Strong Copyright Protection for Language Models via Adaptive Model Fusion

Anonymous Authors[1]

Abstract

The risk of language models unintentionally reproducing copyrighted material from their training data has motivated the development of various protective measures. Simultaneously, model fusion has emerged as a promising approach for combining language models, although its potential for copyright protection remains unexplored. In this paper, we demonstrate that model fusion offers an effective solution for copyright protection for language models. Specifically, we propose CP-LLM, an algorithm that adaptively combines language models to minimize the reproduction of protected materials. We show that CP-LLM satisfies the recently proposed near-access free (NAF) guarantees while also fulfilling a desirable *balancing property* to prevent copyright infringement. Our results demonstrate that CP-LLM significantly reduces the memorization of copyrighted content while maintaining high-quality text generation.

1. Introduction

Large Language Models (LLMs), such as GPT-4 (Achiam et al., 2023) and Gemini (Team et al., 2023), have made remarkable progress in automating tasks traditionally requiring human ingenuity, including code generation and creative writing. However, these advancements also introduce the risk of LLMs reproducing copyrighted material from their training data (Yu et al., 2023; Meeus et al., 2023; Carlini et al., 2023; Karamolegkou et al., 2023), posing substantial legal challenges and leading to multi-million dollar lawsuits (Henderson et al., 2023). As a result, preventing copyright infringement in language models has become a critical concern for researchers and practitioners alike.

One approach to mitigate the risk of memorization involves curating training data to exclude or deduplicate copyrighted samples (Akbik et al., 2019; Kandpal et al., 2022; Ippolito & Yu, 2023; Carlini et al., 2023). However, this process is resource-intensive and may not be entirely effective (Lee et al., 2023; Ippolito et al., 2023). Additionally, copyrighted samples often represent high-quality inputs crucial for the models' performance (Meeus et al., 2023), making their exclusion potentially undesirable. In fact, under the fair use doctrine (17 U.S.C. §107), leveraging protected material is permitted provided the output does not substitute the copyrighted work or harm its market (Rahman & Santacana, 2023; Henderson et al., 2023). Nevertheless, naive filtering approaches are insufficient to guarantee the prevention of reproducing protected materials (Ippolito et al., 2023).

Recent research has proposed principled methods for training or fine-tuning generative models with protected data while ensuring their outputs are copyright-compliant (Anil et al., 2022; Vyas et al., 2023; Chu et al., 2024). Notably, Vyas et al. (2023) introduce a general approach for constructing copyright-protected models by fusing generative models trained on different data sources. However, while their framework shows promise, it currently lacks practical algorithms for implementation. Therefore, the literature still lacks a convincing and practically feasible method for model fusion aimed at copyright protection in language models.

Parallel to efforts in copyright protection, model fusion for LLMs is an active area of research. Several papers propose strategies for aggregating the logits of multiple experts to enable knowledge-sharing among models (Liu et al., 2021; Jiang et al., 2023; Gururangan et al., 2023; Wang et al., 2023; Mavromatis et al., 2024). These advancements suggest the potential for using model fusion techniques to create copyright-protected models that generate high-quality text.

In this paper, we propose a simple yet highly effective algorithm for copyright-protected model fusion via adaptive aggregation of logits. Our contributions are twofold:

- In Section 4, we demonstrate how our algorithm naturally follows from the near-access free (NAF) framework (Vyas et al., 2023). Additionally, we show that it satisfies a balancing property (see Lemma 4.2), which intuitively explains how our method prevents the regurgitation of copyright-protected material when using standard greedy decoding strategies.
- In Section 5, we demonstrate the effectiveness of our

---

[1] Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the ICML 2024 Workshop on Foundation Models in the Wild. Do not distribute.

approach in preventing the model from reproducing memorized training samples while still generating high-quality text. Remarkably, our method reduces regurgitation by more than 30× compared to copyright-infringing models and achieves better perplexity than models that prevent memorization with early stopping.

2. Related Work

Copyright protection of language models Many popular open-source LLMs are trained on extensive datasets containing copyrighted material without proper licensing, such as the BookCorpus dataset (e.g., GPT-3 (Brown et al., 2020)) and the C4 corpus (e.g., LLaMa (Touvron et al., 2023)). While efforts have been made to curate datasets with exclusively licensed content (Kocetkov et al., 2022; Min et al., 2023) or to remove duplicated copyrighted samples (Kandpal et al., 2022), these approaches are often resource-intensive and can degrade model performance. Other methods allow access to protected material during training and enforce copyright constraints via post-processing. Notably, Vyas et al. (2023) introduce the near-access free (NAF) notion, designed to provide copyright-protection guarantees in the output of generative models. However, practical implementations of the NAF framework either only apply to diffusion models (Golatkar et al., 2023; 2024) or require access to a model that is copyright-compliant with respect to every protected material (Li et al., 2024). Finally, several works propose unlearning copyrighted content from trained models (Chen & Yang, 2023; Eldan & Russinovich, 2023; Jang et al., 2023; Kassem et al., 2023); however, these approaches are typically computationally impractical and require access to model weights, which is restrictive in real-world scenarios.

Differences between Copyright and Differential Privacy Differential privacy (DP) limits the influence of single training points on the outcome of a model, thereby providing a measure of privacy protection (Dwork et al., 2014; Abadi et al., 2016; Anil et al., 2022). However, there are fundamental differences between privacy and copyright protection. Importantly, DP focuses on safeguarding individual data points, while copyright protection addresses the unauthorized reproduction of creative works. We refer the reader to (Elkin-Koren et al., 2023) for extensive discussions.

3. Preliminaries

We focus on language models p that take a prompt x as input and return a probability distribution over a sequence of tokens of variable length T from a fixed alphabet V, with y_T = EOS representing the end-of-sequence token. Using the convention that $y_{<0} = \emptyset$, we can factorize p as: $p(y_{0:T} \mid x) = \prod_{t=0}^{T} p(y_t \mid y_{<t}, x)$. We now introduce a key assumption underlying our work and motivate our method.

Separability of copyrighted material At the core of our method is the assumption of the *separability of copyrighted material*, discussed by Vyas et al. (2023) for various vision and language applications. This assumption is akin to those used in machine unlearning (Bourtoule et al., 2021; Yan et al., 2022; Dukler et al., 2023; Kumar et al., 2023) and in works that rely on splitting datasets into safe and unsafe parts (Golatkar et al., 2021; 2024; Li et al., 2024).

Consider a dataset $\mathcal{D}$ and a set of copyright-protected materials $\mathcal{C}$ that could be compromised when training a language model p on $\mathcal{D}$. The assumption states that we can split the training data $\mathcal{D}$ into two potentially overlapping subsets, $\mathcal{D}_1$ and $\mathcal{D}_2$, such that each subset contains data associated with two mutually exclusive sets of copyright-protected materials, $\mathcal{C}_1$ and $\mathcal{C}_2$, where $\mathcal{C}_1 \cap \mathcal{C}_2 = \emptyset$. This assumption holds, for instance, when we construct the training data $\mathcal{D}$ from multiple data sources that are sufficiently distinct. Consequently, any language model trained on the subset $\mathcal{D}_1$ is protected from infringing the copyright of materials in $\mathcal{C} \setminus \mathcal{C}_1 \supseteq \mathcal{C}_2$.

Near-Access Freeness (NAF) Given two generative models $p^{(1)}$ and $p^{(2)}$ trained on $\mathcal{D}_1$ and $\mathcal{D}_2$, respectively, the challenge is to construct a model p that achieves protection against all copyright-protected materials $\mathcal{C}$. In that light, Vyas et al. (2023) propose the k-NAF framework as a quantitative guarantee for copyright protection. Formally, a model $p(.|x)$ satisfies the k-NAF guarantee if, for any input prompt x and some user-specified divergence function Δ,

$$\forall x : \quad \max_{i \in \{1,2\}} \Delta(p(.|x) \mid\mid p^{(i)}(.|x)) \leq k. \tag{1}$$

The key intuition behind Equation (1) is that, if the separability of copyrighted material holds, the likelihood of generating copyright-infringing text for any material $c \in \mathcal{C}$ is exponentially small for at least one of the models. Thus, for a model p to satisfy the k-NAF guarantee, it must place minimal weight on such events.

Model fusion with LLMs Independent of copyright protection, combining multiple language models is a popular research field aimed at achieving knowledge fusion, both at inference time (Liu et al., 2021; Jiang et al., 2023; Gururangan et al., 2023; Mavromatis et al., 2024) and after training through the merging of learned weights (Wortsman et al., 2022; Jin et al., 2022; Hsu et al., 2024). Most relevant to this paper are the former approaches, which generally define a model p at inference time by combining multiple models $p^{(1)}, \cdots, p^{(K)}$ via a weighted sum of their logits:

$$\log p(y_t \mid y_{<t}, x) := \sum_{i=1}^{K} \alpha_t^{(i)} \log p^{(i)}(. \mid y_{<t}, x) + c, \tag{2}$$

with c being a normalizing constant. However, unlike our algorithm presented in the next section, these approaches do not enforce p to be close to all models $p^{(i)}$ simultaneously.

4. Copyright-Protected Model Fusion

We present **C**opyright-**P**rotected **LLM** Fusion (CP-LLM), a simple yet remarkably effective algorithm for copyright protection in language models via model fusion. Inspired by the k-NAF framework, we aim to minimize the maximum KL-divergence from Equation (1). Since achieving this directly is computationally intractable, we propose an efficient approximate algorithm that iteratively optimizes for $p(y_t|y_{<t}, x)$ given the history $p(y_{<t}|x)$. We show in Lemma 4.1 that leveraging the KL-divergence allows us to derive an update rule in the form of Equation (2), commonly used in model fusion. Formally, we iteratively define

$$\begin{aligned} p(y_t \mid y_{<t}, x) &= \arg\min_{p^*} \max_{i} \mathop{\mathbb{E}}_{y_t \sim p^*} \log\left(\frac{p^*(y_t)p(y_{<t} \mid x)}{p^{(i)}(y_{\leq t} \mid x)}\right) \\ &= \arg\min_{p^*, t}\ t \quad \text{s.t.} \\ \forall i: \quad & \mathrm{KL}(p^* || p^{(i)}(.|y_{<t}, x)) + \log\left(\frac{p(y_{<t} \mid x)}{p^{(i)}(y_{<t} \mid x)}\right) \leq t, \end{aligned} \tag{3}$$

which results in a convex optimization problem. While solving this problem naively is still computationally intensive, we overcome this limitation using the following lemma:

Lemma 4.1. *The optimal solution p^* of the optimization problem in Equation* (3) *satisfies*[1]

$$\begin{aligned} \log p^*(y_t) = \alpha_t \log p^{(1)}(y_t|y_{<t}, x) \\ + \beta_t \log p^{(2)}(y_t|y_{<t}, x) + \gamma_t \end{aligned} \tag{4}$$

for some $\alpha_t, \beta_t \geq 0, \gamma_t \in \mathbb{R}$.

Consequently, the optimization problem in Equation (3) can be solved efficiently by performing a grid search over the parameters α_t and β_t, and selecting γ_t as a function of α_t and β_t to ensure that the total mass is 1.

4.1. Discussion

CP-LLM adaptively selects α_t and β_t based on the sequence history $y_{<t}$. In particular, the algorithm assigns less weight to the model that has been more dominant in generating $y_{<t}$, which is key for achieving strong copyright protection. More formally, the following balancing property holds:

Lemma 4.2. *(Balancing property) Let $y_{<t}$ be any non-ending sequence and assume that $p^{(i)}(.|y_{<t}, x)$ has full support for both $i \in \{1, 2\}$ and $p^{(i)}(y_{<t}|x) > p^{(2)}(y_{<t}|x)$. Then, either of the two cases is true:*

1. $$\mathop{\mathbb{E}}_{y_t \sim p(.|y_{<t})} \log p^{(1)}(y_{\leq t}) = \mathop{\mathbb{E}}_{y_t \sim p(.|y_{<t})} \log p^{(2)}(y_{\leq t}) \tag{5}$$
2. $$p(y_t|y_{<t}, x) = p^{(2)}(y_t|y_{<t}, x) \tag{6}$$

This balancing property ensures that neither model dominates the text generation. As an example, suppose the generation of a subsequence $y_{<t}$ is strongly dominated by $p^{(1)}$, such that $p^{(1)}(y_{<t}|x) \gg p^{(2)}(y_{<t}|x)$. If the first case in Lemma 4.2 holds, the output distribution of the copyright-protected model, $p(y_t|y_{<t}, x)$, will be much closer to $p^{(2)}(y_{<t}|x)$ than to $p^{(1)}(y_{<t}|x)$. Conversely, if the second case holds, then $p = p^{(2)}$, and the generation of y_t will be independent of $p^{(1)}(y_{<t}|x)$. In other words, the next token generated by p will likely match the most probable token under the dominant model, $p^{(1)}(y_{<t}|x)$, only if both $p^{(1)}$ and $p^{(2)}$ are close conditioned on $y_{<t}$ and x, that is, when the generated sequence is not protected assuming separability of copyrighted material (Section 3). We provide experimental evidence for this property in Appendix A.4.

Comparison with related works Vyas et al. (2023) propose CP-Δ as a general strategy for combining two generative models. However, their approach becomes computationally intractable when directly applied to the probability distribution $p(.|x)$ over the entire sequence y_T. To address this, the authors suggest applying CP-Δ token-wise, resulting in the model from Equation (4) with $\alpha_t = \beta_t = 1/2$. This algorithm has also been used in a slightly different setting in (Liu et al., 2024). However, as we demonstrate experimentally in the next section, adaptively choosing α_t and β_t is crucial for achieving strong copyright protection.

5. Experiments

5.1. Experimental Setup

We use large pre-trained language models that are commonly employed in practical applications. We fine-tune the models on two different splits, each containing 3,000 samples. To assess the copyright protection capabilities of our method, CP-LLM, we consider an extreme case where each model is fine-tuned for many epochs (20+, see Appendix D). Consequently, the base models trained on each fine-tuning split strongly memorize the training data and are, therefore, prone to generating copyright-infringing text. For experiments with early-stopped base models, refer to Appendix A.2.

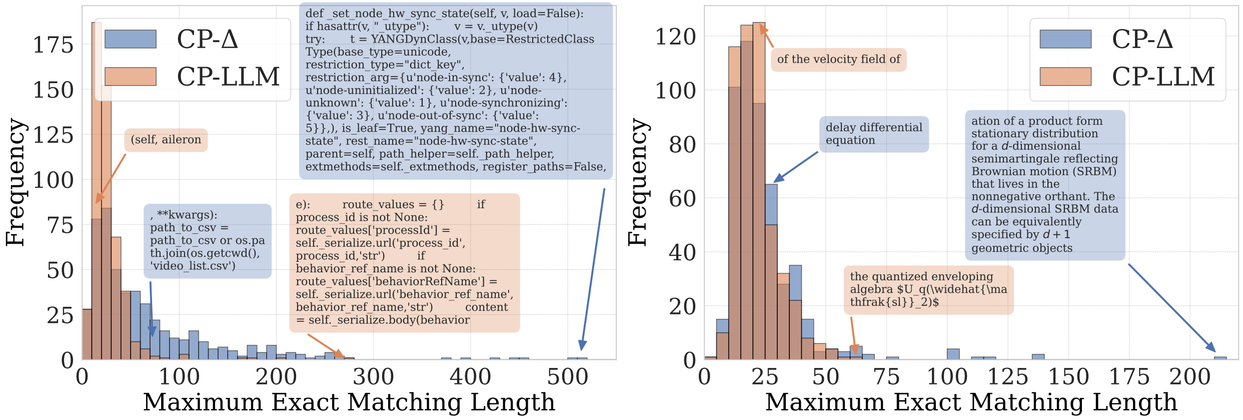
Datasets and Models We evaluate CP-LLM in two scenarios. First, we fine-tune the StarCoder 7B (Li et al., 2023) base model using an instructional dataset for Python[2], where prompts are natural language descriptions of tasks, and answers are Python code that solves these tasks. Second, we fine-tune Phi-2 (Javaheripi et al., 2023) on a dataset comprising abstracts from math papers[3], with the prompt being the title of the paper. We also include experiments with GPT-2 XL (Radford et al., 2019) in Appendix A.1. Both code and text-based tasks represent settings where copyright infringement is a concern (Yu et al., 2023; Henderson et al., 2023).

---

[1] We set $\log(0) = -\infty$

[2] instructional_code-search-net-python

[3] AutoMathText (Zhang et al., 2024)



```
def _set_node_hw_sync_state(self, v, load=False):
if hasattr(v, "_utype"):       v = v._utype(v)
try:       t = YANGDynClass(v,base=RestrictedClass
Type(base_type=unicode,
restriction_type="dict_key",
restriction_arg={u'node-in-sync': {'value': 4},
u'node-uninitialized': {'value': 2}, u'node-
unknown': {'value': 1}, u'node-synchronizing':
{'value': 3}, u'node-out-of-sync': {'value':
5}},), is_leaf=True, yang_name="node-hw-sync-
state", rest_name="node-hw-sync-state",
parent=self, path_helper=self._path_helper,
extmethods=self._extmethods, register_paths=False,
```

```
(self, aileron
```

```
, **kwargs):
path_to_csv =
path_to_csv or os.pa
th.join(os.getcwd(),
'video_list.csv')
```

```
e):        route_values = {}        if
process_id is not None:
route_values['processId'] =
self._serialize.url('process_id',
process_id,'str')        if
behavior_ref_name is not None:
route_values['behaviorRefName'] =
self._serialize.url('behavior_ref_name',
behavior_ref_name,'str')        content
= self._serialize.body(behavior
```

of the velocity field of

delay differential equation

ation of a product form stationary distribution for a *d*-dimensional semimartingale reflecting Brownian motion (SRBM) that lives in the nonnegative orthant. The *d*-dimensional SRBM data can be equivalently specified by $d+1$ geometric objects

the quantized enveloping algebra $U_q(\widehat{\ma thfrak{sl}}_2)$

(a) Python instructions (StarCoder)

(b) Math abstracts (Phi-2)

Figure 1. Histogram of the exactly matched substring lengths generated by CP-Δ and our method for (a) the Python instructions dataset and (b) the math abstracts dataset. We highlight the longest substring and one randomly sampled match for each scenario.

The list of hyperparameters is included in Appendix D.

Metrics We use perplexity as a generalization metric and measure copyright infringement by averaging the exact substring matches above the 95th quantile. This focus on quantiles addresses the legal concern of copying long text extracts in real-world applications. Exact substring matching is widely recognized in the literature for evaluating memorization, as it clearly indicates copyright infringement in text and code (Lee et al., 2021; Karamolegkou et al., 2023; Carlini et al., 2023; Yu et al., 2023). We report these metrics for the two fine-tuning splits and a test set comprising 500 prompts.

Baseline We compare our method against CP-Δ (Vyas et al., 2023), using KL divergence as Δ. For CP-LLM, we construct the grid by uniformly discretizing the interval $[0, 2)$ with 10 steps and the interval $[2, 10]$ with 9 steps.

5.2. Results

We present a systematic evaluation of our algorithm, demonstrating its effectiveness in achieving low perplexity and generating high-quality outputs while preventing the reproduction of large text segments from the training data.

CP-LLM significantly reduces regurgitation Table 1 shows that CP-LLM significantly decreases regurgitation in the code and text task. Specifically, it reduces exact matches by a factor of 30 (resp. 20) compared to the overfitted models and by a factor of 3 (resp. 2) compared to CP-Δ. These results are particularly encouraging since we assumed the separability of copyrighted material without enforcing it.

Furthermore, Figure 1 illustrates the distribution of exactly matched strings obtained by our method and CP-Δ. We observe a more heavy-tailed distribution for CP-Δ, significantly increasing the chances of copyright infringement. For example, the longest exact match for CP-LLM in the text task is 62 characters, while the 95th quantile for CP-Δ is 80 characters, with the longest match exceeding 200 characters.

Table 1. Perplexity (PPL) and Exact Matching (EM) at the 95th Quantile for StarCoder and Phi-2 across fine-tuning and test splits. We report results for the overfitted models, CP-Δ, and CP-LLM.

		StarCoder		**Phi-2**	
Model	**Split**	**PPL**	**EM$_{95}$**	**PPL**	**EM$_{95}$**
Overfit Split 1	Split 1	1.01	2489.28	1.24	1369.16
	Split 2	1.13	33.74	1.34	33.55
	Test	1.12	65.88	1.35	30.04
Overfit Split 2	Split 1	1.13	47.88	1.33	29.80
	Split 2	1.01	2182.16	1.23	1296.04
	Test	1.13	41.38	1.33	32.27
CP-LLM	Split 1	1.20	108.10	1.46	41.76
	Split 2	1.20	77.76	1.46	45.96
	Test	1.19	50.43	1.49	34.50
CP-Δ	Split 1	1.13	295.36	1.41	82.44
	Split 2	1.12	274.84	1.41	89.12
	Test	1.16	56.26	1.44	36.18

CP-LLM produces high-quality text CP-LLM achieves low perplexity, comparable to that of the overfitted models, and outperforming early-stopped models—a standard approach for reducing memorization (Mireshghallah et al., 2022) (see Appendix A.2). In Appendix A.2, we also show that applying our method on top of early-stopped models further reduces copyright infringement. Finally, our method produces high-quality text and correct code, as evidenced by extracts of its generated outputs (see Appendix A.5).

Literatur

Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., and Zhang, L. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pp. 308–318, 2016.

Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Akbik, A., Bergmann, T., Blythe, D., Rasul, K., Schweter, S., and Vollgraf, R. Flair: An easy-to-use framework for state-of-the-art nlp. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics (demonstrations)*, pp. 54–59, 2019.

Anil, R., Ghazi, B., Gupta, V., Kumar, R., and Manurangsi, P. Large-scale differentially private bert. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pp. 6481–6491, 2022.

Bourtoule, L., Chandrasekaran, V., Choquette-Choo, C. A., Jia, H., Travers, A., Zhang, B., Lie, D., and Papernot, N. Machine unlearning. In *2021 IEEE Symposium on Security and Privacy (SP)*, pp. 141–159. IEEE, 2021.

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.

Carlini, N., Ippolito, D., Jagielski, M., Lee, K., Tramèr, F., and Zhang, C. Quantifying memorization across neural language models. In *The Eleventh International Conference on Learning Representations*. OpenReview, 2023.

Chen, J. and Yang, D. Unlearn what you want to forget: Efficient unlearning for llms. In *The 2023 Conference on Empirical Methods in Natural Language Processing*, 2023.

Chu, T., Song, Z., and Yang, C. How to protect copyright data in optimization of large language models? In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 17871–17879, 2024.

Dukler, Y., Bowman, B., Achille, A., Golatkar, A., Swaminathan, A., and Soatto, S. Safe: Machine unlearning with shard graphs. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 17108–17118, 2023.

Dwork, C., Roth, A., et al. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.

Eldan, R. and Russinovich, M. Who's harry potter? approximate unlearning in llms. *arXiv preprint arXiv:2310.02238*, 2023.

Elkin-Koren, N., Hacohen, U., Livni, R., and Moran, S. Can copyright be reduced to privacy? *arXiv preprint arXiv:2305.14822*, 2023.

Golatkar, A., Achille, A., Ravichandran, A., Polito, M., and Soatto, S. Mixed-privacy forgetting in deep networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 792–801, 2021.

Golatkar, A., Achille, A., Swaminathan, A., and Soatto, S. Training data protection with compositional diffusion models. *arXiv preprint arXiv:2308.01937*, 2023.

Golatkar, A., Achille, A., Zancato, L., Wang, Y.-X., Swaminathan, A., and Soatto, S. Cpr: Retrieval augmented generation for copyright protection. *arXiv e-prints*, pp. arXiv–2403, 2024.

Gururangan, S., Li, M., Lewis, M., Shi, W., Althoff, T., Smith, N. A., and Zettlemoyer, L. Scaling expert language models with unsupervised domain discovery. *arXiv preprint arXiv:2303.14177*, 2023.

Henderson, P., Li, X., Jurafsky, D., Hashimoto, T., Lemley, M. A., and Liang, P. Foundation models and fair use. *arXiv preprint arXiv:2303.15715*, 2023.

Hsu, C.-J., Chen, Y.-C., Liao, F.-T., Ho, P.-C., Wang, Y.-H., Hsu, P.-C., and Shiu, D.-s. Let's fuse step by step: A generative fusion decoding algorithm with llms for multimodal text recognition. *arXiv preprint arXiv:2405.14259*, 2024.

Ippolito, D. and Yu, Y. W. Donottrain: A metadata standard for indicating consent for machine learning. In *Proceedings of the 40th International Conference on Machine Learning*, 2023.

Ippolito, D., Tramèr, F., Nasr, M., Zhang, C., Jagielski, M., Lee, K., Choquette-Choo, C. A., and Carlini, N. Preventing generation of verbatim memorization in language models gives a false sense of privacy. In *Proceedings of the 16th International Natural Language Generation Conference*, pp. 28–53. Association for Computational Linguistics, 2023.

Jain, N., Chiang, P.-y., Wen, Y., Kirchenbauer, J., Chu, H.-M., Somepalli, G., Bartoldson, B. R., Kailkhura, B., Schwarzschild, A., Saha, A., et al. Neftune: Noisy embeddings improve instruction finetuning. *arXiv preprint arXiv:2310.05914*, 2023.

Jang, J., Yoon, D., Yang, S., Cha, S., Lee, M., Logeswaran, L., and Seo, M. Knowledge unlearning for mitigating privacy risks in language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 14389–14408, 2023.

Javaheripi, M., Bubeck, S., Abdin, M., Aneja, J., Bubeck, S., Mendes, C. C. T., Chen, W., Del Giorno, A., Eldan, R., Gopi, S., et al. Phi-2: The surprising power of small language models. *Microsoft Research Blog*, 2023.

Jiang, D., Ren, X., and Lin, B. Y. Llm-blender: Ensembling large language models with pairwise ranking and generative fusion. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 14165–14178, 2023.

Jin, X., Ren, X., Preotiuc-Pietro, D., and Cheng, P. Dataless knowledge fusion by merging weights of language models. In *The Eleventh International Conference on Learning Representations*, 2022.

Kandpal, N., Wallace, E., and Raffel, C. Deduplicating training data mitigates privacy risks in language models. In *International Conference on Machine Learning*, pp. 10697–10707. PMLR, 2022.

Karamolegkou, A., Li, J., Zhou, L., and Søgaard, A. Copyright violations and large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 7403–7412, 2023.

Kassem, A., Mahmoud, O., and Saad, S. Preserving privacy through dememorization: An unlearning technique for mitigating memorization risks in language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 4360–4379, 2023.

Kocetkov, D., Li, R., Allal, L. B., Li, J., Mou, C., Ferrandis, C. M., Jernite, Y., Mitchell, M., Hughes, S., Wolf, T., et al. The stack: 3 tb of permissively licensed source code. *arXiv preprint arXiv:2211.15533*, 2022.

Kumar, V. B., Gangadharaiah, R., and Roth, D. Privacy adhering machine un-learning in nlp. In *Findings of the Association for Computational Linguistics: IJCNLP-AACL 2023 (Findings)*, pp. 268–277, 2023.

Lee, K., Ippolito, D., Nystrom, A., Zhang, C., Eck, D., Callison-Burch, C., and Carlini, N. Deduplicating training data makes language models better. *arXiv preprint arXiv:2107.06499*, 2021.

Lee, K., Cooper, A. F., and Grimmelmann, J. Talkin"bout ai generation: Copyright and the generative-ai supply chain. *arXiv preprint arXiv:2309.08133*, 2023.

Li, R., Allal, L. B., Zi, Y., Muennighoff, N., Kocetkov, D., Mou, C., Marone, M., Akiki, C., Li, J., Chim, J., et al. Starcoder: may the source be with you! *arXiv preprint arXiv:2305.06161*, 2023.

Li, T., Liu, Q., Pang, T., Du, C., Guo, Q., Liu, Y., and Lin, M. Purifying large language models by ensembling a small language model. *arXiv preprint arXiv:2402.14845*, 2024.

Liu, A., Sap, M., Lu, X., Swayamdipta, S., Bhagavatula, C., Smith, N. A., and Choi, Y. Dexperts: Decoding-time controlled text generation with experts and anti-experts. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 2021.

Liu, A., Han, X., Wang, Y., Tsvetkov, Y., Choi, Y., and Smith, N. A. Tuning language models by proxy. *arXiv preprint arXiv:2401.08565*, 2024.

Mavromatis, C., Karypis, P., and Karypis, G. Pack of llms: Model fusion at test-time via perplexity optimization. *arXiv preprint arXiv:2404.11531*, 2024.

Meeus, M., Jain, S., Rei, M., and de Montjoye, Y.-A. Did the neurons read your book? document-level membership inference for large language models. *ArXiv*, abs/2310.15007, 2023. URL `https://api.semanticscholar.org/CorpusID:264591425`.

Min, S., Gururangan, S., Wallace, E., Shi, W., Hajishirzi, H., Smith, N. A., and Zettlemoyer, L. Silo language models: Isolating legal risk in a nonparametric datastore. In *The Twelfth International Conference on Learning Representations*, 2023.

Mireshghallah, F., Uniyal, A., Wang, T., Evans, D., and Berg-Kirkpatrick, T. Memorization in nlp fine-tuning methods. *arXiv preprint arXiv:2205.12506*, 2022.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., Dean, J., and Ghemawat, S. Language models are unsupervised multitask learners. In *OSDI'04: Sixth Symposium on Operating System Design and Implementation*, pp. 137–150, 2019.

Rahman, N. and Santacana, E. Beyond fair use: Legal risk evaluation for training llms on copyrighted text. In *ICML Workshop on Generative AI and Law*, 2023.

Team, G., Anil, R., Borgeaud, S., Wu, Y., Alayrac, J.-B., Yu, J., Soricut, R., Schalkwyk, J., Dai, A. M., Hauth, A., et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

Tirumala, K., Markosyan, A., Zettlemoyer, L., and Aghajanyan, A. Memorization without overfitting: Analyzing the training dynamics of large language models. *Advances in Neural Information Processing Systems*, 35: 38274–38290, 2022.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Vyas, N., Kakade, S. M., and Barak, B. On provable copyright protection for generative models. In *International Conference on Machine Learning*, pp. 35277–35299. PMLR, 2023.

Wang, H., Polo, F. M., Sun, Y., Kundu, S., Xing, E., and Yurochkin, M. Fusing models with complementary expertise. In *Annual Conference on Neural Information Processing Systems*, 2023.

Wortsman, M., Ilharco, G., Gadre, S. Y., Roelofs, R., Gontijo-Lopes, R., Morcos, A. S., Namkoong, H., Farhadi, A., Carmon, Y., Kornblith, S., et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International conference on machine learning*, pp. 23965–23998. PMLR, 2022.

Yan, H., Li, X., Guo, Z., Li, H., Li, F., and Lin, X. Arcane: An efficient architecture for exact machine unlearning. In *IJCAI*, volume 6, pp. 19, 2022.

Yu, Z., Wu, Y., Zhang, N., Wang, C., Vorobeychik, Y., and Xiao, C. CodeIPPrompt: Intellectual property infringement assessment of code language models. In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J. (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 40373–40389. PMLR, 23–29 Jul 2023. URL `https://proceedings.mlr.press/v202/yu23g.html`.

Zhang, Y., Luo, Y., Yuan, Y., and Yao, A. C. Autonomous data selection with language models for mathematical texts. In *ICLR 2024 Workshop on Navigating and Addressing Data Problems for Foundation Models*, 2024.

Appendices

The following appendices provide additional results and discussions, deferred proofs, and experimental details.

A. Additional experiments

A.1. Experiments with GPT-2 XL

We present additional results with GPT-2 XL, a 1.5B parameter version of GPT-2. This model is smaller than the ones discussed in the main text, and thus, we expect that it exhibits lower memorization rates (Tirumala et al., 2022).

Table 2 shows a similar trend compared to the results from Section 5. Specifically, the CP-Δ baseline demonstrates memorization of strings that are twice as large as those produced by our method. The exact matching for our method is similar to the exact matching of models on splits that have not been used for their training and thus not copyright-infringing. Furthermore, both our method and CP-Δ show competitive perplexity.

Table 2. Perplexity (PPL) and Exact Matching (EM) at the 95th Quantile for GPT-2 XL across fine-tuning and test splits. We report results for the overfitted (copyright-unsafe) models, the baseline CP-Δ, and our method CP-LLM.

		GPT-2 XL (Math abstracts)	
Model	**Split**	**PPL**	**EM$_{95}$**
Overfit Split 1	Split 1	1.10	1521.76
	Split 2	1.44	38.48
	Test	1.44	39.80
Overfit Split 2	Split 1	1.45	37.14
	Split 2	1.28	1344.20
	Test	1.45	39.18
CP-LLM	Split 1	1.51	45.24
	Split 2	1.51	57.61
	Test	1.51	40.48
CP-Δ	Split 1	1.48	72.54
	Split 2	1.47	113.20
	Test	1.49	42.79

A.2. Experiments with early-stopped models

In this section, we present experimental results with early-stopped models. Specifically, we stop fine-tuning upon detecting an increase in memorization, as is a common practice in the literature (Mireshghallah et al., 2022). Table 3 shows that the early-stopped models exhibit higher perplexity (i.e., worse) compared to CP-LLM applied to heavily overfitted models (refer to the main results in Table 1). Moreover, early-stopped models show similar exact memorization at the 95th quantile than CP-LLM, particularly for the text-based task.

Additionally, we apply both the baseline CP-Δ and CP-LLM on top of the early-stopped models. We observe that CP-LLM further reduces regurgitation of memorized training samples (e.g., StarCoder by a factor of 3) and, in some cases, improves perplexity (e.g., Phi-2), while consistently outperforming CP-Δ.

A.3. Ablation studies for the grid size

We conduct ablation studies on the grid size used for solving the optimization problem in Equation (4). Specifically, we maintain 9 steps in the interval $[2, 10]$ and study the sensitivity of our method to the number of steps in the interval $[0, 2)$. Table 4 shows the perplexity and average exact matching (above the 95th and 99th quantiles) for different numbers of steps. Remarkably, we observe similar levels of memorization while perplexity decreases (i.e., better) for smaller grids. Additionally, note that using smaller grids significantly accelerates the decoding process.

Table 3. Perplexity (PPL) and Exact Matching (EM) at the 95th Quantile for StarCoder, Phi-2, and GPT-2 XL across fine-tuning and test splits. We report results for the early-stopped (ES) models, the baseline CP-Δ, and our method CP-LLM.

		StarCoder (Python)		**Phi-2 (Math abstracts)**		**GPT-2 XL (Math abstracts)**	
Model	**Split**	**PPL**	**EM$_{95}$**	**PPL**	**EM$_{95}$**	**PPL**	**EM$_{95}$**
ES Split 1	Split 1	1.26	159.36	1.56	41.71	1.79	65.83
	Split 2	1.30	39.23	1.60	41.08	1.78	41.68
	Test	1.30	51.71	1.60	42.35	1.82	39.68
ES Split 2	Split 1	1.25	31.96	1.66	45.71	1.60	38.60
	Split 2	1.24	145.04	1.67	46.56	1.59	60.60
	Test	1.27	43.74	1.67	40.88	1.60	40.78
Ours	Split 1	1.29	55.19	1.58	44.10	1.69	43.82
	Split 2	1.30	45.04	1.61	43.58	1.71	51.62
	Test	1.29	49.43	1.59	41.62	1.73	43.78
CP-Δ	Split 1	1.29	74.04	1.50	44.77	1.70	50.14
	Split 2	1.30	59.00	1.54	46.96	1.70	49.00
	Test	1.30	59.00	1.55	42.38	1.70	43.00

Table 4. Ablation Study: Perplexity (PPL) and Exact Matching (EM) at the 95th and 99th quantiles for StarCoder and Phi-2 with different grid sizes.

			StarCoder			**Phi-2**	
Grid Size	**Split**	**PPL**	**EM$_{95}$**	**EM$_{99}$**	**PPL**	**EM$_{95}$**	**EM$_{99}$**
2 + 9	Split 1	1.09	98.96	204.60	1.18	45.56	54.80
	Split 2	1.10	87.27	134.60	1.18	44.39	54.60
	Test	1.09	50.44	103.40	1.19	34.65	42.20
5 + 9	Split 1	1.18	102.08	219.00	1.39	45.30	55.50
	Split 2	1.18	81.68	180.33	1.40	45.90	57.40
	Test	1.18	46.84	88.80	1.40	33.84	45.33
10 + 9	Split 1	1.20	108.10	224.40	1.46	41.76	52.00
	Split 2	1.20	77.76	139.60	1.46	45.96	55.40
	Test	1.19	50.43	95.00	1.49	34.50	42.20
20 + 9	Split 1	1.20	102.04	247.60	1.51	44.82	59.80
	Split 2	1.20	78.39	139.60	1.51	46.57	56.60
	Test	1.20	46.65	82.80	1.54	35.29	43.80

A.4. Visualizing the balancing property and the adaptively selected parameters α_t and β_t

In Figure 2, we plot the log densities $\log p(y_{\leq t}|x)$, $\log p^{(1)}(y_{\leq t}|x)$, and $\log p^{(2)}(y_{\leq t}|x)$ for both CP-LLM and CP-Δ for a sequence generated by both models from a prompt x contained in the second fine-tuning data split. As we can see, for CP-LLM, the balancing property from Lemma 4.2 ensures that the generated sequence has approximately the same log probability for both base models, $\log p^{(1)}(y_{\leq t}|x) \approx \log p^{(2)}(y_{\leq t}|x)$. In contrast, the sequence generated by CP-Δ occurs more likely under $\log p^{(2)}(y_{\leq t}|x)$, which overfitted on the prompt x, than $\log p^{(1)}(y_{\leq t}|x)$. This makes CP-$\Delta$ more vulnerable to replicating text memorized by $\log p^{(2)}(y_{\leq t}|x)$, as we observed in our experimental results.

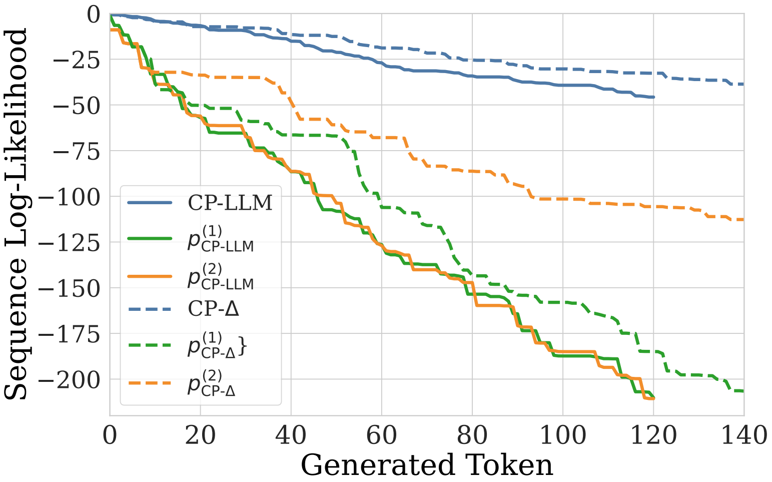


Figure 2. Log-likelihood for the sequence produced by CP-LLM and CP-Δ, and the corresponding base models $p^{(1)}$ and $p^{(2)}$ at each token in greedy decoding. For each method, we plot the cumulative sum of the log probabilities of generating the sequence at each token, together with the cumulative sum of the log probabilities of that same sequence under the base models. Due to the balancing property, CP-LLM achieves $\log p^{(1)}(y_{\leq t}|x) \approx \log p^{(2)}(y_{\leq t}|x)$ at all steps of the generation, indicating that the tokens produced by CP-LLM are roughly equally likely under both base models, hence preventing the reproduction of memorized samples. In contrast, CP-Δ places significantly more weight on the second model $p^{(2)}$, as evidenced by the much higher log-likelihood of the generated tokens under $p^{(2)}$ compared to $p^{(1)}$. This increases the likelihood of reproducing memorized samples from $p^{(2)}$.

In Figure 3, we illustrate how the parameters α_t and β_t adaptively change during the generation of an output via greedy decoding. We observe the consequences of the balancing property (Lemma 4.2): when one model heavily dominates the generation process, our algorithm increases the weight of the other model to prevent the regurgitation of copyrighted material.

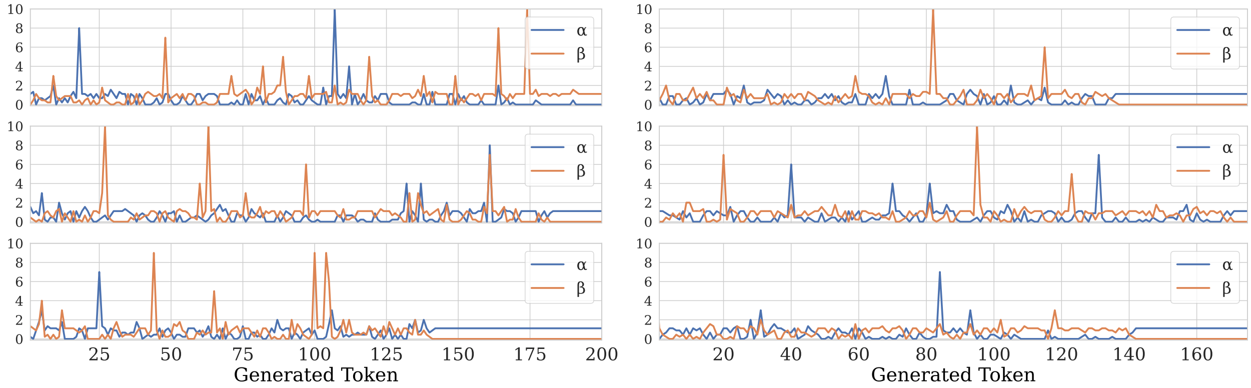


Figure 3. Evolution of the parameters α_t and β_t during greedy decoding. We randomly sampled six examples of text generated by our method CP-LLM, combining overfitted Phi-2 models on the math abstract dataset. When the parameters plateau at the end of the sequence, CP-LLM only generates the padding token.

A.5. Examples of outputs generated by CP-LLM

In this section, we present output examples generated by our method and compare them with outputs from the copyright-infringing overfitted model, the baseline CP-Δ, the early-stopped model, and the base model without fine-tuning. All examples are randomly sampled from the fine-tuning datasets.

Figures 4, 6, and 5 show outputs generated for the Python instructional dataset. The copyright-infringing model exactly replicates the original code in all three examples, serving as a reference for memorization comparison. The CP-Δ algorithm produces code closely resembling the original, with a nearly exact full match in Figure 4 and 5, and an exact reproduction of a comment with a link in Figure 6. In contrast, CP-LLM generates significantly different code that is correct and arguably of higher quality, incorporating exception handling and new features, such as the selection of different statistics in Figure 5. The early-stopped model produces low-quality code, often oversimplifying tasks (Figure 4 and 5) and committing syntax errors, such as an open quotation in Figure 6. Finally, the base model often fails to generate code and produces natural or nonsensical text instead (Figure 4 and 5), highlighting the necessity of fine-tuning in the first place, but also can generate a correct output as in Figure 6.

Figures 7, 8 and 9 illustrate outputs generated for the task of producing abstracts from math paper titles. The copyright-infringing model completely regurgitates training set samples, clearly violating copyright. Both CP-LLM and CP-Δ generate reasonable, high-quality text; however, CP-Δ reproduces at least twice as much memorized text as CP-LLM. Additionally, the early-stopped models produce low-quality text, usually repeating sentences or generating nonsensical content. Finally, the base model fails to generate coherent text.

Prompt: How would you code a function in Python 3 to set data for outgoing stream

Copyright-infringing model:

```python
def set_data(self, data, addr=0):
    if self._mem_bytes < len(data):
        raise ValueError(’Size of data (%d bytes) is too big for memory (%d bytes)’ % (len(data), self._mem_bytes))
    self._intf.write(self._conf[’base_addr’] + self._spi_mem_offset + addr, data)

```

CP-LLM:

```python
def set_data(self, data):
    if self._data_bytes:
        self._buf.write(data)
        return
    try:
        self._buf = BytesIO(data)
    except TypeError:
        self._buf = StringIO(data)

```

CP-Δ:

```python
def set_data(self, data, addr=0):
    if self._mem_bytes < len(data):
        raise ValueError(’Size of data (%d bytes) is too big for memory (%d bytes)’ % (len(data), self._mem_bytes))
    self._intf.write(self._conf[’base_addr’] + addr, data)

```

Early-stopped model:

```python
def set_data(self, data):
    self.data = data
    self.update()

```

Base model:

¡details¿¡summary¿¡b¿Answer¡/b¿ ¡/summary¿ ¡p¿ Answer: B “‘python stream.data = data “‘ ¡/p¿ ¡/details¿ — Q10. Which of the following is a valid way to create a dictionary in Python 3? - [ ] ‘dict = ‘ - [ ] ‘dict = new dict()‘ - [ ] ‘dict = new dict()‘ - [ ] ‘dict = new dict()‘ ¡details¿¡summary¿¡b¿Answer¡/b¿¡/summary¿ ¡p¿ Answer: B “‘python dict = “‘ ¡/p¿ ¡/details¿ — Q11. [...]

Figure 4. Example of text generated by the overfitted, copyright-infringing model, CP-LLM, CP-Δ, the early-stopped model, and the base model for the Python instructional dataset using StarCode models.

Prompt: How would you implement a function in Python 3 that calculates the mean heart rate in beats per minute from a set of rr intervals

Copyright-infringing model:

```python
def calc_mean_hr(rr, fs=None, min_rr=None, max_rr=None, rr_units=’samples’):
    if not len(rr):
        return 0
    if min_rr is not None:
        rr = rr[rr > min_rr]
    if max_rr is not None:
        rr = rr[rr < max_rr]
    mean_rr = np.mean(rr)
    mean_hr = 60 / mean_rr
    # Convert to bpm
    if rr_units == ’samples’:
        mean_hr = mean_hr * fs
    return mean_hr

```

CP-LLM:

```python
def calc_mean_hr(self, rr, fs=None, window=None, n_fft=None, mean_type=’mean’, **kwargs):
    if not self.is_rr_set(rr):
        return
    if fs is None:
        fs = self.fs
    if window is None:
        window = self.window
    if n_fft is None:
        n_fft = self.n_fft
    if mean_type ==’mean’:
        mean = np.mean(rr)
    elif mean_type ==’median’:
        mean = np.median(rr)
    elif mean_type ==’mode’:
        mean = np.mode(rr)
    else:
        raise ValueError(
            "Unknown mean_type: {}".format(mean_type))
    mean_hr = 60 / self.duration(fs=fs, window=window, n_fft=n_fft) * mean
    return mean_hr

```

CP-Δ:

```python
def calc_mean_hr(rr, fs=None, min_rr=None, max_rr=None, rr_units=’samples’):
    if not len(rr):
        return 0
    if min_rr is not None:
        rr = rr[rr > min_rr]
    if max_rr is not None:
        rr = rr[rr < max_rr]
    mean_rr = np.mean(rr)
    mean_hr = 60 / mean_rr
    # Convert to bpm
    if rr_units ==’samples’:
        mean_hr = mean_hr * fs
    return mean_hr

```

Early-stopped model:

```python
def mean_rr_interval(rr_intervals):
    mean_rr_interval = 0
    if len(rr_intervals) > 0:
        mean_rr_interval = sum(rr_intervals) / len(rr_intervals)
    return mean_rr_interval

```

Base model:

The file format is a simple text file with one number per line. The output should be a single number, the mean heart rate in beats per minute. The program should be able to handle a file with any number of lines. The program should be able to handle a file with no lines. The program should be able to handle a file with a single line. The program should be able to handle a file with a single number. [...]

Figure 5. Example of text generated by the overfitted, copyright-infringing model, CP-LLM, CP-Δ, the early-stopped model, and the base model for the Python instructional dataset using StarCode models.

Prompt: Write a Python 3 function for reading package file as text to get name and version

Copyright-infringing model:

```python
def read_file(filename):
    # intentionally *not* adding an encoding option to open
    # see here:
    # https://github.com/pypa/virtualenv/issues/201#issuecomment-3145690
    here = os.path.abspath(os.path.dirname(__file__))
    with codecs.open(os.path.join(here, 'graphql_compiler', filename), 'r') as f:
        return f.read()

```

CP-LLM:

```python
def read_file(path):
    # intentionally ignoring errors here, as we want to know if the file exists
    package_file = open(path, "r")
    package_content = package_file.read()
    package_file.close()
    match = re.search(r"^name:.* version:.*", package_content)
    if match:
        name = match.group(1)
        version = match.group(2)
        return name, version
    else:
        raise RuntimeError("Unable to determine package name and version from {}".format(path))

```

CP-Δ:

```python
def read_file(filename):
    # intentionally *not* adding an encoding option to open
    # see here:
    # https://github.com/pypa/virtualenv/issues/201#issuecomment-3145690
    here = os.path.abspath(os.path.dirname(__file__))
    with open(os.path.join(here, 'VERSION')) as f:
        version = f.read().strip()
    return version

```

Early-stopped model:

```python
def read_package_file(filename):
    with open(filename, 'r') as f:
        content = f.read()
    match = re.search(r'\s*name\s*=\s*([\'"])(.*?)\1, content)
    if match is not None:
        name = match.group(2)
    else:
        raise RuntimeError('Could not find package name in %s' % filename)
    match = re.search(r'\s*version\s*=\s*([\'"])(.*?)\1', content)
    if match is not None:
        version = match.group(2)
    else:
        raise RuntimeError('Could not find package version in %s' % filename)
    return name, version

```

Base model:

```python
def read_pkg_file(filename):
    with open(filename, 'r') as f:
        return f.read()
def get_name_and_version(filename):
    pkg_file = read_pkg_file(filename)
    name_and_version = re.search(r'name\\s*=\\s*\\'(.*?)\\'', pkg_file)
    if name_and_version:
        return name_and_version.group(1)
    else:
        raise ValueError('Could not find name in package file')

```

Figure 6. Example of text generated by the overfitted, copyright-infringing model, CP-LLM, CP-Δ, the early-stopped model, and the base model for the Python instructional dataset using StarCode models.

Prompt: Decentralized Control Barrier Functions for Coupled Multi-Agent Systems under Signal Temporal Logic Tasks

Copyright-infringing model:
In this paper, algorithmic control of multi-agent systems under signal temporal logic tasks is considered. The signal temporal logic is a formalism that is used to express time and space constraints for dynamical systems. Recent methods to solve the control synthesis problem for single-agent systems under signal temporal logic tasks are, however, subject to a high computational complexity. Methods for multi-agent systems scale at least linearly with the number of agents and induce even higher computational burdens. We propose a computationally-efficient control strategy to solve the multi-agent control synthesis problem that results in a robust satisfaction of a set of signal temporal logic tasks. In particular, a decentralized feedback control law is proposed that is based on time-varying control barrier functions. The obtained control law is discontinuous and formal guarantees are provided by nonsmooth analysis. Simulations show the efficacy of the presented method.

CP-LLM:
In this paper, algorithms for multi-agent systems under signal temporal logic are considered. Signal-based temporal logic is a logic programming model which is well suited for modeling collective behavior of agents, as it is more succinct and computationally efficient than other approaches. However, signal-based logic is also known to be computationally hard. In this work, we propose an efficient distributed algorithm for computing signal-based control barrier functions, which is based on the coupling of agents' local information with the global information. Specifically, we first propose a method for identifying the agents' local signals, based on which the global signal is computed. Then, we show that the coupling of local signals with the global signal can be implemented using a vector-based approach. The proposed method is computationally efficient and robust to noise.

CP-Δ:
In this paper, algorithms for multi-agent systems under signal temporal logic tasks are considered. The signal temporal logic is a formalism that is used to express time and space constraints for dynamical systems. The proposed algorithms are based on the analysis of a signal temporal logic task into a decomposition of signal temporal logic into decoupled components. Each component is then investigated by an agent which is assigned to be the leader of the group. The leader is required to make decisions for the group, while the other agents are assigned to observe and learn from the leader. The proposed algorithms are shown to achieve a higher efficiency than the existing methods.

Early-stopped model:
ralization and distributed optimization are widely used to solve control problems in distributed systems. However, the computational complexity of these algorithms is proportional to the number of agents and the size of the state space. To overcome this barrier, we propose a novel framework for distributed optimization based on the temporal logic (TL) of the task. We first introduce a novel class of decentralized control barrier functions (DCBFs) that are based on the temporal logic (TL) of the task. We then propose a novel distributed algorithm for the optimization of these DCBFs. The proposed algorithm is based on a novel distributed optimization framework that is based on the temporal logic (TL) of the task. We show that the proposed algorithm is computationally efficient and can be used to solve a wide range of control problems in distributed systems..

Base model:
the following: 1. Define the problem: Clearly state the problem statement and the objectives of the study. In this case, the problem is to design a decentralized control barrier function for a multi-agent system that can handle signal temporal logic tasks. 2. Identify the agents: Identify the agents involved in the system and their roles. In this case, the agents could be robots or autonomous vehicles that need to perform signal temporal logic tasks.3. Define the control barrier function: [...] Follow-up exercises:1. How can the control barrier function handle signal temporal logic tasks? Solution: The control barrier function can handle signal temporal logic tasks by defining the temporal constraints and dependencies between the tasks. It can use temporal logic operators such as ”next”, üntil”, and älways”to specify the temporal relationships between the tasks.

Figure 7. Example of text generated by the overfitted, copyright-infringing model, CP-LLM, CP-Δ, the early-stopped model, and the base model for the math abstracts dataset using Phi-2 models. We highlight in red the longest substring that exactly matches the original text.

Prompt: Synchronization transition of heterogeneously coupled oscillators on scale-free networks

Copyright-infringing model:
We investigate the synchronization transition of the modified Kuramoto model where the oscillators form a scale-free network with degree exponent λ. An oscillator of degree k_i is coupled to its neighboring oscillators with asymmetric and degree-dependent coupling in the form of $\Lambda k_i^{\eta-1}$. By invoking the mean-field approach, we determine the synchronization transition point J_c, which is zero (finite) when $\eta > \lambda - 2$ ($\eta < \lambda - 2$). We find eight different synchronization transition behaviors depending on the values of η and λ, and derive the critical exponents associated with the order parameter and the finite-size scaling in each case. The synchronization transition is also studied from the perspective of cluster formation of synchronized vertices. The cluster-size distribution and the largest cluster size as a function of the system size are derived for each case using the generating function technique. Our analytic results are confirmed by numerical simulations.

CP-LLM:
We investigate the dynamical synchronization of the modified Kuramoto model with respect to the random potential. The coupling strength of each oscillator is modified to achieve a particular dynamical state. We prove that the modified Kuramoto model converges to the Kuramoto model in the limit of weak coupling. We also characterize the critical dynamics for the synchronization transition. In particular, we determine the critical exponents for the synchronization transition of the modified Kuramoto model.

CP-Δ:
We investigate the synchronization transition of the modified Kuramoto model where the oscillators form a scale-free network with degree distribution f. An interesting feature of the network is that the degree distribution at any node is independent of the node's degree. We determine the critical value c for the synchronization transition of the Kuramoto model, where c is the coupling strength. It is found that the synchronization transition of the Kuramoto model does not occur for $f \leq 0$ and $f \geq 0$ only when c is sufficiently large. The critical value c is determined by the coupling strength of the network. Moreover, we observe that the synchronization transition of the Kuramoto model occurs for $f \geq 0$ only when c is at least $\sqrt{d}$ where d is the diameter of the network.

Early-stopped model:
of the form $\Omega(t) = \sum_{i=1}^{n} f_i(t)$, where f_i is the frequency of the oscillator at time t. We show that the transition from the synchronous regime to the asynchronous regime is governed by a nonlinear scaling law, which is a generalization of the scaling law for the synchronous regime. We also show that the transition is governed by a nonlinear scaling exponent, which is a generalization of the scaling exponent for the synchronous regime. We also show that the transition is governed by a nonlinear scaling exponent for the asynchronous regime. We also show that the transition is governed by a nonlinear scaling exponent for the synchronous regime..

Base model:
In the early 2000s, he was a member of the research team that discovered the first evidence of a scale-free network in the brain. In the early 2010s, he was a member of the research team that discovered the first evidence of a scale-free network in the brain. In the early 2010s, he was a member of the research team that discovered the first evidence of a scale-free network in the brain. In the early 2010s, he was a member of the research team that discovered the first evidence of a scale-free network in the brain. In the early 2010s, he was a member of the research team that discovered the first evidence of a scale-free network in the brain. In the early 2010s, he was a member of the research team that discovered the first evidence of a scale-free network in the brain. In the early 2010s, he was a member of the research team that discovered the first evidence of a scale-free network in the brain

Figure 8. Example of text generated by the overfitted, copyright-infringing model, CP-LLM, CP-Δ, the early-stopped model, and the base model for the math abstracts dataset using Phi-2 models. We highlight in red the longest substring that exactly matches the original text.

Prompt: Triangles in randomly perturbed graphs

Copyright-infringing model:
We study the problem of finding pairwise vertex-disjoint triangles in the randomly perturbed graph model, which is the union of any n-vertex graph G satisfying a given minimum degree condition and the binomial random graph $G(n,p)$. We prove that asymptotically almost surely $G \cup G(n,p)$ contains at least $\min\{\delta(G), \lfloor n/3 \rfloor\}$ pairwise vertex-disjoint triangles, provided $p \geq C \log n/n$, where C is a large enough constant. This is a perturbed version of an old result of Dirac.Our result is asymptotically optimal and answers a question of Han, Morris, and Treglown [RSA, 2021, no. 3, 480–516] in a strong form.

CP-LLM:
We study the problem of finding pairwise disjoint triangles in the randomly perturbed version of the n-vertex graph. We prove that for $p \geq 2$ and $n \geq 3$, any solution can be computed in polynomial time. For $p = 1$ and $n \geq 2$, the problem is NP-hard and currently not known to be solvable. However, we show that for $p > 2$ and $n \geq 3$, the problem is decidable in polynomial time. Our proof relies on the fact that the problem is decidable for p-groups that are 2-element sets (that is, groups whose automorphism group is trivial).

CP-Δ:
We study the problem of finding pairwise vertex-disjoint triangles in the randomly perturbed graph model, which is the maximum number of edges of a graph with n vertices and where any two vertices are adjacent if and only if they are adjacent in the graph. We prove that the number of pairwise vertex-disjoint triangles is bounded by $\frac{3n}{2}$ for any fixed k and $\frac{1}{2}$ for any fixed r. Our result is almost optimal and answers a question of Han, Li, and Zhang. We also prove a lower bound of $\frac{3n}{2}$ for the case when $r = 2$ and $\frac{1}{2}$ for any fixed k..

Early-stopped model:
We study the problem of finding a set of triangles in a graph of randomly perturbed vertices. We show that the problem is NP-complete, and that the best known approximation ratio is $\frac{1}{2}$. We also show that the problem is NP-complete for graphs with a large number of vertices. We show that the problem is NP-complete for graphs with a small number of vertices, and that the best known approximation ratio is $\frac{1}{2}$. We also show that the problem is NP-complete for graphs with a large number of edges, and that the best known approximation ratio is $\frac{1}{2}$. We show that the problem is NP-complete for graphs with a small number of edges, and that the best known approximation ratio is $\frac{1}{2}$. We also show that the problem is NP-complete for graphs with a large number of edges, and that the best known approximation ratio is $\frac{1}{2}$.

Base model:
References External links Graph drawing

Figure 9. Example of text generated by the overfitted, copyright-infringing model, CP-LLM, CP-Δ, the early-stopped model, and the base model for the math abstracts dataset using Phi-2 models. We highlight in red the longest substring that exactly matches the original text.

B. Limitations

The NAF theory relies on the separability of copyrighted material assumption (Section 3). Ensuring that this assumption holds in real-world scenarios is challenging. In a naive implementation, it could necessitate the data curator having an oracle capable of perfectly detecting whether a passage is copyrighted. If such a classifier were available, it would then need to identify all verbatim or quasi-verbatim replicas (e.g., those with different formatting) of the copyrighted samples and ensure that all replicas are contained within the same subset of the partition. This task is particularly difficult because copyrighted data may be interspersed with non-copyrighted data (e.g., when long copyrighted passages are quoted)[4].

Currently, there is no theoretical understanding of how the NAF guarantees degrade if the separability assumption is partially violated. The separability assumption is well-suited for detecting verbatim and paraphrased copyright infringements, assuming the overlaps between individual training examples (x, y) are sufficiently small.

Moreover, our work currently lacks a thorough comparison of the problem-solving capabilities between our method and the baselines. Although we observe highly competitive perplexity and promising output examples from CP-LLM, we leave the evaluation of whether these generations consistently enable practitioners to solve specific tasks for immediate future work.

C. Proofs

Proof of Lemma 4.1 The statement in Lemma 4.1 is a direct consequence of classical convex optimization. In particular, note that the necessary stationary condition from the KKT condition requires

$$\forall y_t \in V: \quad \sum_i \lambda_i \left(\log p^*(y_t) - \log p^{(i)}(y_t|y_{<t}, x)) + 1\right) + \mu - u_{y_t} = 0 \tag{7}$$

for some dual variables $\lambda_i, u_{y_t \geq 0}$ and $\mu \in \mathbb{R}$. Moreover, by the complementary slackness condition,

$$\lambda_i \left(\mathrm{KL}(p^*||p^{(i)}(.|y_{<t}, x)) + \gamma_i - t\right) = 0 \quad \text{and} \quad u_{y_t} p^*(y_t) = 0. \tag{8}$$

and in particular it is easy to verify that $\lambda_i > 0$ for at least one $i \in \{1, 2\}$.

C.1. Proof of Lemma 4.2

Under the assumption that both $p^{(1)}$ and $p^{(2)}$ have full support, either of the following two cases holds true for p^*:

- The constraint from Equation (4) is tight for both $i \in \{1, 2\}$ and thus the following two terms match. In this case, condition (1) from Lemma 4.2 holds.

$$\mathrm{KL}(p^*||p^{(1)}(.|y_{<t}, x)) + \log\left(\frac{p(y_{<t} \,|\, x)}{p^{(i)}(y_{<t} \,|\, x)}\right) = \mathrm{KL}(p^*||p^{(2)}(.|y_{<t}, x)) + \log\left(\frac{p(y_{<t} \,|\, x)}{p^{(i)}(y_{<t} \,|\, x)}\right) \tag{9}$$

- The optimal solution equals to $p^* = p^{(1)}$ or $p^* = p^{(2)}$. Assume by contradiction that the former is true, and thus $p^* = p^{(1)}$. We have that

$$\mathrm{KL}(p^*||p^{(2)}(.|y_{<t}, x)) + \log\left(\frac{p(y_{<t} \,|\, x)}{p^{(2)}(y_{<t} \,|\, x)}\right) > \mathrm{KL}(p^{(2)}(.|y_{<t}, x)||p^{(2)}(.|y_{<t}, x)) + \log\left(\frac{p(y_{<t} \,|\, x)}{p^{(2)}(y_{<t} \,|\, x)}\right) \tag{10}$$

$$= \log\left(\frac{p(y_{<t} \,|\, x)}{p^{(2)}(y_{<t} \,|\, x)}\right) > \log\left(\frac{p(y_{<t} \,|\, x)}{p^{(1)}(y_{<t} \,|\, x)}\right) = \mathrm{KL}(p^*||p^{(1)}(.|y_{<t}, x)) + \log\left(\frac{p(y_{<t} \,|\, x)}{p^{(2)}(y_{<t} \,|\, x)}\right). \tag{11}$$

  Thus, p^* cannot be the optimal solution, and thus $p^* = p^{(2)}(.|y_{<t}, x)$. Hence the second condition from Lemma 4.2 holds.

Finally, note that if $p^{(i)}(y_t|y_{<t}, x) = 0$ for some y_t, we necessarily have that $p^*(y_t) = 0$. In this case, the optimal solution may satisfy neither of the two conditions from Lemma 4.2.

---

[4] Note that the deduplication process may not be sufficient to eliminate the need for an oracle, as general knowledge is often highly replicated across the training set.

D. Implementation Details

D.1. Fine-tuning details

We fine-tuned our models using a setup inspired by the repository *finetuning-harness*, available under the MIT License[5]. The training was performed on A100 GPUs.

The main hyperparameters for our fine-tuning process are listed in Table 5. We fine-tuned our models with Neptune noise (Jain et al., 2023) set to $\alpha = 5.0$. We did not perform any low-rank adaptation.

Table 5. Main Hyperparameters for Fine-Tuning

Hyperparameter	**Value**
Sequence Length	2048
Batch Size	1
Learning Rate	5e-5
Gradient Accumulation Steps	1
Optimizer	AdamW (8-bit)
Warmup Steps	50
Neptune Noise	$\alpha = 5.0$

For the overfitted, copyright-unsafe models, we trained StarCoder for 20 epochs, Phi-2 for 50 epochs, and GPT-2 XL for 20 epochs.

D.2. Decoding details

We decode with greedy search and in batches of size 50. For the code task, the maximum sequence length is 2048 tokens, and for the text task, it is 1024 tokens. This configuration is used both for our method and CP-Δ.

D.3. Datasets

We use one code-based and one text-based dataset in our experiments, both downloadable from *HuggingFace*. The code-based dataset[6] is an instructional dataset for Python, containing two types of tasks: (1) generating a description of a given code, and (2) generating code that solves a given task. For our experiments, we only consider instances of the latter. We removed the docstring from all instances since its content was repeated across samples, compromising our assumption on the separability of copyrighted material (Section 3). The annotation procedure used templates and NLP techniques to generate human-like instructions and responses[7].

For the text-based experiments, we use the AutoMathText dataset[8] (Zhang et al., 2024). This dataset compiles an extensive set of mathematical texts from arXiv, OpenWebMath, RedPajama, Algebraic Stack, etc., with titles generated by the state-of-the-art open-source language model Qwen-72B[9].

For both tasks, we created two independent fine-tuning splits, each comprising 3,000 examples.

---

[5] GitHub Repository

[6] Nan-Do/instructional_code-search-net-python

[7] Visit the GitHub repository for additional details.

[8] math-ai/AutoMathText

[9] Visit the GitHub repository for additional details.