\
&= |\mathbf{W}| \cdot \sum_{i=0:k} \sum_j \left| \nabla_{d\mathbf{W}} \mathcal{L}_j^i \right| \\
&= |\mathbf{W}| \cdot \nabla'_{d\mathbf{W}} \mathcal{L}^k \\
&= |\mathbf{W}| \cdot (\nabla'_{d\mathbf{W}} \tilde{\mathcal{L}}^k + \nabla'_{d\mathbf{W}} \mathcal{L}_{k-1}) \\
&= \sum_j \left| \mathbf{W} \cdot \nabla_{d\mathbf{W}} \mathcal{L}_j^k \right| + \mathbf{W}_{k-1}^*.
\end{aligned}
\tag{11}
$$

This $\mathbf{W}_k^*$ serves as a measure of the importance or sensitivity of the weights $\mathbf{W}$ in affecting the loss function. A high $\mathbf{W}_k^*$ value indicates that the loss function is highly sensitive to changes in the weights along the direction of $\mathbf{W}$, making those weights particularly crucial for the model's performance.

4.3. Algorithm

We present the COPAL algorithm in Algorithm 1. For each dataset in the sequence $\mathcal{D}_i$, we calculate the sensitivity in output $d\mathbf{y}_j^i$ using sensitivity measures $S_\mathbf{W}^{ij}, S_\mathbf{x}^{ij}$. The gradient of the loss function $\nabla_{d\mathbf{W}} \mathcal{L}_j^i$ is computed using the output sensitivity $d\mathbf{y}_j^i$ and the differential $\frac{\partial f}{\partial \mathbf{W}}$. We update the importance of weights W_i^* using our sensitivity metric. A threshold $\mathcal{T}_s$ is determined based on the desired level of sparsity, and a pruning mask $\mathcal{M}_i$ is created to selectively prune weights that fall below this threshold. As the algorithm progresses through the dataset sequence, it continually updates the importance of weights and prunes accordingly, resulting in a model that retains only the most critical weights for its function across all datasets.

5. Experiments

5.1. Experimental Setting

Setup. Our method and the baseline models are implemented using the Pytorch framework (Paszke et al., 2019), and we employ the Hugging Face Transformers library (Wolf et al., 2019), for managing models and datasets.

Algorithm 1 COPAL

> **Input:** Weights $\mathbf{W}$,   Sparsity ratio s,
>   Datasets $\mathcal{D}_1, \cdots, \mathcal{D}_k$,
>   using j-th input data from dataset i the input
>   and output feature of the layer f are $(\mathbf{x}_j^i, \mathbf{y}_j^i)$
> **Output**: Pruned weights $\mathbf{W}_k^p$
> **Initialize**: $\mathbf{W}_0^* = 0$
> **for** i = 1:k **do**
> $\quad S_{\mathbf{W}}^{ij} \leftarrow f(\mathbf{W} + \Delta\mathbf{W}, \mathbf{x}_j^i) - \mathbf{y}_j^i$
> $\quad S_{\mathbf{x}}^{ij} \leftarrow f(\mathbf{W}, \mathbf{x}_j^i + \Delta\mathbf{x}_j^i) - \mathbf{y}_j^i$
> $\quad d\mathbf{y}_j^i \leftarrow S_{\mathbf{W}}^{ij} + S_{\mathbf{x}}^{ij}$
> $\quad \frac{\partial f}{\partial \mathbf{W}} \leftarrow \begin{cases} S_{\mathbf{W}}^{ij}\Delta\mathbf{W}^+, & \text{if } f \text{ is non-linear layer} \\ x, & \text{if } f \text{ is linear layer} \end{cases}$
> $\quad \nabla_{d\mathbf{W}}\mathcal{L}_j^i \leftarrow 2d\mathbf{y}_j^i \frac{\partial f}{\partial \mathbf{W}}$
> $\quad \mathbf{W}_i^* \leftarrow \sum_j \left| \mathbf{W} \cdot \nabla_{d\mathbf{W}}\mathcal{L}_j^i \right| + \mathbf{W}_{i-1}^*$
> $\quad N \leftarrow$ total number of elements in $\mathbf{W}_i^*$
> $\quad \mathcal{T}_s \leftarrow$ Sorted $\mathbf{W}_i^* [\lceil(1 - s/100) \times N\rceil]$
> $\quad \mathcal{M}_i \leftarrow \begin{cases} 0, & \text{if } w^i < \mathcal{T}_s, \ \ w^i \in \mathbf{W}_i^* \\ 1, & \text{otherwise} \end{cases}$
> $\quad \mathbf{W}_i^p \leftarrow \mathbf{W}_i \cdot \mathcal{M}_i$
> **end for**
> **Return**: $\mathbf{W}_k^p$

Our pruning experiments utilize a single NVIDIA A100 GPU with 80 GB of memory.

Following the methodologies in (Yao et al., 2022; Frantar et al., 2022; Sun et al., 2023; Frantar & Alistarh, 2023), we sequentially sparsify Transformer layers, significantly reducing memory requirements. Our experiments are conducted in a single step, without fine-tuning, similar to recent studies on post-training quantization and pruning of GPT-scale models, as seen in (Frantar et al., 2022; Yao et al., 2022; Dettmers et al., 2022; Sun et al., 2023; Frantar & Alistarh, 2023). For calibration, we use 16 segments of 2048 tokens each, randomly chosen from the first shard of each dataset.

Models, Datasets. In this paper, we conducted a comprehensive series of experiments utilizing the LLaMA model family, which includes models with 7B, 13B, 30B, and 65B parameters, as recently introduced by Touvron et al. (2023). This family of models represents a significant advancement in language model capabilities, offering a range of scales to cater to different computational requirements and performance benchmarks.

Our experimentation focused on three of the most commonly used language datasets in the field: Wikitext-2 (Merity et al., 2016), the Penn Treebank (PTB) (Marcus et al., 1993), and the Colossal Clean Crawled Corpus (C4) (Raffel et al., 2020). Wikitext-2, introduced by Merity et al. (2016), is renowned for its collection of high-quality, well-structured textual data, predominantly comprising Wikipedia articles. The PTB dataset is a widely-used resource for syntactic analysis (Marcus et al., 1993). Lastly, the C4 dataset, part of the T5 model training corpus, provides a broad and diverse range of internet text, essential for evaluating model performance across various linguistic contexts (Raffel et al., 2020).

Baselines. In our comparison, we evaluate the standard magnitude pruning approach, as established by Zhu & Gupta (2017), alongside the more recent developments in post-training pruning works WANDA (Sun et al., 2023) and SparseGPT (Frantar & Alistarh, 2023). Each of these techniques is applied in a layer-wise manner, facilitating scalability even in the context of exceptionally large models. Magnitude pruning, a method detailed by Zhu & Gupta (2017), effectively compresses models by removing weights with the smallest absolute values, which are deemed least impactful on the network's output. In contrast, SparseGPT, introduced by Frantar & Alistarh (2023), integrates sparsity into the post-training process of transformer-based models. It often employs methods such as utilizing the Hessian matrix to identify weights that can be pruned with minimal loss in performance, effectively making the model sparse from the outset. Finally, WANDA, as proposed by Sun et al. (2023), does pruning by analyzing weight and activation distributions. This technique identifies and prunes network segments that minimally contribute to outputs.

In Table 2, baseline models employ global initialization due to weight stasis in sequential setups, as detailed in Section 3.2 and Table 1. Global initialization also leads to better average perplexity compared to sequential initialization.

Sparsity. In our evaluation, we specifically target the linear layers of large language models (LLMs), excluding the initial embedding layer and the final classification head. These linear layers constitute approximately 99% of the total parameters in LLMs (Sun et al., 2023), making them the primary focus for pruning. Uniform sparsity is maintained across all linear layers for consistency. We explore three distinct sparsity types: unstructured sparsity, and semi-structured sparsities with 4:8 and 2:4 configurations (Sun et al., 2023; Frantar & Alistarh, 2023). This allows for a more comprehensive comparison and understanding of the impacts of different sparsity structures on LLM performance.

Evaluation Metrics. In this study, we focus on the impact of dataset sequence permutations (Π) on language model performance in continual learning scenarios. We assess this through two primary metrics: perplexity and Backward Transfer (BWT).

Our primary metric for evaluating the efficacy of continual pruning methods in language models is perplexity, a

well-established and robust metric (Yao et al., 2022; Frantar et al., 2022; Sun et al., 2023; Frantar & Alistarh, 2023; Dettmers et al., 2022). Perplexity is calculated for each dataset permutation. We determine both the mean and maximum perplexity values across all permutations. The mean perplexity $\mathbf{P}_\pi^{i,j}$ for each example e in a dataset j after pruning on dataset i, with a permutation of datasets ordered π is defined as:

$$\mathbf{P}_\pi^{i,j} = \frac{1}{M} \sum_{e=1}^{M} \exp\left(-\sum_{c=1}^{N_e} \log(p(\mathbf{w}_c^e | \mathbf{w}_{1:c}^e))\right) \quad (12)$$

where M is the number of examples, N_e is the number of words in example e, and $p(\mathbf{w}_c^e | \mathbf{w}_{1:c}^e)$ represents the probability of the cth word in example e, $\mathbf{w}_{1:c}^e$ is the context or the past words in the example. The equations for the mean and maximum perplexity across permutations are:

$$\mathbf{P}_{avg} = \frac{1}{|\Pi|} \sum_{\pi \in \Pi, i,j \in D} \left(\mathbf{P}_\pi^{i,j}\right),$$
$$\mathbf{P}_{max} = \max_{\pi \in \Pi, i,j \in D} \left(\mathbf{P}_\pi^{i,j}\right), \quad (13)$$

where $\mathbf{P}_\pi$ is the mean perplexity for the permutation π and D is the total number of Datasets,

In addition to perplexity, we incorporate Backward Transfer (BWT) as a metric, crucial in continual learning scenarios. BWT, which assesses a model's ability to retain knowledge from previous tasks, is a crucial metric in continual learning (Lopez-Paz & Ranzato, 2017; Kirkpatrick et al., 2017).

BWT for a particular permutation π is defined as the average decrease in performance of previously learned tasks after learning a new task in the permutation. It is calculated as:

$$\text{BWT}_\pi^{i,j} = \left(\mathbf{P}_\pi^{i,j} - \mathbf{P}_\pi^{j,j}\right) \quad (14)$$

where $\mathbf{P}_{d,d}$ is the performance on dataset d immediately after learning it, and $\mathbf{P}_{d,D}$ is the performance on the same task after learning all D datasets.

For each dataset permutation, we calculate BWT and then determine both the average and maximum BWT values across all permutations, similar to Equation 13. This approach ensures a comprehensive evaluation of model performance across diverse continual pruning conditions.

5.2. Results

The performance of the dense (no pruning) models sets a fundamental baseline, indicating the optimal performance without any compromise due to pruning. In the LLaMA-7B model, the dense configuration yields an average PPL of 7.714. This benchmark is slightly lower in the LLaMA-30B and LLaMA-65B models, with average PPLs of 6.131 and 6.139, respectively. These variations suggest a nuanced

impact of model scale on language processing capabilities, with larger models inherently capable of better performance pre-pruning.

Unstructured Continual Pruning. In Table 2, the LLaMA-7B model under unstructured pruning, COPAL exhibits a pronounced superiority. It drastically lowers average BWT to 0.016, compared to best performing baseline Wanda's 0.569, demonstrating a 97.2% improvement. This indicates a highly effective pruning process with minimal knowledge loss. In Table 3, COPAL's PPL performance on individual datasets like Wikitext2 and PTB shows PPL performance improvement of 7.144 (compared to best performing baseline Wanda's 7.208) and 12.567 (compared to best performing baseline SparseGPT's 13.141), respectively at 50% sparsity ratio and still shows comparable performance to the base model. For the LLaMA-13B model, COPAL reduces average BWT to 0.029 and average PPL to 8.354, indicating substantial improvements in efficiency and comprehension. This consistency across diverse datasets underlines the method's robustness and adaptability.

Shifting to the LLaMA-30B model with unstructured pruning, COPAL again leads in performance metrics. It achieves a notable 98.2% improvement in average BWT over SparseGPT and a minimal PPL performance drop, for instance, from 8.159 (Dense) to 9.077 on PTB. These results highlight COPAL's scalability and its effectiveness in handling moderately larger models without compromising on the pruning efficiency or language understanding.

In the LLaMA-65B model, COPAL's efficacy is further amplified. The method achieves nearly negligible average BWT (0.001), a 99.7% improvement over SparseGPT. On the C4 dataset, COPAL significantly achieves PPL performance improvement from 8.878 (Dense) to 8.696, showcasing its exceptional capability in large-scale models. This performance indicates an continual pruning strategy that not only reduces model size but also preserves, and in some aspects, enhances model comprehension abilities.

Semi-structured N:M Continual Pruning. In the realm of semi-structured N:M continual pruning, the COPAL technique stands out for its effectiveness across various pruning configurations, notably within the context of the LLaMA-65B model. When applied to a 2:4 pruning pattern, COPAL achieves a 98.2% improvement in average Backward Weight Transfer (BWT), reducing it to 0.038 compared to SparseGPT's 2.128. Additionally, it enhances the average Perplexity (PPL) by 1.8%, bringing it down to 9.161 from SparseGPT's 9.333. The proficiency of COPAL is further evidenced in the 4:8 pruning pattern, where it markedly reduces average BWT by 99.3% to 0.019, compared to SparseGPT's 0.907, and achieves a 0.9% improvement in average PPL, lowering it to 8.291 from 8.363. In the 2:4 pruning configuration for LLaMA-13B, COPAL achieves

Table 2: Results of continual pruning on wikitext2, ptb, c4 datasets with all permutations. Unstructured with 50% sparsity ratio, Semi structured 2:4 and 4:8 pruning. A- indicates average of a metric, M- indicates maximum of a metric.

	LLAMA-7B				LLAMA-30B			
	A-BWT	M-BWT	A-PPL	M-PPL	A-BWT	M-BWT	A-PPL	M-PPL
DENSE (NO PRUNING)	-	-	7.714	10.120	-	-	6.131	8.159
UNSTRUCTURED								
MAGNITUDE	WS	WS	30.246	49.670	WS	WS	10.958	14.638
SPARSEGPT	0.591	0.690	10.166	13.253	0.395	0.730	7.452	9.520
WANDA	0.569	1.072	9.991	13.626	0.132	0.192	7.261	9.231
COPAL (OURS)	**0.016**	**0.032**	**9.728**	**12.585**	**0.007**	**0.025**	**7.240**	**9.081**
SEMI STRUCTURED 2:4								
MAGNITUDE	WS	WS	131.653	303.710	WS	WS	13.757	19.139
SPARSEGPT	4.365	6.391	15.744	21.771	1.436	3.015	9.592	12.657
WANDA	1.667	3.192	16.154	23.266	0.493	1.079	9.363	12.183
COPAL (OURS)	**0.009**	**0.075**	**15.335**	**21.159**	**0.036**	**0.100**	**9.274**	**11.478**
SEMI STRUCTURED 4:8								
MAGNITUDE	WS	WS	32.105	56.652	WS	WS	12.998	16.881
SPARSEGPT	1.929	3.045	11.924	15.884	0.838	1.670	8.351	10.790
WANDA	0.771	1.645	11.929	16.631	0.231	0.486	8.094	10.353
COPAL (OURS)	**0.038**	**0.075**	**11.734**	**15.532**	**0.012**	**0.050**	**8.032**	**9.982**

	LLAMA-13B				LLAMA-65B			
	A-BWT	M-BWT	A-PPL	M-PPL	A-BWT	M-BWT	A-PPL	M-PPL
DENSE (NO PRUNING)	-	-	6.990	9.081	-	-	6.139	8.878
UNSTRUCTURED								
MAGNITUDE	WS	WS	28.935	41.368	WS	WS	9.399	13.701
SPARSEGPT	0.606	1.123	8.467	11.094	0.334	0.564	7.235	9.874
WANDA	0.203	0.298	8.570	10.896	0.172	0.628	7.584	11.354
COPAL (OURS)	**0.029**	**0.078**	**8.354**	**10.818**	**0.001**	**0.208**	**6.791**	**8.839**
SEMI-STRUCTURED (2:4)								
MAGNITUDE	WS	WS	28.702	44.072	WS	WS	10.544	14.704
SPARSEGPT	2.670	5.978	11.970	17.386	2.128	6.979	9.333	16.037
WANDA	0.871	1.698	13.209	18.920	0.184	0.389	9.230	12.665
COPAL (OURS)	**-0.007**	**0.340**	**11.455**	**17.155**	**0.038**	**0.258**	**9.161**	**11.233**
SEMI-STRUCTURED (4:8)								
MAGNITUDE	WS	WS	20.476	29.055	WS	WS	9.247	12.568
SPARSEGPT	1.248	2.905	9.773	13.456	0.907	2.992	8.363	13.358
WANDA	0.375	0.793	9.946	13.281	0.172	0.610	8.315	11.746
COPAL (OURS)	**-0.019**	**0.160**	**9.402**	**12.338**	**0.019**	**0.260**	**8.291**	**10.807**

BWT of -0.007 and PPL of 11.455, illustrating a marked enhancement in the model's pruning efficacy. These outcomes underscore COPAL's robust adaptability and efficiency in managing semi-structured pruning, particularly within large-scale models like LLaMA-65B, where balancing efficiency and performance is paramount.

Turning attention to the LLaMA-7B and LLaMA-30B models, COPAL's superior performance persists, showcasing a consistent pattern of effectiveness in both 2:4 and 4:8 pruning patterns. For the LLaMA-7B model, COPAL not

only significantly lowers the average BWT to 0.009, marking a 99.4% improvement over next best baseline Wanda, but also decreases average PPL to 15.335, illustrating a 5.1% improvement. In the 4:8 pruning pattern, it continues to demonstrate efficacy by reducing average BWT by 95.0% and improving average PPL by 1.6%, compared to SparseGPT. The LLaMA-30B model similarly benefits from COPAL's application, yielding substantial improvements in BWT and PPL across both pruning patterns, further reinforcing COPAL's capability to effectively navigate the trade-offs between both the metrics.

Table 3: Results of per dataset performance continual pruning on wikitext2, ptb, c4 datasets with all incremental permutations on LLaMA-7B,13B,30B,65B. Unstructured pruning with 50% sparsity ratio. Overall Standard deviations in Appendix E

|	WIKITEXT2		PTB		C4	
	BWT	PPL	BWT	PPL	BWT	PPL
LLAMA-7B						
DENSE (NO PRUNING)	-	5.677 ± 0.000	-	10.120 ± 0.000	-	7.344 ± 0.000
MAGNITUDE	WS	17.288 ± 0.000	WS	49.671 ± 0.000	WS	23.778 ± 0.000
SPARSEGPT	0.687 ± 0.003	7.352 ± 0.343	0.506 ± 0.083	13.141 ± 0.260	0.580 ± 0.146	10.005 ± 0.308
WANDA	0.192 ± 0.077	7.208 ± 0.111	1.288 ± 0.378	13.286 ± 0.697	0.227 ± 0.032	9.479 ± 0.116
COPAL (OURS)	**0.017 ± 0.011**	**7.144 ± 0.013**	**0.027 ± 0.021**	**12.567 ± 0.024**	**0.005 ± 0.003**	**9.475 ± 0.004**
LLAMA-13B						
DENSE (NO PRUNING)	-	5.091 ± 0.000	-	9.081 ± 0.000	-	6.798 ± 0.000
MAGNITUDE	WS	20.215 ± 0.000	WS	41.368 ± 0.000	WS	25.222 ± 0.000
SPARSEGPT	0.465 ± 0.093	6.301 ± 0.241	1.121 ± 0.002	10.532 ± 0.561	0.233 ± 0.049	8.568 ± 0.121
WANDA	0.165 ± 0.040	6.138 ± 0.087	0.285 ± 0.013	10.663 ± 0.143	0.159 ± 0.028	8.262 ± 0.082
COPAL (OURS)	**0.052 ± 0.022**	**6.096 ± 0.032**	**0.011 ± 0.018**	**10.472 ± 0.020**	**0.023 ± 0.026**	**8.176 ± 0.019**
LLAMA-30B						
DENSE (NO PRUNING)	-	4.101 ± 0.000	-	8.159 ± 0.000	-	6.131 ± 0.000
MAGNITUDE	WS	7.542 ± 0.000	WS	14.638 ± 0.000	WS	10.696 ± 0.000
SPARSEGPT	0.291 ± 0.039	5.483 ± 0.148	0.683 ± 0.048	9.131 ± 0.343	0.211 ± 0.007	7.743 ± 0.105
WANDA	0.092 ± 0.033	5.238 ± 0.052	0.149 ± 0.025	9.132 ± 0.077	0.156 ± 0.036	7.414 ± 0.082
COPAL (OURS)	**0.012 ± 0.010**	**5.229 ± 0.009**	**0.005 ± 0.005**	**9.077 ± 0.004**	**0.005 ± 0.005**	**7.415 ± 0.004**
LLAMA-65B						
DENSE (NO PRUNING)	-	3.562 ± 0.000	-	8.878 ± 0.000	-	5.978 ± 0.000
MAGNITUDE	WS	5.901 ± 0.000	WS	13.701 ± 0.000	WS	8.595 ± 0.000
SPARSEGPT	0.263 ± 0.047	4.791 ± 0.136	0.561 ± 0.003	9.591 ± 0.281	0.178 ± 0.165	7.323 ± 0.147
WANDA	0.075 ± 0.003	4.616 ± 0.038	0.440 ± 0.188	10.946 ± 0.257	0.001 ± 0.038	7.189 ± 0.027
COPAL (OURS)	**-0.004 ± 0.026**	**4.639 ± 0.018**	**0.066 ± 0.093**	**8.696 ± 0.065**	**-0.059 ± 0.035**	**7.036 ± 0.033**
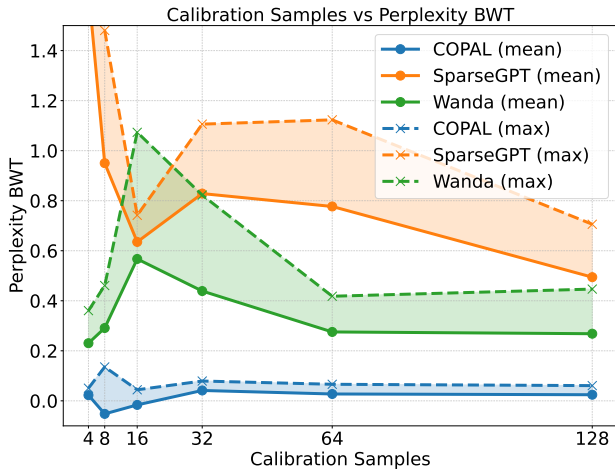
Figure 3: Motivation: average (mean) and worst (max) case scenarios of backward transfer in perplexity with increase in samples in unstructured continual pruning of LLaMA-7B with 50% sparsity

5.3. Discussions

Ablation of Sparsity Ratio. As shown in Figure 1, with increased sparsity ratios the effect of forgetting or poor performance on previous tasks is observed with other pruning methods. It demonstrates that our approach is better formu-

lated to reduce the forgetting or maintain performance on previously observed tasks in continual pruning setting.

Ablation of Different Sample Sizes. In post-training pruning usually contains a calibration data that contains a set of calibration samples. As shown in Figure 3, COPAL is sample efficient and the average and worst case are minimal and the difference between them is also minimal. We observe that with as little as 16 samples our approach is able to perform better and stable performance even with more samples.

6. Conclusion

In this paper, we introduced the concept of *continual pruning* that addresses pruning under a continual model adaptation setting, bypassing the requirement for model retraining. Two common problems observed in continual pruning, weight stasis and forgetting, were addressed through our proposed COPAL framework. We presented mathematical background, formulations, and full derivations of COPAL. Throughout the comprehensive evaluation, COPAL significantly outperformed existing methods in both Backward Transfer (BWT) reduction and Perplexity (PPL) performance, showcasing remarkable consistency and adaptability across various pruning structures, datasets and models, including LLaMA-7B, 13B, 30B, 65B.

Impact Statement

Firstly, it's crucial to be aware of the potential for democratizing AI technology. COPAL's ability to reduce computational demands means that powerful LLMs can become more accessible to a wider range of users and organizations, potentially driving innovation across various sectors. However, this democratization also brings with it the responsibility to ensure that the models remain fair and unbiased, particularly as they adapt to changing data streams.

We must also consider the ethical implications of dynamic pruning. The adaptability of these models might lead to challenges in maintaining transparency and explainability, which are critical for user trust and accountability. As we move forward with this technology, it's essential to balance the benefits of efficiency and adaptability with the ongoing commitment to ethical AI practices and responsible usage.

References

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901, 2020.

Dekhovich, A., Tax, D. M., Sluiter, M. H., and Bessa, M. A. Continual prune-and-select: class-incremental learning with specialized subnetworks. *Applied Intelligence*, pp. 1–16, 2023.

Dettmers, T., Lewis, M., Belkada, Y., and Zettlemoyer, L. Llm. int8 (): 8-bit matrix multiplication for transformers at scale. *arXiv preprint arXiv:2208.07339*, 2022.

Frankle, J. and Carbin, M. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

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

Frantar, E., Ashkboos, S., Hoefler, T., and Alistarh, D. Gptq: Accurate post-training quantization for generative pretrained transformers. *arXiv preprint arXiv:2210.17323*, 2022.

Gururangan, S., Marasović, A., Swayamdipta, S., Lo, K., Beltagy, I., Downey, D., and Smith, N. A. Don't stop pretraining: Adapt language models to domains and tasks. *arXiv preprint arXiv:2004.10964*, 2020.

Han, S., Mao, H., and Dally, W. J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

He, Y., Zhang, X., and Sun, J. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE international conference on computer vision*, pp. 1389–1397, 2017.

He, Y., Kang, G., Dong, X., Fu, Y., and Yang, Y. Soft filter pruning for accelerating deep convolutional neural networks. *arXiv preprint arXiv:1808.06866*, 2018.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

LeCun, Y., Denker, J., and Solla, S. Optimal brain damage. In Touretzky, D. (ed.), *Advances in Neural Information Processing Systems*, volume 2. Morgan-Kaufmann, 1989. URL https://proceedings.neurips.cc/paper_files/paper/1989/file/6c9882bbac1c7093bd25041881277658-Paper.pdf.

Lee, N., Ajanthan, T., and Torr, P. H. Snip: Single-shot network pruning based on connection sensitivity. *arXiv preprint arXiv:1810.02340*, 2018.

Liu, J., Xu, Z., Shi, R., Cheung, R. C., and So, H. K. Dynamic sparse training: Find efficient sparse network from scratch with trainable masked layers. *arXiv preprint arXiv:2005.06870*, 2020.

Liu, Z., Li, J., Shen, Z., Huang, G., Yan, S., and Zhang, C. Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE international conference on computer vision*, pp. 2736–2744, 2017.

Lopez-Paz, D. and Ranzato, M. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30, 2017.

Luo, J.-H., Wu, J., and Lin, W. Thinet: A filter level pruning method for deep neural network compression. In *Proceedings of the IEEE international conference on computer vision*, pp. 5058–5066, 2017.

Marcus, M., Santorini, B., and Marcinkiewicz, M. A. Building a large annotated corpus of english: The penn treebank. 1993.

McCloskey, M. and Cohen, N. J. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pp. 109–165. Elsevier, 1989.

Merity, S., Xiong, C., Bradbury, J., and Socher, R. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551, 2020.

Rolnick, D., Ahuja, A., Schwarz, J., Lillicrap, T., and Wayne, G. Experience replay for continual learning. *Advances in Neural Information Processing Systems*, 32, 2019.

Rusu, A. A., Rabinowitz, N. C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., and Hadsell, R. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

Sun, M., Liu, Z., Bair, A., and Kolter, J. Z. A simple and effective pruning approach for large language models. *arXiv preprint arXiv:2306.11695*, 2023.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Wang, C., Zhang, G., and Grosse, R. Picking winning tickets before training by preserving gradient flow. *arXiv preprint arXiv:2002.07376*, 2020.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., et al. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.

Xia, M., Zhong, Z., and Chen, D. Structured pruning learns compact and accurate models. *arXiv preprint arXiv:2204.00408*, 2022.

Yan, Q., Gong, D., Liu, Y., van den Hengel, A., and Shi, J. Q. Learning bayesian sparse networks with full experience replay for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 109–118, 2022.

Yang, Z., Cui, Y., Yao, X., and Wang, S. Gradient-based intra-attention pruning on pre-trained language models. *arXiv preprint arXiv:2212.07634*, 2022.

Yao, Z., Yazdani Aminabadi, R., Zhang, M., Wu, X., Li, C., and He, Y. Zeroquant: Efficient and affordable post-training quantization for large-scale transformers. *Advances in Neural Information Processing Systems*, 35: 27168–27183, 2022.

Yoon, J., Yang, E., Lee, J., and Hwang, S. J. Lifelong learning with dynamically expandable networks. *arXiv preprint arXiv:1708.01547*, 2017.

Yu, R., Li, A., Chen, C.-F., Lai, J.-H., Morariu, V. I., Han, X., Gao, M., Lin, C.-Y., and Davis, L. S. Nisp: Pruning networks using neuron importance score propagation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 9194–9203, 2018.

Zenke, F., Poole, B., and Ganguli, S. Continual learning through synaptic intelligence. In *International conference on machine learning*, pp. 3987–3995. PMLR, 2017.

Zhu, M. and Gupta, S. To prune, or not to prune: exploring the efficacy of pruning for model compression. *arXiv preprint arXiv:1710.01878*, 2017.

A. Theoretical Sensitivity to Practical Metrics

In a general setting, let us consider a function $\mathbf{y}_j^i = f(\mathbf{x}_j^i, \mathbf{W})$ representing the output based on input $\mathbf{x}_j^i$ and parameter $\mathbf{W}$. The theoretical sensitivity $\mathbf{S}_\mathbf{W}^{ij}$ is described by the derivative:

$$\mathbf{S}_\mathbf{W}^{ij} = \frac{\partial f}{\partial \mathbf{W}} \times \Delta \mathbf{W} \tag{15}$$

This derivative represents the instantaneous rate of change of f with respect to $\mathbf{W}$ and serves as the theoretical measure of sensitivity.

Directly calculating this derivative is often challenging or infeasible, especially when $f(\mathbf{x}_j^i, \mathbf{W})$ is complex. Therefore, a common approach is to approximate this derivative using finite differences. The finite difference approximation of $\frac{\partial f}{\partial \mathbf{W}}$ is given by:

$$\frac{\partial f}{\partial \mathbf{W}} \approx \frac{f(\mathbf{x}_j^i, \mathbf{W} + \Delta \mathbf{W}) - f(\mathbf{x}_j^i, \mathbf{W})}{\Delta \mathbf{W}} \tag{16}$$

Using this approximation, a practical sensitivity metric $\mathbf{S}_\mathbf{w}^{ij}$ can be introduced as follows:

$$\begin{aligned}
\mathbf{S}_\mathbf{w}^{ij} &= \frac{f(\mathbf{x}_j^i, \mathbf{W} + \Delta \mathbf{W}) - f(\mathbf{x}_j^i, \mathbf{W})}{\Delta \mathbf{W}} \times \Delta \mathbf{W} \\
&= f(\mathbf{x}_j^i, \mathbf{W} + \Delta \mathbf{W}) - f(\mathbf{x}_j^i, \mathbf{W}) \\
&= f(\mathbf{x}_j^i, \mathbf{W} + \Delta \mathbf{W}) - \mathbf{y}_j^i
\end{aligned} \tag{17}$$

This $\mathbf{S}_\mathbf{w}^{ij}$ serves as a computable stand-in for the theoretical $\frac{\partial f}{\partial \mathbf{W}} \Delta \mathbf{W}$. It provides a way to empirically evaluate the impact of a finite change $\Delta \mathbf{W}$ in the parameter $\mathbf{W}$ on the output f.

By using $\mathbf{S}_\mathbf{w}^{ij}$, we effectively transition from a theoretical concept to a practical, computable metric that can be used in various applications, including but not limited to, network pruning, feature selection, and model interpretation.

Similarly,

$$\mathbf{S}_\mathbf{x}^{ij} = f(\mathbf{x}_j^i + \Delta \mathbf{x}_j^i, \mathbf{W}) - \mathbf{y}_j^i \tag{18}$$

B. Loss Function to its Gradient

The local loss function $\mathcal{L}_j^i$ in its expanded form is:

$$\mathcal{L}_j^i = \left(\frac{\partial f}{\partial \mathbf{x}_j^i} d\mathbf{x}_j^i + \frac{\partial f}{\partial \mathbf{W}} d\mathbf{W} \right)^T \left(\frac{\partial f}{\partial \mathbf{x}_j^i} d\mathbf{x}_j^i + \frac{\partial f}{\partial \mathbf{W}} d\mathbf{W} \right) \tag{19}$$

We aim to find $\nabla_{d\mathbf{W}} \mathcal{L}_j^i$, which is the gradient of $\mathcal{L}_j^i$ with respect to $d\mathbf{W}$.

First, let's define the term inside the squared norm:

$$\mathbf{u} = \frac{\partial f}{\partial \mathbf{x}_j^i} d\mathbf{x}_j^i + \frac{\partial f}{\partial \mathbf{W}} d\mathbf{W} \tag{20}$$

Differentiating $\mathcal{L}_j^i$ with respect to $d\mathbf{W}$, and treating $\mathbf{u}$ as a function of $d\mathbf{W}$ and $d\mathbf{x}_j^i$, we have:

$$\frac{\partial \mathcal{L}_j^i}{\partial d\mathbf{W}} = 2\mathbf{u} \frac{\partial \mathbf{u}}{\partial d\mathbf{W}} \tag{21}$$

First, recall the term $\mathbf{u}$ which is defined as:

$$\mathbf{u} = \frac{\partial f}{\partial \mathbf{x}_j^i} d\mathbf{x}_j^i + \frac{\partial f}{\partial \mathbf{W}} d\mathbf{W} \tag{22}$$

Consider each term in $\mathbf{u}$:

1. The term $\frac{\partial f}{\partial \mathbf{x}_j^i} d\mathbf{x}_j^i$ does not contain $d\mathbf{W}$, so its derivative with respect to $d\mathbf{W}$ is zero.

2. The term $\frac{\partial f}{\partial \mathbf{W}} d\mathbf{W}$ is directly dependent on $d\mathbf{W}$.

So when we differentiate, we have:

$$\frac{\partial \mathbf{u}}{\partial d\mathbf{W}} = 0 + \frac{\partial(\frac{\partial f}{\partial \mathbf{W}} d\mathbf{W})}{\partial d\mathbf{W}} \tag{23}$$

Simplifying, we get:

$$\frac{\partial \mathbf{u}}{\partial d\mathbf{W}} = \frac{\partial(\frac{\partial f}{\partial \mathbf{W}} \mathcal{K})}{\partial \mathcal{K}}; \mathcal{K} = d\mathbf{W}$$
$$\frac{\partial \mathbf{u}}{\partial d\mathbf{W}} = \frac{\partial f}{\partial \mathbf{W}} \tag{24}$$

Substituting back $\mathbf{u}$:

$$\frac{\partial \mathcal{L}_j^i}{\partial d\mathbf{W}} = 2\left(\frac{\partial f}{\partial \mathbf{x}_j^i} d\mathbf{x}_j^i + \frac{\partial f}{\partial \mathbf{W}} d\mathbf{W}\right)\frac{\partial f}{\partial \mathbf{W}} \tag{25}$$

Since $\mathbf{u} = d\mathbf{y}$ for simplification, we have:

$$\nabla_{d\mathbf{W}}\mathcal{L}_j^i = 2d\mathbf{y}_j^i\frac{\partial f}{\partial \mathbf{W}} \tag{26}$$

To elaborate, if $d\mathbf{W}$ is a small change in $\mathbf{W}$, it can be thought of as $\Delta\mathbf{W}$, a small increment. In this framework, the derivative of this small change $\Delta\mathbf{W}$ with respect to $\mathbf{W}$ is simply 1, or the identity matrix if $\mathbf{W}$ is a vector or matrix.

C. Approximating the Gradient of f with Respect to $\mathbf{W}$

Consider a function $\mathbf{y}_j^i = f(\mathbf{W}, \mathbf{x}_j^i)$ where $\mathbf{W}$ is a matrix of shape $[M, D]$ and $\mathbf{x}_j^i$ is a matrix of shape $[1, D]$. We wish to approximate the gradient of f with respect to $\mathbf{W}$, denoted as $\frac{\partial f}{\partial \mathbf{W}}$.

We introduce a small perturbation $\Delta\mathbf{W}$ of shape $[M, D]$ to $\mathbf{W}$ and compute the change in f as $S_\mathbf{W}^{ij} = f(\mathbf{W} + \Delta\mathbf{W}, \mathbf{x}_j^i) - f(\mathbf{W}, \mathbf{x}_j^i)$.

The approximation of $\frac{\partial f}{\partial \mathbf{W}}$ can be expressed as:

$$\frac{\partial f}{\partial \mathbf{W}} \approx S_\mathbf{W}^{ij}\Delta\mathbf{W}^+ \tag{27}$$

C.1. Full-rank of $\triangle\mathbf{W}$

To ensure that $\Delta\mathbf{W}$ is invertible, it must be a full-rank matrix. A matrix is said to have full rank if its rank is equal to the minimum of its number of rows and columns. In our case, $\Delta\mathbf{W}$ should have a rank of $\min(M, D)$.

C.2. Moore-Penrose Pseudoinverse

In cases where $\Delta\mathbf{W}$ is not of full rank or is not square, we resort to using the Moore-Penrose pseudoinverse, denoted by $\Delta\mathbf{W}^+$. The pseudoinverse provides a least-squares approximation solution to the problem, and it is computed using singular value decomposition (SVD).

D. Automating threshold for r% sparsity

$$\text{Sorted } \mathbf{W}_i^* = \text{sort}(\mathbf{W}_i^*)$$
$$\mathcal{T}_s = \text{Sorted } \mathbf{W}_i^* \left[\lceil (1 - s/100) \times N\rceil\right] \tag{28}$$

To achieve a sparsity ratio of $s\%$, the threshold is set by sorting $\mathbf{W}_i^*$ and selecting the value at the $(1 - s/100)$-th percentile, where N is the total number of elements in $\mathbf{W}_i^*$.

E. Overall standard deviations

Table 4: Results of overall mean and standard deviation for continual pruning on wikitext2, ptb, c4 datasets with all incremental permutations on LLaMA-7B, 13B, 30B, 65B. Unstructured pruning with 50% sparsity ratio.

	OVERALL	
	BWT	PPL
LLAMA-7B		
DENSE (NO PRUNING)	-	7.714 ± 1.833
MAGNITUDE	WS	30.246 ± 13.987
SPARSEGPT	0.591 ± 0.122	10.166 ± 2.386
WANDA	0.569 ± 0.5556	9.991 ± 2.541
COPAL (OURS)	**0.016 ± 0.017**	**9.728 ± 2.221**
LLAMA-13B		
DENSE (NO PRUNING)	-	6.990±1.634
MAGNITUDE	WS	28.935 ± 9.026
SPARSEGPT	0.606 ± 0.381	8.467 ± 1.766
WANDA	0.203 ± 0.065	8.570 ± 1.893
COPAL (OURS)	**0.029 ± 0.028**	**8.354 ± 1.851**
LLAMA-30B		
DENSE (NO PRUNING)	-	6.131 ± 1.657
MAGNITUDE	WS	10.958 ± 2.903
SPARSEGPT	0.395 ± 0.209	7.452 ± 1.520
WANDA	0.132 ± 0.043	7.261 ± 1.595
COPAL (OURS)	**0.007 ± 0.008**	**7.240 ± 1.576**
LLAMA-65B		
DENSE (NO PRUNING)	-	6.139 ± 2.173
MAGNITUDE	WS	9.399 ± 3.235
SPARSEGPT	0.334 ± 0.192	7.235 ± 1.971
WANDA	0.172 ± 0.221	7.584 ± 2.603
COPAL (OURS)	**0.001 ± 0.078**	**6.791 ± 1.666**