Controllable Memorization in LLMs via Weight Pruning

Chenjie Ni[1], Zhepeng Wang[2], Runxue Bao[3], Shangqian Gao[4]*, Yanfu Zhang[5]*,
[1]Northwestern University, [2]George Mason University,
[3]University of Pittsburgh, [4]Florida State University, [5]William and Mary
[1]chenjieni2027@u.northwestern.edu, [2]zwang48@gmu.edu, [3]baorunxue@gmail.com,
[4]sgao@cs.fsu.edu, [5]yzhang105@wm.edu

Abstract

The evolution of pre-trained large language models (LLMs) has significantly transformed natural language processing. However, these advancements pose challenges, particularly the unintended memorization of training data, which raises ethical and privacy concerns. While prior research has largely focused on mitigating memorization or extracting memorized information, the deliberate control of memorization has been underexplored. This study addresses this gap by introducing a novel and unified gradient-based weight pruning framework to freely control memorization rates in LLMs. Our method enables fine-grained control over pruning parameters, allowing models to suppress or enhance memorization based on application-specific requirements. Experimental results demonstrate that our approach effectively balances the trade-offs between memorization and generalization, with an increase of up to 89.3% in Fractional ER suppression and 40.9% in Exact ER amplification compared to the original models. [1]

1 Introduction

The rapid advancement of large language models (LLMs; Devlin, 2018; Radford et al., 2019b; Raffel et al., 2020) has revolutionized the field of natural language processing, enabling them to generate high-quality text, solve complex tasks, and even emulate human-like reasoning (Yang et al., 2022; Goyal et al., 2022; Celikyilmaz et al., 2020). However, a significant challenge is memorization, where LLMs retain sensitive details from their training data even without overfitting (Radford et al., 2019a). This raises ethical concerns, such as data leakage and privacy violations.

The phenomenon of memorization in LLMs has motivated many researchers to tackle this issue.

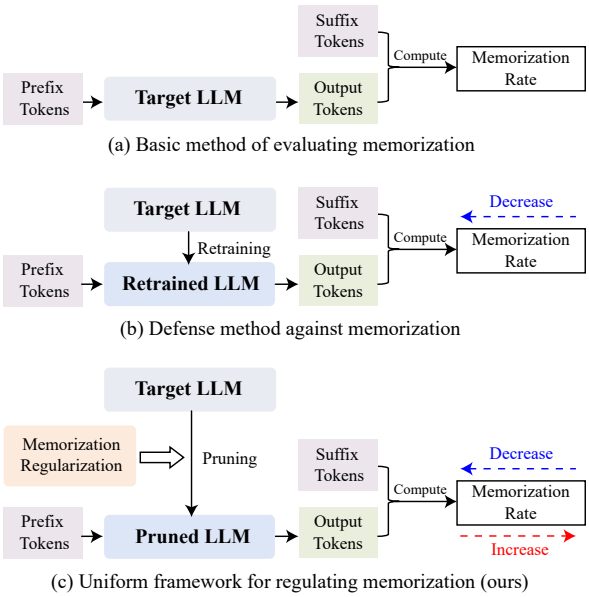


Figure 1: An abstract comparison of methods of regulating memorization.

Previous studies suggest that a key factor that influences memorization is model capacity (Chen et al., 2024; Wei et al., 2024; Carlini et al., 2022). Larger models, while exhibiting superior generalization capabilities, are often prone to memorizing more training data. This suggests that controlling model capacity could be a direct lever for controlling memorization.

Among the various techniques for modulating model capacity, pruning has emerged as a classic yet effective approach. Despite its potential, the relationship between pruning and memorization remains underexplored. Traditional pruning techniques primarily aim to improve inference efficiency or reduce computational requirements. The memorization behavior of LLMs receives limited attention in the context of pruning, with only Gupta et al. (2025) briefly explores the effectiveness of pruning in reducing memorization.

In addition, efforts to address memorization in

* Corresponding author.
[1]Our source code is publicly available at: https://github.com/artemis03976/pruning_memorization

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 15130–15145
November 4-9, 2025

LLM have focused mainly on minimizing susceptibility (i.e. defense) or exploiting memorization (i.e. attack) (Ozdayi et al. (2023); Kassem et al. (2024); Dong et al. (2024); Hans et al. (2024)). To the best of our knowledge, no prior work has explored deliberately increasing memorization rates. Although counterintuitive, there are scenarios where an increase in memorization rate could be highly beneficial for internal high-fidelity knowledge. For example, in tasks such as Encryption and In-context Learning(Stevens and Su, 2023;Golchin et al., 2024), it is often desirable for the model to recall and reproduce information with high fidelity and precision.

To address these gaps, our work investigates gradient-based weight pruning as a targeted mechanism to control the memorization rate in LLMs. By leveraging gradient information, we identify and prune weights that contribute minimally to the training objectives while still retaining sufficient capacity for essential tasks. Crucially, this approach provides a unified framework for controlling the memorization rate—enabling both suppression and amplification—based on task requirements. This adaptability allows LLMs to be aligned with diverse use cases, from enhancing privacy to preserving important task-specific details.

Our contributions are summarized as follows:

- We propose a unified framework leveraging gradient-based weight pruning to regularize memorization in LLMs. The framework is adaptable to diverse architectures and datasets.
- We design a flexible mechanism to adjust the pruning target, providing fine-grained control over memorization rates, allowing both suppression and amplification tailored to application-specific requirements.
- We perform comprehensive experiments to validate the efficacy of our approach. Results show that our method can significantly suppress or amplify memorization while preserving overall model performance.

2 Related Works

LLM Memorization. Memorization in large language models (LLMs) was first highlighted by Carlini et al. (2021), who demonstrated that attackers could extract training data by generating random prefixes to prompt the target model. Memorization in LLMs poses significant ethical and practical concerns. For example, Karamolegkou et al. (2023) highlighted the potential for copyright violations involving proprietary materials. Zhou et al. (2024) demonstrates the possibility of entity-level extraction from the generation. Moreover, researchers reported that confidential source code from Samsung, shared with LLMs from OpenAI, was later exposed to other users due to memorization (Mauran, 2023).

Several factors have been identified as critical to influencing memorization in LLMs. Carlini et al. (2022) demonstrated that model capacity, repetition in the training dataset, and context length are key contributors to memorization. Similarly, Chen et al. (2024) found that model size, complement size, and context size play a significant role in shaping memorization patterns.

To investigate and address memorization, researchers have proposed various attack and defense strategies. For instance, Kassem et al. (2024) introduces an additional attacker LLM to propel the victim LLM. Ozdayi et al. (2023) leverages soft-prompts techniques and designs prompt training strategies for both attack and defense settings. Dong et al. (2024) leverages self-distillation and designs a deliberate imagination mechanism to guide LLMs to unlearn training data. Gupta et al. (2025) proves that pruning is a powerful tool regarding reducing memorization.

Previous studies have focused on reducing or exploiting memorization. In contrast, our work explores a novel perspective: bidirectionally controlling the memorization rate, enabling task-specific adjustments that address both privacy concerns and performance optimization.

Pruning. Pruning is a technique widely adopted to decreases inference latency and enables resource-constrained deployment (LeCun et al., 1989; Hassibi et al., 1993; Hoefler et al., 2021; Gao et al., 2023; Wu et al., 2024; Gao et al., 2024b). Pruning methods are typically categorized into structural pruning, which targets entire layers or blocks, and weight pruning, which operates on individual weights (Cheng et al., 2024).

Various pruning techniques have been explored within the realm of LLMs. One foundational approach is magnitude-based pruning (Han et al., 2015), which eliminates weights with magnitudes below a predefined threshold. Building on this, recent research has focused on layer-wise weight pruning for LLMs. For instance,

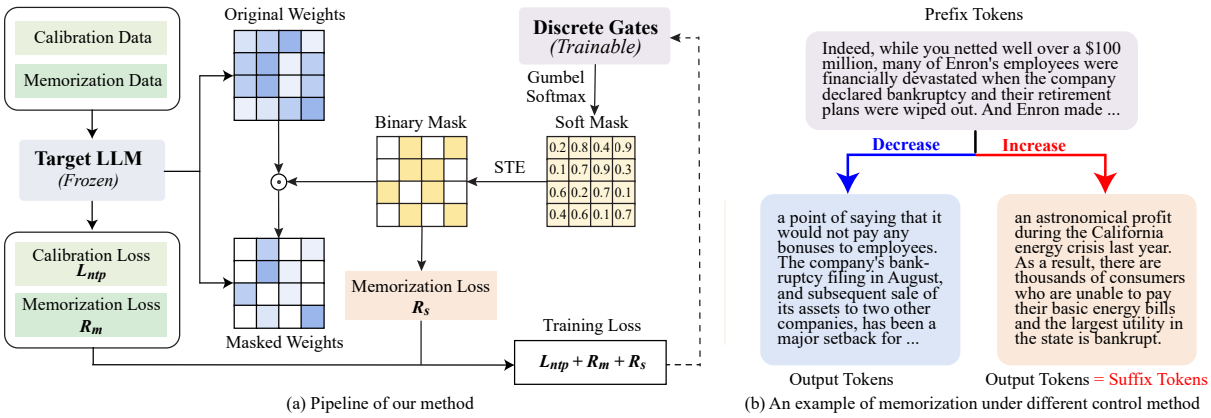


Figure 2: Illustration of our method

SparseGPT (Frantar and Alistarh, 2023) formulates a local layer-wise reconstruction problem and prunes with weight update. Similarly, Wanda (Sun et al., 2023) adopts a computationally efficient magnitude-pruning approach that leverages input activations. In addition to weight pruning, structural pruning techniques have also gained traction. LLM Pruner (Ma et al., 2023) removes noncritical coupled structures based on gradient information while adhering to generalization constraints.

Although previous methods have achieved success in improving model efficiency, their interaction with memorization behaviors remains unclear. Our work leverages weight pruning with targeted regularization to manipulate model capacity, enabling fine-grained control over memorization.

Localization-based methods. Recent studies have explored localization-based methods that aim to suppress memorization. These methods first identify the most influential neurons or weights, and then apply targeted interventions to erase this information. For example, DEPN (Wu et al., 2023) introduces a detect-and-edit framework that leverages gradient attribution to locate "privacy neurons" in feed-forward layers and suppresses memorization by deactivating them. Similarly, EMSO (Zhang et al., 2025) proposes an entropy maximization objective combined with a contrastive gradient metric to localize influential weights and selectively update them, thereby erasing memorization while preserving general model utility.

While both our approach and these localization-based methods leverage gradients to identify influential parameters, their fundamental mechanisms differ significantly. Localization-based methods modify a small subset of existing components. In contrast, our framework permanently removes weights to create a sparser subnetwork. This allows for a more direct manipulation of model capacity. Furthermore, whereas localization methods are primarily designed for erasure, our pruning framework offers both suppression and amplification of memorization.

3 Method

To control the memorization rates of large language models, we propose a gradient-based weight pruning framework. This is motivated by the observation that memorization in LLMs is a holistic property, requiring a complete forward pass to accurately assess it. Existing weight pruning methods, such as SparseGPT and wanda, evaluate individual weights on a layer-by-layer basis, excelling in efficiency but falling short when it comes to directly analyzing memorization. In contrast, our approach introduces trainable masks to the parameters of the model. An overall pipeline of our method is shown in Figure 2. By leveraging gradient information derived from memorization-specific data, these masks enable precise identification of weights that significantly contribute to memorization. This fine-grained control provides a more effective mechanism for addressing limitations in existing approaches while maintaining flexibility and adaptability.

3.1 Preliminaries and Objective

We first formulate a generalized framework for model pruning and then define a memorization-aware optimization problem based on it.

We consider model pruning as an optimization problem with a generic objective function $\mathcal{L}(\cdot)$, defined for a model $f : x \to y$ parameterized by $\mathcal{W}$. To control model sparsity, we introduce

a binary weight mask $\mathbf{g}_l \in \mathbb{R}^{H\times W}$ for the l-th linear layer, which activates or deactivates specific neurons in the corresponding layers. Let r denote the desired sparsity ratio. The pruning problem is formally expressed as:

$$\min_{\Theta} \mathcal{L}(f(x;\mathcal{W},\Theta),y)$$
$$\text{s.t.} \sum_{l=1}^{L}\sum_{i=1}^{H}\sum_{j=1}^{W} g_{l_{ij}} - rP = 0, \quad g_{l_{ij}} \in \{0,1\}, \tag{1}$$

Here, $g_{l_{ij}}$ denotes the entries of $\mathbf{g}_l \in \mathbb{R}^{H\times W}$, Θ represents the learnable parameters governing $\mathbf{g}_l$, P is the total number of weight parameters across all linear layers, L is the number of decoder layers in LLM, and H and W denotes the height and width of the weight matrices for each linear layer respectively. Since LLMs are composed of linear layers, we then focus on pruning all key/value/query projections and output projections of attention modules, as well as the feedforward layers in each transformer block.

To solve (1) using gradient-based optimization, we apply Lagrangian relaxation to transform the constrained problem into an unconstrained formulation with a sparsity-regularization term:

$$\min_{\Theta} F(\Theta) = \mathcal{L}_{clm} + \lambda_s R_s \tag{2}$$

where the first term $\mathcal{L}_{clm}$ represents a standard loss function for LLMs, namely causal language modeling (CLM) loss, R_s stands for the specific regularization function detailed in Section 3.3, and λ_s is the corresponding coefficient.

To explicitly regularize memorization during training, we extend the objective function with an additional regularization term that accounts for model performance on memorization-specific tasks. Specifically, we evaluate the CLM loss on the memorization dataset. This is based on the idea that *memorization can be conceptualized as a specific form of generation capability.* The resulting optimization problem is expressed as:

$$\min_{\Theta} F(\Theta) = \mathcal{L}_{clm} + \lambda_s R_s + \lambda_m R_m \tag{3}$$

where R_m is the specific regularization function discussed in Sections 3.4 and 3.5. λ_m is a hyperparameter that balances the trade-off between model generalization and memorization control. The new term $\lambda_m R_m$ explicitly encourages or suppresses memorization behavior, depending on the task requirements. To achieve this, we further design specific regularization functions tailored to the desired effect in the following sections.

3.2 Generating Binary Mask

We leverage a stochastic gate function, where binary masks are sampled probabilistically based on the value of θ. This stochastic approach expands the search space, allowing the model to explore a broader range of potential weight configurations during training. A more detailed formulation is demonstrated in Appendix A.

Since sampling discrete binary masks is inherently non-differentiable, we adopt the Gumbel-Softmax trick (Jang et al., 2016) to sample soft masks from θ, then with Straight-Through Estimator (STE; Bengio et al., 2013) to binarize these masks. In our case of generating binary masks, the Gumbel-Softmax trick simplifies into Gumbel-Sigmoid. Accordingly, the soft mask is sampled through:

$$m_i = \sigma\left(\frac{\log\theta_i + g_i}{\tau}\right)$$
$$g = -\log(-\log(u)), u \sim U(0,1) \tag{4}$$

where $\tau > 0$ is the temperature parameter that controls the softness of the output, σ denotes the sigmoid function.

3.3 Optimization of Sparsity

To achieve the desired sparsity, the sparsity regularization term should converge to near zero to meet the constraint during the early stages of training and remain minimal for the rest of the training process. This strategy allows for better exploration of the trade-offs between memorization and generalization under the target sparsity constraint. Building on prior research that demonstrated the advantages of global sparsity optimization, we calculate the loss over the global sparsity ratio, which is defined as,

$$R_s = \log(|\frac{P_a}{P} - (1-r)| + 1) \tag{5}$$

where P_a indicates the number of remaining weights or, equivalently, the number of activated gates. This global setting allows the optimization to focus on pruning the most memorization-related weights across the entire model, considering these weights may not spread uniformly in every layer.

Rather than directly regularizing the absolute number of activated weights, we focus on the sparsity ratio. This choice is motivated by the sheer

scale of parameters in LLMs, where even minor deviations from the desired sparsity ratio can result in significant differences in absolute weight counts. While regularizing the sparsity ratio does introduce certain limitations, these can be effectively mitigated by adjusting the coefficient λ_s.

3.4 Regularization for Increasing Memo Rate

As discussed in Section 3.2, we treat memorization as a specific form of a generation task. Increasing the memorization rate of a model entails improving its generation accuracy on memorization-specific data.

Building on the optimization problem described in Section 3.2, we incorporate the CLM loss of the model on a memorization dataset as an additional regularization term to increase memorization during weight pruning. The revised optimization objective is expressed as:

$$\min_{\Theta} F(\Theta) = \mathcal{L}(f(x_c; \mathcal{W}, \Theta), y_c) + \lambda_s R_s + \lambda_m \mathcal{L}(f(x_m; \mathcal{W}, \Theta), y_m) \tag{6}$$

where x_c, x_m denotes calibration data and memorization data respectively, λ_m regulates the impact of the memorization-specific regularization term. The calibration data is a training set used to maintain generalization ability of the model during pruning, while memorization data evaluate the performance of the model on memorization-specific tasks.

Notably, when increasing the memorization rate, the optimization goals for the calibration dataset and the memorization dataset align, as both aim to minimize the CLM loss. This observation implies that, in principle, a single dataset could suffice for optimizing both objectives. However, to ensure robust generalization and precise control over the model's memorization capabilities, we use distinct datasets for calibration and memorization tasks.

3.5 Regularization for Decreasing Memo Rate

In contrast to increasing the memorization rate, decreasing it requires raising the CLM loss on the memorization dataset. A straightforward approach might involve directly subtracting the memorization regularization term from the total loss. However, this could be ineffective because it creates an unbounded optimization objective, amplifying gradient sensitivity through the division effect of the log function in CLM loss. Consequently, it will disrupt pruning mask updates and risk degradation of the model's generation ability.

Instead, we are inspired by recent advancements in machine unlearning (Yao et al., 2023; Halimi et al., 2022; Yao et al., 2024), where gradient ascent techniques are used to "forget" specific data points from the model. By increasing the CLM loss on memorization data with unlearning methods, the model effectively learns to unmemorize these patterns, thereby reducing its dependence on memorized information. To ensure stability and control over the extent of memorization reduction, we adopt the concept of *learning threshold* from Ozdayi et al. (2023) and introduce a two-step optimization process, executed sequentially per iteration to avoid conflicts between objectives:

Step 1: Minimize the pruning loss, which includes the calibration loss and sparsity regularization.

$$\min_{\Theta} G(\Theta) = \mathcal{L}(f(x_c; \mathcal{W}, \Theta), y_c) + \lambda_s R_s \tag{7}$$

Step 2: Optimize the memorization loss R_m to meet learning threshold t, ensuring controlled reduction of memorization rate.

$$\min_{\Theta} |\lambda_m R(f(x_m; \mathcal{W}, \Theta), y_m) - t| \tag{8}$$

By setting an appropriate threshold t, this process allows us to find a trade-off between the overall performance of the model and memorization reduction, rather than assuming perfect separability.

4 Experiment

4.1 Experiment Setting

Chosen models. Investigating memorization requires the target models to be open-sourced regarding both model and training data. We selected two representative families of large language models: GPT-Neo (125M, 1.3B, 2.7B) (Black et al., 2021) and Pythia (410M, 1.4B, 2.8B) (Biderman et al., 2023). GPT-Neo and Pythia are pretrained on the Pile dataset (Gao et al., 2020a).

Datasets. We used two types of training datasets: Calibration data for stabilizing generalization ability of the model during pruning, and memorization data for evaluating the memorization ability of the model. For calibration, we sampled 45,000 sentences from C4 training set (Raffel et al., 2020)

and concatenated them to match the model's context length, separated by a special SEP token. We evaluated the model's loss and perplexity on the held-out WikiText (Merity et al., 2016) validation set. For memorization, we extracted training data of GPT-Neo and Pythia with the Language Model Extraction Benchmark dataset (Google-research, 2022). This dataset is a curated subset of the Pile, containing 15,000 sequences sampled specifically for studying memorization behaviors. We split this dataset into 14,000 samples for training and the remaining 1,000 for testing.

Baseline. We compared our method against three state-of-the-art pruning baselines commonly used for large language models: magnitude pruning (Han et al., 2015), a straightforward yet effective approach in which weights are discarded based on their magnitudes; SparseGPT (Frantar and Alistarh, 2023), a second-order pruning method solving a layer-wise reconstruction problem; and Wanda (Sun et al., 2023), a computationally efficient layer-wise pruning method leveraging activation outcomes without any retraining. We also choose CSP (Ozdayi et al., 2023) as a baseline to compare against the effect of memorization mitigation.

Implementation. All main experiments were conducted with 30% pruning sparsity. We used AdamW optimizer with weight decay = 0.001 and cosine annealing scheduler. Learning threshold t is set to different values for each kind of model. For more details, refer to Appendix E.

4.2 Main Results

We present the evaluation results of our proposed method in Tables 1 and 2 for the family of GPT-Neo and Pythia, respectively. Following Ozdayi et al. (2023) and Wang et al. (2024), we use Fractional Extraction Rate (Fractional ER) and Exact Extraction Rate (Exact ER) as the main metric for evaluating memorization. Exact ER measures the proportion of the outputs where the model reproduces the exact suffix from training data, and Fractional ER measures the average proportion of matching tokens between generated output and the actual suffix, normalized over token length.

Performance on Pruning. Our method consistently performs either equally or better than the other pruning baselines. This is mainly due to the use of global gradient information and its ability to adjust pruning configurations. Specifically, for the GPT-Neo family, our method provides a notable boost in generation ability, improving performance by 39.4%, 14.1% and 30.4%, respectively. These results demonstrate the effectiveness of our pruning approach, laying a solid foundation for the subsequent memorization regularization task.

Performance on Memorization. We evaluated the performance of our method in regularizing memorization with various settings. Since the three baseline pruning methods lack dedicated mechanisms to control memorization, their memorization rates are passively affected by pruning, showing the tendency to keep a certain amount of training data. For instance, pruning GPT-Neo-2.7B with Wanda reserves 65.9% of the Fractional ER and 41.9% of the Exact ER.

In comparison, the proposed methods work efficiently in their respective setting. For the case of increasing the memorization rate, our method can maximize the growth of the model's memorization, or even exceed the original model's memorization rate, while basically retaining the original performance of the model. For the GPT-Neo family, we level the memorization rate on an average of 20.0% for Fractional ER and 40.9% for Exact ER, and for the Pythia family, the corresponding increase is 11.7% and 31.8%.

Moreover, our method can also largely suppress the output of the training content in decreasing setting. For instance, we can reach a nearly 100% reduction in Exact ER and 89.4% reduction in Fractional ER based on the learning threshold t. Compared with the CSP method for suppressing memorization, our method works well for both GPT-Neo and Pythia families with an superior performance. Furthermore, our approach ensures that the model's performance on memorized data stays around the learning threshold t, demonstrating its effectiveness in controlling memorization.

Extra Analysis. Besides the above analysis on the effectiveness of our methods, several additional observations are worth noting. First of all, an interesting phenomenon is that our method, even when no explicit memorization regularization is applied, can achieve a significant reduction in memorization rate, compared to the other three baselines. We attribute this to the increase in the generalization ability of the model after pruning using our method. Considering that memorization affects the generalization ability of the model, it is likely that if the generalization ability is strengthened, the problem of memorization can be mitigated accordingly.

Moreover, we observe a trend in the model performance under different memorization settings.

Table 1: Results on GPT-Neo Family with 30% sparsity

Model	Method	WikiText Evaluation		Memorization Evaluation			
		Test Loss	Test PPL	Fractional ER	Exact ER	Test Loss	Test PPL
GPT-Neo 125M	Origin	7.219	1364.782	0.349	0.172	2.078	7.989
	Magnitude	9.813	18260.582	0.053	**0.000**	4.531	92.875
	SparseGPT	7.688	2180.915	0.118	**0.000**	2.578	13.172
	Wanda	7.469	1752.415	0.097	**0.000**	2.719	15.161
	CSP (down)	7.963	2871.701	0.084	0.003	3.155	23.447
	Ours (no memo)	**4.313**	**74.627**	0.060	**0.000**	3.422	30.627
	Ours (up)	4.438	84.563	**0.485**	**0.289**	2.406	11.092
	Ours (down)	4.375	79.440	**0.024**	**0.000**	6.594	730.515
GPT-Neo 1.3B	Origin	4.656	105.241	0.637	0.451	1.273	3.573
	Magnitude	5.656	286.074	0.149	0.003	2.156	8.639
	SparseGPT	4.656	105.241	0.391	0.145	1.484	4.412
	Wanda	4.844	126.945	0.397	0.154	1.531	4.624
	CSP (down)	6.556	703.190	0.166	0.012	2.560	12.938
	Ours (no memo)	3.453	31.599	0.115	0.001	2.531	12.569
	Ours (up)	3.563	35.251	**0.729**	**0.603**	1.852	6.370
	Ours (down)	**3.406**	**30.152**	**0.083**	**0.000**	2.703	14.926
GPT-Neo 2.7B	Origin	5.344	209.296	0.685	0.498	1.133	3.104
	Magnitude	6.031	416.235	0.144	0.004	2.078	7.989
	SparseGPT	5.281	196.615	0.477	0.230	1.289	3.629
	Wanda	5.531	252.459	0.452	0.209	1.344	3.833
	CSP (down)	6.288	538.141	0.133	0.007	2.422	11.270
	Ours (no memo)	3.469	32.097	0.104	**0.000**	2.625	13.805
	Ours (up)	3.578	35.806	**0.732**	**0.610**	2.078	7.989
	Ours (down)	**3.422**	**30.627**	**0.082**	**0.000**	2.671	14.467

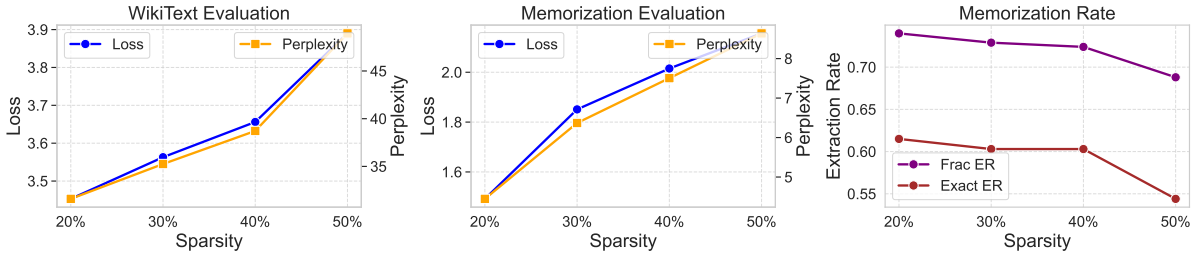


Figure 3: Ablation study of sparsity ratio on GPT-Neo-1.3B

Specifically, when the memorization rate is increased, the pruned model using our method typically exhibits slightly worse performance on the WikiText task compared to other cases. This observation aligns with the conclusion that increasing memorization negatively impacts the model's generalization performance, further highlighting the trade-off between memorization control and generalization ability.

Task-wise Performance. We also evaluate the downstream impact using the lm-evaluation-harness benchmark(Gao et al., 2024a) to illustrate the harmlessness of our pruning method on original generation tasks. Results on Pythia-1.3B show that our method maintains competitive performance across a variety of reasoning and comprehension tasks, with minimal degradation compared to the original model and clear advantages over existing pruning baselines. Full task-wise results and comparisons are reported in Appendix B.2.

Training Efficiency. Due to the global mask training procedure, our pruning method is more computationally expensive compared to layer-wise pruning methods like SparseGPT and Wanda. To quantify this, our method takes nearly one and a half hours to train on GPT-Neo-125M compared to layer-wise pruning within 5 minutes. However, these baselines are not designed for memorization control and are significantly less effective. Besides, compared to CSP, which is a strong memorization suppression baseline, our method performs better, while the cost of CSP is on par with our method.

4.3 Ablation Study

Various Sparsity. To better understand the impact of sparsity on memorization, we conduct an abla-

Table 2: Results on Pythia Family with 30% sparsity

Model	Method	WikiText Evaluation		Memorization Evaluation			
		Test Loss	Test PPL	Fractional ER	Exact ER	Test Loss	Test PPL
Pythia 410M	Origin	3.516	33.637	0.456	0.227	1.523	4.588
	Magnitude	4.563	95.823	**0.047**	**0.000**	3.172	23.852
	SparseGPT	4.125	61.868	0.066	0.001	2.578	13.172
	Wanda	3.891	48.941	0.143	0.010	2.125	8.373
	CSP (down)	7.083	1191.223	0.493	0.258	3.035	20.809
	Ours (no memo)	**3.625**	**37.525**	0.134	0.007	2.625	13.805
	Ours (up)	3.734	41.862	**0.604**	**0.392**	1.953	7.051
	Ours (down)	3.719	41.213	0.057	**0.000**	3.047	21.049
Pythia 1.4B	Origin	3.156	23.482	0.655	0.402	1.148	3.153
	Magnitude	3.594	36.370	0.144	0.007	2.125	8.373
	SparseGPT	3.250	25.790	0.305	0.060	1.477	4.378
	Wanda	3.250	25.790	0.367	0.128	1.375	3.955
	CSP (down)	6.399	601.291	0.678	0.415	2.531	12.568
	Ours (no memo)	3.266	26.196	0.148	0.015	2.688	14.695
	Ours (up)	**3.250**	**25.790**	**0.695**	**0.481**	1.930	6.887
	Ours (down)	3.281	26.609	**0.057**	**0.000**	3.188	24.228
Pythia 2.8B	Origin	3.000	20.086	**0.738**	0.494	1.016	2.761
	Magnitude	3.297	27.028	0.169	0.026	1.984	7.274
	SparseGPT	3.078	21.718	0.410	0.162	1.242	3.463
	Wanda	3.063	21.381	0.483	0.215	1.172	3.228
	CSP (down)	6.196	490.748	0.746	0.501	2.476	11.893
	Ours (no memo)	3.141	23.118	0.102	0.003	2.844	17.180
	Ours (up)	**3.047**	**21.718**	0.714	**0.510**	1.812	6.126
	Ours (down)	3.141	23.118	**0.074**	**0.000**	2.703	14.926

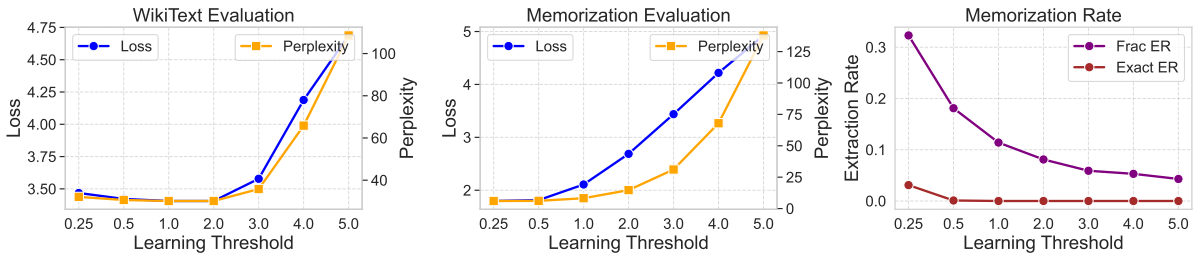


Figure 4: Ablation study of learning threshold on GPT-Neo-1.3B

tion study on the GPT-Neo-1.3B model, ranging from 20% to 50%. The results are presented in Figure 3.

As pruning inevitably leads to increased test loss and decreased memorization rates, it is hard to observe changes in memorization when applying suppression. Our primary focus here is on the behavior of the model under our amplification setting. As shown in Figure 3, both Fractional ER and Exact ER metrics exhibit a consistent decreasing trend as sparsity increases. For example, Fractional ER declines from 0.740 at 20% sparsity to 0.688 at 50%, and Exact ER drops from 0.615 to 0.544 over the same range.

On one hand, these results demonstrate the effectiveness of our approach in maximizing memorization rate across different sparsity levels. On the other hand, it also indicates that higher sparsity mitigates memorization to some degree, as fewer parameters are available to encode memorized information. However, the relatively modest decrease suggests that memorization behavior in large language models is more complex and resilient than initially anticipated.

Impact of Unlearning Threshold. In the unlearning setting, we introduce a learning threshold t from previous work (Ozdayi et al., 2023) to control the extent of memorization suppression. We further conduct experiments on the GPT-Neo-1.3B model, testing various threshold values to evaluate its influence. The results are presented in Figure 4.

Our analysis reveals that as t is incrementally raised above the baseline loss observed on the memorization dataset, there is a consistent reduction in

both Fractional ER and Exact ER metrics, which shows an increasingly potent suppression effect as t increases. However, it is important to note that when t is set significantly higher than the calibration loss of the model, the losses on both generalization and memorization data converge toward t. This suggests that overly aggressive suppression can unintentionally impair the overall ability to generate coherent outputs. This challenge in balancing memorization and generation will be revisited in Appendix C.

Interestingly, the learning threshold t also offers flexibility in amplification settings due to its optimization target. By lowering t below the original loss on the memorization dataset, it becomes possible to amplify the memorization. These findings demonstrate the dual role of the learning threshold t as a critical hyperparameter. While it effectively controls the degree of suppression, careful calibration is necessary to balance memorization suppression with the preservation of generalization capabilities.

Training Strategy We investigate how training strategies affect the performance of our pruning method. Our results indicate that using smaller batch sizes can achieve comparable performance to large batch sizes while being more computationally economical. Furthermore, prolonged training over multiple epochs tends to harm generalization due to overfitting on parameter redundancy. We find that one-pass training over a sufficiently large dataset is preferable. More detailed results and analysis are provided in Appendix B.1.

5 Conclusion

In this work, we explored the interplay between model pruning and memorization in LLMs, introducing a novel approach to control memorization rates using gradient-based weight pruning. Memorization in LLMs, while often regarded as a liability due to privacy and ethical concerns, can also serve as a valuable feature in specific applications, such as knowledge retention in domain-specific tasks. Our study presents a flexible framework that enables both the suppression and amplification of memorization by simply adjusting the pruning target to select parameters based on gradient information.

Through extensive experiments on open-source LLMs, including GPT-Neo and Pythia families, we demonstrated the effectiveness of our method across multiple model scales. By incorporating tailored regularization terms, we were able to achieve precise control over memorization rates while preserving or even enhancing generalization performance. Moreover, our comparative analysis against state-of-the-art pruning baselines highlighted the superior adaptability and efficiency of our approach in controlling memorization rate for LLMs.

Limitations

Despite the effectiveness of our proposed gradient-based weight pruning framework on controlling memorization rate, several limitations remain.

First, while the combination of Gumbel-Softmax sampling and STE allows for the generation of binary pruning masks with gradient flow, the reliance on stochastic methods may introduce inherent variability. This can lead to suboptimal convergence in certain scenarios, particularly when the model is sensitive to slight perturbations in the sparsity pattern. As a result, the method may require careful tuning of hyperparameters and learning rates, such as the temperature in Gumbel-Softmax and the sparsity loss coefficient λ_s, to ensure stability and optimal performance.

Second, considering the property of memorization, some weights simultaneously contribute to both memorization and generalization. This overlap can complicate the optimization process, as the removal of such weights might inadvertently degrade overall model quality. The trade-off between memorization and generalization needs a curated selection in learning threshold to have an ideal outcome.

Finally, the computational overhead of training with stochastic gates, such as Gumbel-Softmax sampling, is higher than that of deterministic methods like layer-wise pruning. This increased cost may pose challenges for scaling the framework to much larger language models.

Ethics Statement

This work explores the use of gradient-based pruning to control memorization in LLMs. While our method offers a novel mechanism to either suppress or enhance memorization, we acknowledge the ethical implications that arise from both aspects. Reducing memorization can contribute to mitigating privacy risks, such as the unintended leakage of sensitive or copyrighted training data. This has positive implications for deploying LLMs

in privacy-sensitive domains and aligning with data protection standards.

Conversely, our method also enables the amplification of memorization, which—if misused—could potentially increase the risk of reproducing proprietary or confidential content. We emphasize that our proposed framework is intended for controlled and responsible usage and should not be employed to intentionally extract or replicate private data.

We conducted our research without access to any personally identifiable or confidential information, and all experiments were performed on publicly available datasets. We encourage future work to further examine the balance between utility and safety in LLM memorization and to adopt appropriate safeguards when applying similar techniques.

Acknowledgement

This work is supported in part by the National Science Foundation (NSF) grant IIS-2451436 and Commonwealth Cyber Initiative grant HC-4Q24-059.

References

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, and 1 others. 2023. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pages 2397–2430. PMLR.

Sid Black, Leo Gao, Phil Wang, Connor Leahy, and Stella Biderman. 2021. Gpt-neo: Large scale autoregressive language modeling with mesh-tensorflow. *If you use this software, please cite it using these metadata*, 58(2).

Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramer, and Chiyuan Zhang. 2022. Quantifying memorization across neural language models. *arXiv preprint arXiv:2202.07646*.

Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, and 1 others. 2021. Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2633–2650.

Asli Celikyilmaz, Elizabeth Clark, and Jianfeng Gao. 2020. Evaluation of text generation: A survey. *arXiv preprint arXiv:2006.14799*.

Bowen Chen, Namgi Han, and Yusuke Miyao. 2024. A multi-perspective analysis of memorization in large language models. *arXiv preprint arXiv:2405.11577*.

Hongrong Cheng, Miao Zhang, and Javen Qinfeng Shi. 2024. A survey on deep neural network pruning: Taxonomy, comparison, analysis, and recommendations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Jacob Devlin. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Yijiang River Dong, Hongzhou Lin, Mikhail Belkin, Ramon Huerta, and Ivan Vulić. 2024. Unmemorization in large language models via self-distillation and deliberate imagination. *arXiv preprint arXiv:2402.10052*.

Elias Frantar and Dan Alistarh. 2023. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *International Conference on Machine Learning*, pages 10323–10337. PMLR.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, and 1 others. 2020a. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, and 5 others. 2024a. The language model evaluation harness.

Shangqian Gao, Feihu Huang, Jian Pei, and Heng Huang. 2020b. Discrete model compression with resource constraint for deep neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1899–1908.

Shangqian Gao, Junyi Li, Zeyu Zhang, Yanfu Zhang, Weidong Cai, and Heng Huang. 2024b. Device-wise federated network pruning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12342–12352.

Shangqian Gao, Zeyu Zhang, Yanfu Zhang, Feihu Huang, and Heng Huang. 2023. Structural alignment for network pruning through partial regularization. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 17402–17412.

Shahriar Golchin, Mihai Surdeanu, Steven Bethard, Eduardo Blanco, and Ellen Riloff. 2024. Memorization in in-context learning. *CoRR*.

Google-research. 2022. lm-extraction-benchmark. Accessed: 2024-12-02.

Tanya Goyal, Junyi Jessy Li, and Greg Durrett. 2022. News summarization and evaluation in the era of gpt-3. *arXiv preprint arXiv:2209.12356*.

Mansi Gupta, Nikhar Waghela, Sarthak Gupta, Shourya Goel, and Sanjif Shanmugavelu. 2025. Pruning as a defense: Reducing memorization in large language models. In *ICLR 2025 Workshop on Building Trust in Language Models and Applications*.

Anisa Halimi, Swanand Kadhe, Ambrish Rawat, and Nathalie Baracaldo. 2022. Federated unlearning: How to efficiently erase a client in fl? *arXiv preprint arXiv:2207.05521*.

Song Han, Jeff Pool, John Tran, and William Dally. 2015. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28.

Abhimanyu Hans, Yuxin Wen, Neel Jain, John Kirchenbauer, Hamid Kazemi, Prajwal Singhania, Siddharth Singh, Gowthami Somepalli, Jonas Geiping, Abhinav Bhatele, and 1 others. 2024. Be like a goldfish, don't memorize! mitigating memorization in generative llms. *arXiv preprint arXiv:2406.10209*.

Babak Hassibi, David G Stork, and Gregory J Wolff. 1993. Optimal brain surgeon and general network pruning. In *IEEE international conference on neural networks*, pages 293–299. IEEE.

Torsten Hoefler, Dan Alistarh, Tal Ben-Nun, Nikoli Dryden, and Alexandra Peste. 2021. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *Journal of Machine Learning Research*, 22(241):1–124.

Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*.

Antonia Karamolegkou, Jiaang Li, Li Zhou, and Anders Søgaard. 2023. Copyright violations and large language models. *arXiv preprint arXiv:2310.13771*.

Aly M Kassem, Omar Mahmoud, Niloofar Mireshghallah, Hyunwoo Kim, Yulia Tsvetkov, Yejin Choi, Sherif Saad, and Santu Rana. 2024. Alpaca against vicuna: Using llms to uncover memorization of llms. *arXiv preprint arXiv:2403.04801*.

Yann LeCun, John Denker, and Sara Solla. 1989. Optimal brain damage. *Advances in neural information processing systems*, 2.

Yun Li, Zechun Liu, Weiqun Wu, Haotian Yao, Xiangyu Zhang, Chi Zhang, and Baoqun Yin. 2022. Weight-dependent gates for network pruning. *IEEE Transactions on Circuits and Systems for Video Technology*, 32(10):6941–6954.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. 2023. Llm-pruner: On the structural pruning of large language models. *Advances in neural information processing systems*, 36:21702–21720.

Cecily Mauran. 2023. Whoops, samsung workers accidentally leaked trade secrets via chatgpt. *Mashable [online]. Dostupné z: https://mashable.com/article/samsungchatgpt-leak-details*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*.

Mustafa Safa Ozdayi, Charith Peris, Jack FitzGerald, Christophe Dupuy, Jimit Majmudar, Haidar Khan, Rahil Parikh, and Rahul Gupta. 2023. Controlling the extraction of memorized data from large language models via prompt-tuning. *arXiv preprint arXiv:2305.11759*.

Alec Radford, Jeffrey Wu, Dario Amodei, Daniela Amodei, Jack Clark, Miles Brundage, and Ilya Sutskever. 2019a. Better language models and their implications. *OpenAI blog*, 1(2).

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, and 1 others. 2019b. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67.

Samuel Stevens and Yu Su. 2023. Memorization for good: Encryption with autoregressive language models. *arXiv preprint arXiv:2305.10445*.

Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. 2023. A simple and effective pruning approach for large language models. *arXiv preprint arXiv:2306.11695*.

Zhepeng Wang, Runxue Bao, Yawen Wu, Jackson Taylor, Cao Xiao, Feng Zheng, Weiwen Jiang, Shangqian Gao, and Yanfu Zhang. 2024. Unlocking memorization in large language models with dynamic soft prompting. In *EMNLP*.

Jiaheng Wei, Yanjun Zhang, Leo Yu Zhang, Ming Ding, Chao Chen, Kok-Leong Ong, Jun Zhang, and Yang Xiang. 2024. Memorization in deep learning: A survey. *arXiv preprint arXiv:2406.03880*.

Xidong Wu, Shangqian Gao, Zeyu Zhang, Zhenzhen Li, Runxue Bao, Yanfu Zhang, Xiaoqian Wang, and Heng Huang. 2024. Auto-train-once: Controller network guided automatic network pruning from scratch. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16163–16173.

Xinwei Wu, Junzhuo Li, Minghui Xu, Weilong Dong, Shuangzhi Wu, Chao Bian, and Deyi Xiong. 2023. Depn: Detecting and editing privacy neurons in pretrained language models. In *The 2023 Conference on Empirical Methods in Natural Language Processing*.

Antoine Yang, Antoine Miech, Josef Sivic, Ivan Laptev, and Cordelia Schmid. 2022. Zero-shot video question answering via frozen bidirectional language models. *Advances in Neural Information Processing Systems*, 35:124–141.

Jin Yao, Eli Chien, Minxin Du, Xinyao Niu, Tianhao Wang, Zezhou Cheng, and Xiang Yue. 2024. Machine unlearning of pre-trained large language models. *arXiv preprint arXiv:2402.15159*.

Yuanshun Yao, Xiaojun Xu, and Yang Liu. 2023. Large language model unlearning. *arXiv preprint arXiv:2310.10683*.

Zhaohan Zhang, Ziquan Liu, and Ioannis Patras. 2025. Get confused cautiously: Textual sequence memorization erasure with selective entropy maximization. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 10924–10939.

Zhenhong Zhou, Jiuyang Xiang, Chaomeng Chen, and Sen Su. 2024. Quantifying and analyzing entity-level memorization in large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19741–19749.

A Model Details

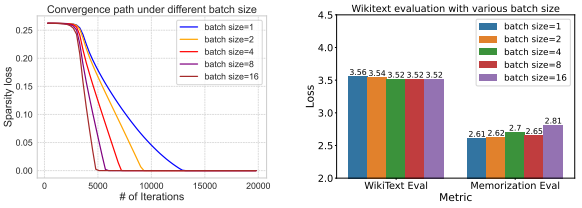
We construct pruning masks with discrete binary gates to achieve global gradient control (Li et al., 2022; Gao et al., 2020b). The basic formulation of a discrete gate can be defined as:

$$g(\theta) = \begin{cases} 1 & \text{if } \theta \in [0.5, 1], \\ 0 & \text{if } \theta \in [0, 0.5). \end{cases} \tag{9}$$

where $\theta \in [0, 1]$ is a learnable parameter.

This deterministic formulation provides a simple and direct mechanism for pruning. However, it introduces several critical limitations. Most notably, it does not account for weights that may appear inconsequential for general language generation but play a pivotal role in the model's memorization capability. Relying solely on deterministic binary gates often leads to suboptimal pruning masks, undermining the intended balance between sparsity and functionality.

To overcome these shortcomings, we extend the discrete gate function into a stochastic framework, allowing for greater flexibility and adaptability. In this approach, we need to maintain gradient flow during optimization while generating effective pruning masks. A straightforward method to achieve this involves applying a Straight-Through Estimator (STE) after Bernoulli sampling from the gate parameter θ. While STE enables gradient-based learning of discrete masks, it often struggles to generate high-quality masks for LLMs, especially when more nuanced control over sparsity and memorization is required.



(a) Training loss using various batch sizes

(b) Performance test on pruned models

Figure 5: Ablation study of batch size on GPT-Neo-1.3B with baseline method (No memorization regulation)

To address these challenges, we adopt a more robust sampling technique, which is Gumbel-Softmax sampling. This method not only provides a differentiable approximation of discrete gates but also enhances the expressiveness of the mask generation process. By leveraging Gumbel-Softmax, we achieve a balance between flexibility and precision, enabling the pruning mechanism to effectively target weights that are critical for controlling memorization while preserving the model's core capabilities. However, Gumbel-Softmax alone generates soft masks, even when the temperature is set to a low level, which is insufficient for our task. We require complete binary masks to represent the activation or deactivation of weight parameters. Thus, we incorporate STE to harden the soft masks produced by Gumbel-Softmax. This combination allows us to retain gradient flow during optimization while ensuring the binary nature of the final masks.

This stochastic gate formulation ensures smoother optimization and better alignment with the goals of sparsity-aware memorization control in LLMs, paving the way for more reliable and adaptable model pruning strategies.

B Complementary Experimental Results

B.1 Ablation Study on Training Strategy

Our experiments reveal several intriguing phenomena related to the choice of training strategies, shedding light on their impact on the effectiveness of our pruning method.

Batch size is a critical hyperparameter in standard model training, often set to large values

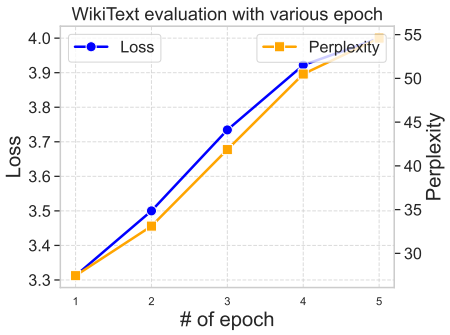


Figure 6: Ablation study of epoch on Pythia-1.4B

to smooth gradients and accelerate convergence. However, due to the large scale of LLMs and the relatively high computational cost associated with our pruning method, it is essential to explore the performance of our method using smaller batch sizes.

Figure 5a illustrates the optimization process with varying batch sizes, keeping the learning rate fixed. Despite that larger batch sizes can accelerate convergence, they also require more samples and increase computational demands. Furthermore, the results in Figure 5b reveal that there is little difference in the final performance across different batch sizes. These findings suggest that using larger batch sizes may not be an "economical" choice, as it increases training time, computational resource consumption, and the number of samples required. In fact, we find that the convergence of our algorithm is more closely tied to the sparsity loss coefficient λ_s, and the number of iterations, rather than batch size. By finetuning λ_s, our method can still converge optimally even when the batch size is as small as 1.

Another critical factor we analyzed is the number of training epochs. As shown in Figure 6, the performance of the pruned model significantly degrades as the number of training epochs increases. This degradation is primarily attributed to the large scale of parameters in LLMs. Despite freezing the original weights and only selecting them, the model still suffers from overfitting when trained for multiple epochs due to the excessive parameter redundancy in LLMs, where prolonged training reduces generalization performance. To mitigate this issue, we recommend using a larger dataset and training the model for a single epoch rather than training for multiple epochs with fewer samples. By focusing on a broader range of data in a single pass, the model can better preserve its generalization ability while achieving the desired pruning effects.

B.2 Task-wise performance

To comprehensively evaluate the impact of our pruning method beyond standard loss or perplexity, we conduct an in-depth task-level analysis on Pythia-1.3B using the lm-evaluation-harness benchmark developed by EleutherAI(Gao et al., 2024a). This benchmark encompasses a diverse set of language understanding tasks, including multiple-choice question answering, commonsense reasoning, and reading comprehension, thereby enabling a fine-grained inspection of the model's generalization capabilities across different reasoning paradigms.

As shown in Table 4, our pruned model demonstrates consistently competitive performance across the various tasks, exhibiting minimal degradation when compared to the unpruned baseline. When contrasted with other pruning baselines, our method yields significantly more favorable task-wise outcomes. For example, while Magnitude pruning causes a substantial drop in BoolQ performance (from 0.632 to 0.378), our approach retains a score above 0.63, closely matching the original model.

In summary, this task-wise evaluation reinforces the conclusion that our pruning strategy introduces negligible harm to the downstream functional performance of the model.

B.3 Ablation Study on Regularization Term

To disentangle the influence of sparsity regularization and memorization regularization, we conduct extra experiments to show their individual impact on the final outcome. The results are presented in Table 5, based on the memorization amplification setting. Here, we only focus on the "Memo-only" setting, as the other two outcomes have been presented and discussed in Section 4.1.

As expected, the final model has negligible sparsity (e.g., 0.003 for GPT-Neo-1.3B), while yields the highest memorization rates, with the Fractional ER reaching 0.770 for the 1.3B model. This outcome clearly demonstrates the effectiveness of the memorization regularization term when its full capacity is available. Compared with the "Both" setting, it proves that our method can enforce memorization even within a significantly pruned subnetwork, though comes at the cost of a slight degradation in performance of both tasks, highlighting the inherent trade-off between promoting memoriza-

Table 3: Results on suppression with reciprocal regularization

Model	WikiText Evaluation		Memorization Evaluation			
	Test Loss	Test PPL	Fractional ER	Exact ER	Test Loss	Test PPL
GPT-Neo-125M	4.375	79.440	0.014	0.000	98.5	5.998e+42
GPT-Neo-1.3B	3.484	32.602	0.030	0.000	69.5	1.526e+30
Pythia-410M	3.641	38.116	0.054	0.000	44.25	1.650e+19
Pythia-1.4B	4.125	61.868	0.036	0.000	55.75	1.628e+24

Table 4: Results on task-wise performance from lm-evaluation-harness benchmark

Model	ARC-e	ARC-c	BoolQ	HellaSwag	WinoGrande	PIQA	OpenBookQA	Avg
Origin	**0.604**	0.260	0.632	0.404	**0.574**	0.708	0.222	**0.486**
Magnitude	0.529	0.253	0.378	0.365	0.539	0.678	0.190	0.419
SparseGPT	0.574	0.249	0.617	0.392	0.558	0.705	0.202	0.471
Wanda	0.590	0.250	0.618	0.396	0.548	0.697	0.204	0.472
No memo	0.585	0.245	**0.641**	0.404	0.548	0.709	**0.224**	0.479
Up	0.587	0.253	0.634	0.402	0.559	0.711	0.220	0.481
Down	0.583	**0.270**	0.596	**0.418**	0.564	**0.720**	0.210	0.480

tion and maintaining general language modeling capabilities.

In summary, the sparsity regularization term is essential for achieving model sparsity, while the memorization regularization term is critical for controlling the model's memorization behavior. The combination of both terms allows for a precise balance, enabling the creation of a sparse yet highly memorizing model, which demonstrates that each component of our proposed loss function is vital to the final result.

C Additional Suppression Strategy

In addition to the unlearning method for suppressing memorization discussed in Section 3.5, we propose and investigate an alternative approach that incorporates the reciprocal of the CLM loss as a regularization term. This formulation inherently maximizes the model's loss on the memorization dataset while stabilizing the training process. By ensuring that the contribution of the memorization term decreases as the corresponding CLM loss on the memorization dataset increases, this strategy creates a smoother and more bounded optimization landscape.

The revised optimization objective is defined as:

$$\min_{\Theta} F(\Theta) = \mathcal{L}(f(x_c; \mathcal{W}, \Theta), y_c) + \lambda_s R_s + \lambda_m \frac{1}{\mathcal{L}(f(x_m; \mathcal{W}, \Theta), y_m)} \tag{10}$$

We implement the reciprocal regularization strategy for the above models and the results are shown in Table 3. The results demonstrate that this approach performs well for smaller models, such as GPT-Neo-125M, GPT-Neo-1.3B, and Pythia-410M. These models successfully retain generalization capability after pruning, while achieving extremely high loss on memorization datasets, effectively defending against extraction attacks.

However, this method often fails for larger or more powerful models, such as Pythia-1.4B. They exhibit degraded performance across datasets under this regularization, indicating a negative impact on their overall capabilities. Moreover, the sampled masks could be extremely unstable, causing inconsistent performance across multiple test runs. Attempts to fine-tune hyperparameters to reduce the weight of the regularization term yielded limited improvements.

These observations suggest that the memorization behavior of large models is more intricately intertwined with their generative capabilities. At the parameter level, this implies that certain weights may simultaneously contribute to both memorization and generalization tasks, making disentanglement by simply selecting weights nearly impossible in larger architectures. Therefore, learning a trade-off through the method in Section 3.5 is a much more realistic and effective way for mitigating memorization.

D Decomposing gate parameters

As discussed in Section 3.2, our approach utilizes stochastic gate functions for each weight parame-

Table 5: Ablation Study on Regularization Term

Model	Method	WikiText Evaluation		Memorization Evaluation				Sparsity
		Test Loss	Test PPL	Fractional ER	Exact ER	Test Loss	Test PPL	
GPT-Neo 125M	Sparsity-only	4.313	74.627	0.060	0.000	3.422	30.627	0.300
	Memo-only	4.187	65.857	0.475	0.307	2.140	8.504	0.007
	Both	4.438	84.563	0.485	0.289	2.406	11.092	0.300
GPT-Neo 1.3B	Sparsity-only	3.453	31.599	0.115	0.001	2.531	12.569	0.300
	Memo-only	3.328	27.886	0.770	0.655	1.336	3.804	0.003
	Both	3.563	35.251	0.729	0.603	1.852	6.370	0.300

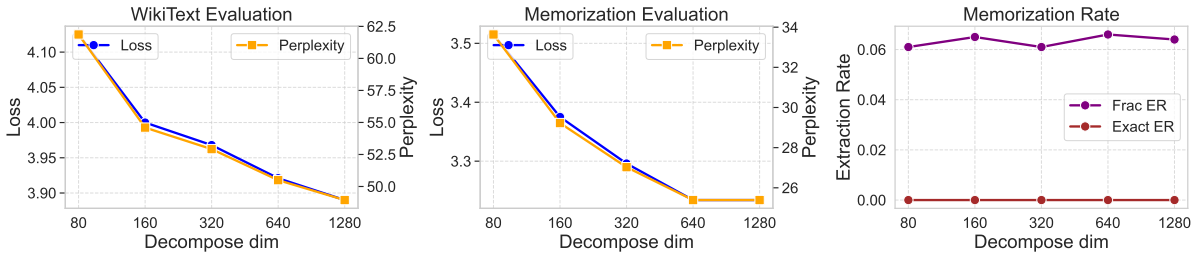


Figure 7: Ablation study of decompose trick on GPT-Neo-2.7B

ter to generate binary masks. While effective, this approach introduces a significant number of additional learnable parameters, which can lead to considerable computational overhead. This issue becomes particularly challenging for large models, such as Pythia-6.9B, where even a system with 8 A100 GPUs struggles to accommodate the resource demands.

To mitigate this problem, we explore a decomposing strategy aimed at reducing the complexity of the gate matrices. Specifically, we break down each linear matrix for the gates into two low-rank sub-matrices, a technique resembles the Low-Rank Adaptation (LoRA) method. This decomposition approach allows us to significantly reduce the number of parameters involved, offering a more computationally efficient alternative. However, this strategy inevitably results in a loss of fine-grained control over the gates, as the ability to manipulate individual gate parameters is diminished. To evaluate the effectiveness of this decomposition strategy, we conduct experiments on larger models, such as GPT-Neo-2.7B. The results of these experiments are presented in Figure 7.

As the dim of the decomposing matrices decreases, we observe a corresponding degradation in the control over gate parameters. Specifically, when the decomposition dimension reaches 1280, which splits the original gate matrix into two low-rank matrices with an equal number of parameters, the final performance of the pruned model deviates significantly from that of the unpruned model and also the pruned model with full gate matrices. This indicates that while the decomposition strategy reduces computational overhead, it also incurs a performance cost.

These findings suggest that further exploration is needed to minimize the performance loss when applying the decomposition technique. Future work could focus on optimizing the decomposition process or integrating additional techniques to better preserve fine-grained control while maintaining computational efficiency.

E Experiment Details

Hyperparameters. All main experiments are conducted with 30% pruning sparsity. We basically use AdamW optimizer with weight decay = 0.001 and cosine annealing scheduler. For GPT-Neo family, we set $\lambda_s = 160, \lambda_m = 1$ for amplification, $\lambda_s = 120, \lambda_m = 1$ for suppression, and train with learning rate = 3e-3. For Pythia family, we set $\lambda_s = 240, \lambda_m = 1$ for amplification, $\lambda_s = 160, \lambda_m = 1$ for suppression, and train with learning rate = 2e-3. Learning threshold t is determined by the model scale and the loss during training. For instance, we set $t = 4.0$ for GPT-Neo-125M, $t = 2.0$ for GPT-Neo-1.3B, GPT-Neo-2.7B, Pythia-410M and Pythia-1.4B, $t = 1.5$ for Pythia-2.8B.

Training Environment. We implement our algorithm in PyTorch and use the HuggingFace Transformers for loading target models and calibration datasets. All experiments on smaller models (with

capacities under 7B parameters) were conducted on a system equipped with 8 NVIDIA A6000 GPUs, each providing 48GB of memory. For experiments involving larger models, we utilized a system with 8 NVIDIA A100 GPUs.