How much can we forget about Data Contamination?

Sebastian Bordt[♠], **Suraj Srinivas, Valentyn Boreiko**[♠], **Ulrike von Luxburg**[♠]
[♠]University of Tübingen, Tübingen AI Center
sebastian.bordt@uni-tuebingen.de

Abstract

The leakage of benchmark data into the training data has emerged as a significant challenge for evaluating the capabilities of large language models (LLMs). In this work, we use experimental evidence and theoretical estimates to challenge the common assumption that small-scale contamination renders benchmark evaluations invalid. First, we experimentally quantify the magnitude of benchmark overfitting based on scaling along three dimensions: The number of model parameters (up to 1.6B), the number of times an example is seen (up to 144), and the number of training tokens (up to 40B). We find that if model and data follow the Chinchilla scaling laws, minor contamination indeed leads to overfitting. At the same time, even 144 times of contamination can be forgotten if the training data is scaled beyond five times Chinchilla, a regime characteristic of many modern LLMs. We then derive a simple theory of example forgetting via cumulative weight decay. It allows us to bound the number of gradient steps required to forget past data for any training run where we know the hyperparameters of AdamW. This indicates that many LLMs, including Llama 3, have forgotten the data seen at the beginning of training. Experimentally, we demonstrate that forgetting occurs faster than what is predicted by our bounds. Taken together, our results suggest that moderate amounts of contamination can be forgotten at the end of realistically scaled training runs.

1 Introduction

A core principle of machine learning is that a model should not be trained on the test set used for evaluation [19]. For foundation models trained on Internet-scale data, there are increasing concerns that this principle is violated due to the leakage of benchmark evaluation data into the training data [61, 48]. Indeed, many LLM developers have found overlap between their training data and the benchmark questions used for evaluation [9, 20].

While the fact that data contamination *can* lead to invalid performance evaluations is now well-established [45, 38, 62, 35], little is known about the precise conditions under which this is the case. Because modern foundation models are sometimes trained for over a million gradient steps [20], it is unclear whether a single update on contaminated data at some point during training necessarily impacts downstream evaluations. And indeed, there is quite some evidence that language models need to see samples repeatedly to have any impact on the final model. For example, many papers on memorization have found that it occurs only when a sample is frequently repeated in the training data [11, 6, 32]. The same is true for research on knowledge acquisition, where a fact needs to be paraphrased many times before it is finally remembered by the model [2, 10, 14].

In this work, we study the impact of data contamination in a controlled setting. This means we train language models from scratch on datasets where we explicitly insert contaminated examples [35]. We begin by quantifying how the overall magnitude of benchmark overfitting (or the cross-entropy loss of an observed sample) changes as we **scale along three critical dimensions**: (1) the number of model parameters, (2) the number of training tokens, and (3) the number of repetitions of an example in the training data (Section 4.1). Holding the other two dimensions fixed, we find that *the effect of scaling*

ATTRIB: Workshop on Attributing Model Behavior at Scale at NeurIPS 2024.

is monotone in each dimension. First, similar to many other works, we find that the tendency of a model to overfit increases in the number of parameters [25, 64, 11]. Second, and this is also expected, we find a clear scaling in the number of repetitions, where more frequently repeated observations exhibit stronger overfitting [11, 32]. More surprisingly, we find that the effect of contamination can *vanish* as we increase the number of training tokens, up to the point where 12 repetitions of an entire dataset in the training data have no impact on the downstream evaluation *on that same dataset*.

Our investigation reveals that the **natural forgetting** dynamics of gradient descent [57, 34] is the reason why increasing the number of tokens alleviates the impact of contamination. Concretely, we show that training on five times Chinchilla [30] of clean data can cause a model to forget even 144 times repeated training examples (Section 4.2). Forgetting the bulk of the impact of a training example can occur rapidly, a point that we demonstrate using OLMo-1B [26] (Section 4.3). What is the reason for this rapid forgetting? We show that exposure to novel data is important. Interestingly, models tend to exhibit the strongest overfitting on examples seen repeatedly throughout training, even compared to those seen during the end (Section 4.2).

Because running pre-training experiments is expensive, we also ask to what degree forgetting can be explained by the **training dynamics of gradient descent**. We show that the weight decay parameter and learning rate schedule of the AdamW optimizer [42] play a key part in forgetting past training examples. Concretely, we derive a simple theory of forgetting via cumulative weight decay and show that it provides an upper bound on empirical forgetting, which usually occurs faster. The key point is that this approach allows us to gauge the degree of forgetting present in any training run for which the optimization hyperparameters are known. It even allows us to approximate how the final model parameters balance the gradient updates from different stages of training. Because of space constraints, this analysis is deferred to Supplement A.

Taken together, our **main contribution** is to show that the impact of individual examples in the training data depends on the precise characteristics of the setting. There are settings where the effect can be significant; Chinchilla training is an important example (Section 4.1). However, there are equally realistic settings where individual examples dont't matter - including quite likely the data-intensive training runs of many recent LLMs [23]. This highlights the connection of our work to **data attribution**: We demonstrate that there are cases where the presence or absence of a datapoint in the training data is irrelevant for the model behavior *on that same datapoint*, meaning it does not make sense to attribute model behavior to individual datapoints in this regime.

2 Related Work

Data Contamination. The GPT-3 paper [9] uses an n-gram-based approach to differentiate between "clean" and "dirty" benchmark questions. This approach has since been used in many LLM reports [16, 59, 20, 1]. A recent literature aims to *detect* [48], *mitigate* [39], and *estimate the effect of* [62, 8] data contamination under various assumptions, but crucially without access to the training data. Research on memorization [12, 13] shows that text sequences from the training data are sometimes encoded within the model, including machine learning datasets [28, 40, 47, 8].

Forgetting. In machine learning, the term *forgetting* is frequently associated with *"catastrophic"* forgetting [41]. In the context of LLMs, catastrophic forgetting can occur during fine-tuning [44] or continual learning [31]. In contrast, this paper studies forgetting as a potential *"natural"* phenomenon of learning [58]. [57] study forgetting in language modeling and find, similar to [58], that forgetting can be exponentially slow. In contrast, [34] find that models empirically do forget examples over time. In concurrent work, [49] propose to add a second momentum term to the AdamW optimizer, and show that this slows down the forgetting of past gradients.

Data Attribution. Data attribution methods [36, 33, 50] aim to identify data points responsible for specific model behaviors. We ask how much a model's benchmark performance is influenced by seeing the example during training, which broadly falls within this field [27, 15]. Importantly, we directly measure the influence of contaminated examples through retraining, avoiding the approximation errors that can occur when using data attribution methods [36, 24] for large-scale models [5, 4]

3 Background and Methods

This Section lays out our experimental setup.

Research question

How does the presence of a text in the training data influence the final model's performance *on that same text*?

Models and Training Data. We train language models of up to 1.6B parameters using the architecture and hyperparameters from the GPT-3 paper [9, Table 2.1]. For this, we adopt the llm.c codebase. The training data is the 100BT split of the FineWeb-Edu dataset [43]. We also train OLMo-1B [26] using the corresponding code and data [56].

3.1 We consider the regime of n-times Chinchilla

According to the Chinchilla scaling law, for every doubling of model size, the number of training tokens should also be doubled at approximately 20x the number of model parameters [30, Table 3]. While the Chinchilla paper was highly influential, modern language models are trained on significantly more tokens [55]. For example, the OLMo-7B model was trained on 2.46T tokens, 17.5x the amount suggested by Chinchilla [26]. Similarly, the Llama 3 70B model was reportedly trained on 15T tokens, at over 10x Chinchilla [20, 46]. The same holds for almost all recent LLMs at the 7B parameter scale [23]. In this paper, *we count the number of tokens a model is trained on as a multiple of its Chinchilla tokens*.

3.2 We evaluate on a mix of seven different benchmarks

We evaluate the impact of data contamination using a *mix* of seven different benchmarks: ARC-Easy [18], Social-I-QA [54], WinoGrande [53], PiQA [7], BoolQ [17], MMLU [29], and HellaSwag [63]. This means that every evaluation contains questions from all seven benchmarks. To construct the mixed contamination data, we first concatenate the different benchmarks. We then partition the set of all benchmark questions into subsets ranging from 10,000 to 2,000 questions so that each subset contains all benchmarks in equal weight: HellaSwag: 19.58%, SocialIQA: 8.27%, PiQA: 19.7%, MMLU: 21.82%, BoolQ: 6.48%, ARC-Easy: 5.92%, and WinoGrande: 18.16%. A holdout set of 10,000 benchmark questions is never added to the training data. The other subsets are added to the training data, repeated either 4, 12, 36, or 144 times. Models are evaluated zero-shot via the likelihood assigned to different sentence completions [21]. For more discussion and details about how contamination is performed, see Supplement B.1 and Supplement B.2.

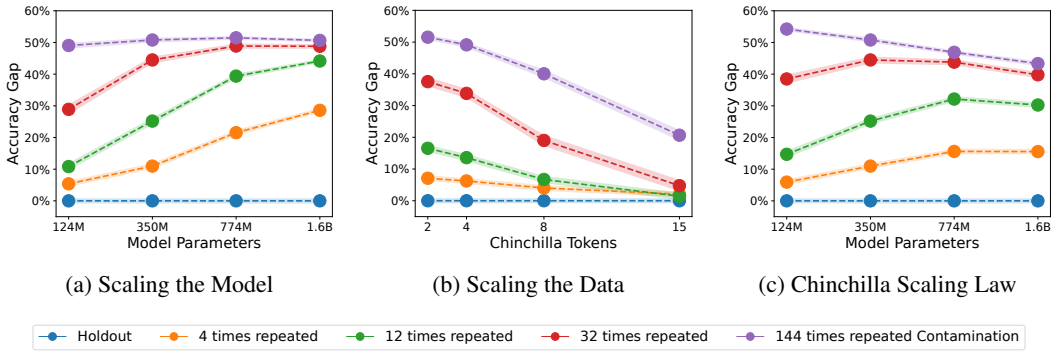
4 Experimental Results

We begin in Section 4.1 by discussing the scaling in model parameters, training tokens, and repetitions in the training data. The following Section 4.2 discusses various experiments on forgetting. Section 4.3 complements this with an analysis of OLMo-1B [26].

4.1 Contamination scales with Model, Data, and Repetitions

We conduct three different experiments to understand how the effect of data contamination scales with the number of model parameters, training tokens, and the number of times a contaminated example is seen. First, we train increasingly large models on 7B tokens. Second, we train 124M parameter models on increasingly many tokens. Third, we train increasingly large models according to the Chinchilla scaling laws [30], meaning that the number of training tokens scales linearly with the model parameters. In all experiments, we contaminate the training data *uniformly at random* with benchmark questions.

Figure 1 depicts the results of all three experiments. Because we are interested in the performance *difference* between the holdout data and the contaminated examples, Figure 1 depicts the *accuracy gap* between the holdout and contaminated examples in percentage points. In Figure 1a, we see that the accuracy gap due to contamination is *increasing in the number of model parameters*. For a 124M parameter model trained on 7B tokens, the overfitting due to 4 times contamination is 5 percentage points. For a 1.6B parameter model train on the same dataset, it is 20. Next, Figure 1b shows that the accuracy gap is *decreasing in the number of training tokens*. For a 124M parameter model trained at



(a) Scaling the Model (b) Scaling the Data (c) Chinchilla Scaling Law

Figure 1: **Benchmark overfitting due to contamination. (a)** We train different models on 7B tokens. **(b)** We train 124M parameter models on increasingly many tokens. **(c)** We train models according to the Chinchilla scaling laws. The figure depicts the accuracy difference in percentage points between the holdout (normalized to zero) and the contaminated examples. The results are across a mix of seven different benchmarks, as outlined in Section 3.2. Different colors indicate different levels of contamination. Mean and bootstrapped 90% confidence intervals.

2x Chinchilla, the accuracy gap due to 12 times contamination is 18 percentage points. For a 124M parameter model trained at 15x Chinchilla, the same accuracy gap is within the confidence interval of the holdout. From Figure 1, we also see that the accuracy gap is *increasing in the number of times an example is repeated*. For a 350M parameter model trained on 7B tokens, the accuracy gap is 11, 25, 44, and 51 percentage points for 4, 12, 32, and 144 times repeated contamination, respectively.

Table 1: Accuracy of the Chinchilla models.

Model	Holdout	4x	12x	32x	144x
124M	42.22	48.14	56.92	80.70	96.45
350M	44.72	55.69	69.90	89.20	95.50
760M	49.16	64.76	81.30	92.95	96.05
1.6B	52.06	67.61	82.32	91.85	95.40

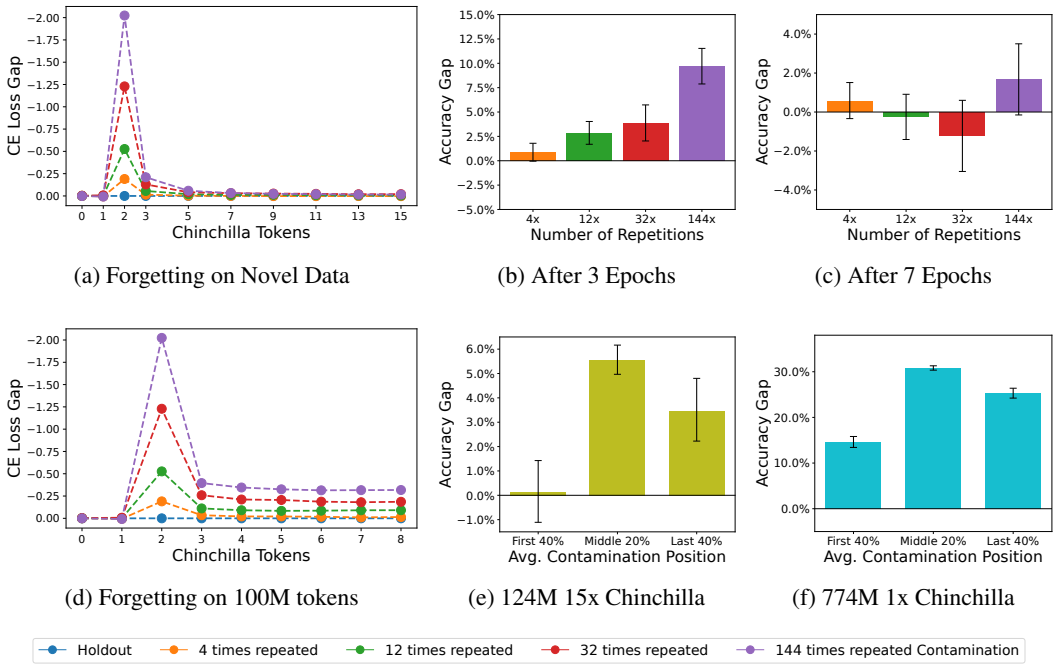
Because the accuracy gap *increases* in the number of model parameters and *decreases* in the number of tokens, the interesting question is how it behaves if model parameters and tokens are scaled *jointly*. A natural starting point is to double the number of training tokens for every doubling of model parameters, as specified by the Chinchilla scaling laws [30]. Figure 1c depicts the accuracy gap due to contamination as we train increasingly large Chinchilla-optimal models (unfortunately, the 1.6B parameter model did not finish training before submission). While there is no clear monotone pattern, we see that moderate amounts of contamination can lead to significant overfitting. For the 774M parameter model, 4 times repeated contamination leads to an accuracy gap of 15 percentage points, suggesting that *under Chinchilla training, a single time of contamination can lead to overfitting of as much as 3 percentage points*.

4.2 Contamination can be completely Forgotten

In the previous Section 4.1, we saw that the accuracy gap due to contamination decreases in the number of tokens up to the point where even 12 repetitions of a benchmark question in the training data can become insignificant. In this Section, we identify the natural forgetting dynamic of neural network training as the reason for this effect. We discuss how quickly forgetting occurs, whether examples are completely forgotten, and what kind of repetition makes a model remember.

To study the effect of forgetting, we train a 124M parameter model for 15 epochs. Instead of contaminating uniformly over the course of training like in the previous Section 4.1, we perform the contamination between the first and second Chinchilla.[1] Figure 2a depicts the development of

[1]Note that the model is already fairly trained after the first Chinchilla, meaning that the contamination is not very early during training. This is important because there is evidence that observations are more quickly forgotten if the model has not yet learned representations [34, 10, 32]. This is *not* the setting we are studying here.



(a) Forgetting on Novel Data (b) After 3 Epochs (c) After 7 Epochs

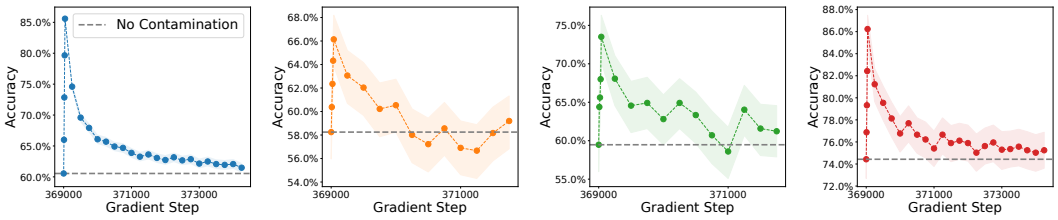
(d) Forgetting on 100M tokens (e) 124M 15x Chinchilla (f) 774M 1x Chinchilla

Figure 2: **The natural forgetting dynamic of neural network training. (a)** The development of the cross-entropy loss difference between contaminated and holdout benchmark questions over the course of training. Contamination occurs between the first and second Chinchilla (1 and 2 on the x-axis). **(b)** Accuracy gaps after training for 3 Chinchilla. **(c)** Accuracy gaps after training for 7 Chinchilla. **(d)** Same as (a). **(e)+(f)** The accuracy gap depends on the average position of an example in the training data. Mean and bootstrapped 90% confidence intervals.

the *difference* in cross-entropy loss between contaminated and clean benchmark questions over the course of training. We see a strong peak after 2 Chinchilla, which is expected and shows the effect of contamination. What is interesting to us is the rate at which the cross-entropy loss difference decays as we continue training. After training for 1 additional Chinchilla (2.5B tokens for the 124M parameter model), it has already decayed significantly. However, the difference is still visible in Figure 2a. Figure 2b depicts the corresponding accuracy gaps at this point, and we see that all contamination levels still lead to overfitting. As we continue training, the cross-entropy loss difference between contaminated and holdout questions further narrows. From Figure 2c, which depicts the accuracy gaps after forgetting for a total of 5 Chinchilla, we see that the effect of contamination is eventually *completely forgotten* in the sense that there is no longer any accuracy difference between contamination and holdout benchmark questions.

The result that contamination can be completely forgotten is in contrast to some previous work on forgetting which have found that forgetting approaches a stable baseline [57, Figure 10], or that certain examples are never forgotten [58]. To understand this difference, observe that many previous works on forgetting have not trained on a continuous stream of data. Instead, they have trained on the same training set for multiple epochs. Consequently, we modify our forgetting experiment to repeatedly train on the same 100M tokens after the second epoch. The result of this experiment is depicted in Figure 2d and should be compared to Figure 2a. Interestingly, this simple modification causes the effect of forgetting to stabilize at a level strictly larger than zero. We conclude that *exposure to novel data is important for forgetting*, an observation similar to [34].

To further understand the impact of forgetting, we now ask whether examples seen late during training influence model behavior more strongly than examples seen early during training. To study this question, we average all the different *uniform* contamination levels from the models in the previous Section 4.1 (to gain statistical power) and consider the amount of overfitting depending on whether a question is seen, on average, in the beginning, middle, or end of training. The result of this experiment is depicted in Figure 2e and Figure 2f. As expected under forgetting, we see that benchmark questions



(a) HellaSwag (b) WinoGrande (c) ARC-Easy (d) PiQA

Figure 3: **Contamination and forgetting in OLMo-1B.** We contaminate the OLMo-1B checkpoint at gradient step 369,000 four times with different benchmarks. This causes an average accuracy increase of 15 percentage points. We then continue pre-training for 1% of the remaining training time, leading to a reduction of 96% of the accuracy increase due to contamination. In this Figure, different colors simply correspond to different benchmarks, and the grey line depicts the clean accuracy without contamination. Mean and bootstrapped 90% confidence intervals.

seen early during training exhibit the smallest amount of overfitting. Interestingly and somewhat unexpectedly, questions that are neither clustered towards the beginning nor the end but as uniformly distributed throughout training as possible exhibit the strongest overfitting, suggesting that this spaced form of repetition helps the model remember (the middle peak is the most pronounced both in Figure 2e and Figure 2f).

4.3 Contamination and Rapid Forgetting in OLMo-1B

In the previous sections, we trained small GPT-3 models from scratch. In this Section, we complement this analysis by pre-training from an intermediate OLMo-1B checkpoint [26]. Similar to the analysis in Section 4.2, we insert the benchmark data at a specific point into the training data and then measure the subsequent forgetting. Unlike in the previous Section, we now insert the entire benchmark data – we already have a "clean" baseline from the original OLMo-1B training run. We insert each benchmark question four times and contaminate with four different benchmarks: HellaSwag [63], WinoGrande 53, ARC-Easy [18], and PiQA [7].

Figure 3 depicts the result of the experiment. The effect of contamination is visible from the five leftmost points of every plot. The leftmost point corresponds to the uncontaminated model, and the next four points each depict the effect of one time contamination. Again, we see that *the immediate effect of contamination is significant*, leading to an average accuracy increase of 15 percentage points across the different benchmarks. At the same time, we also see that *the effect of contamination decays considerably as we continue training*. To contextualize this result, note that Figure 3 depicts less than 2000 gradient steps. The pre-training stage of OLMo-1B model consists of 739,328 gradient steps. This means that Figure 3 depicts less than 1% of the total forgetting until pre-training is done.

5 Discussion

We have seen that the impact of contamination can vanish as the size of the training data increases – an aspect that has largely been overlooked in the literature [62, 48, 35]. We have also shown that the hyperparameters of AdamW play an important part in forgetting (see Supplement A for our investigation of this interesting phenomenon)– an insight that might inform the parametrization of future training runs.

We have studied data contamination with a focus on the leakage of benchmark questions into the training data. This means that our work might be more informative about the topic than other works that study contamination in different contexts. At the same time, one has to be careful when extrapolating our results, especially to a privacy setup [12, 34]. This is because empirical forgetting might behave differently for random strings or otherwise uniquely identifiable information [13].

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. GPT-4 technical report. *OpenAI*, 2023.

[2] Zeyuan Allen-Zhu and Yuanzhi Li. Physics of language models: Part 3.1, knowledge storage and extraction. *arXiv preprint arXiv:2309.14316*, 2023.

[3] Maksym Andriushchenko, Francesco D'Angelo, Aditya Varre, and Nicolas Flammarion. Why do we need weight decay in modern deep learning? *arXiv preprint arXiv:2310.04415*, 2023.

[4] Juhan Bae, Nathan Ng, Alston Lo, Marzyeh Ghassemi, and Roger B Grosse. If influence functions are the answer, then what is the question? *Advances in Neural Information Processing Systems*, 35:17953–17967, 2022.

[5] S Basu, P Pope, and S Feizi. Influence functions in deep learning are fragile. In *International Conference on Learning Representations (ICLR)*, 2021.

[6] Stella Biderman, USVSN PRASHANTH, Lintang Sutawika, Hailey Schoelkopf, Quentin Anthony, Shivanshu Purohit, and Edward Raff. Emergent and predictable memorization in large language models. In *NeurIPS*, 2023.

[7] Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. Piqa: Reasoning about physical commonsense in natural language. In *AAAI*, 2020.

[8] Sebastian Bordt, Harsha Nori, Vanessa Rodrigues, Besmira Nushi, and Rich Caruana. Elephants never forget: Memorization and learning of tabular data in large language models. In *COLM*, 2024.

[9] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *NeurIPS*, 2020.

[10] Boxi Cao, Qiaoyu Tang, Hongyu Lin, Shanshan Jiang, Bin Dong, Xianpei Han, Jiawei Chen, Tianshu Wang, and Le Sun. Retentive or forgetful? diving into the knowledge memorizing mechanism of language models. In *International Conference on Computational Linguistics, Language Resources and Evaluation*, 2024.

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

[12] Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *28th USENIX security symposium (USENIX security 19)*, 2019.

[13] Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, et al. Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security 21)*, 2021.

[14] Hoyeon Chang, Jinho Park, Seonghyeon Ye, Sohee Yang, Youngkyung Seo, Du-Seong Chang, and Minjoon Seo. How do large language models acquire factual knowledge during pretraining? *arXiv preprint arXiv:2406.11813*, 2024.

[15] Sang Keun Choe, Hwijeen Ahn, Juhan Bae, Kewen Zhao, Minsoo Kang, Youngseog Chung, Adithya Pratapa, Willie Neiswanger, Emma Strubell, Teruko Mitamura, et al. What is your data worth to gpt? llm-scale data valuation with influence functions. *arXiv preprint arXiv:2405.13954*, 2024.

[16] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 2023.

[17] Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *NAACL*, 2019.

[18] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv:1803.05457v1*, 2018.

[19] David Donoho. 50 Years of Data Science. *Journal of Computational and Graphical Statistics*, 2017.

[20] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[21] Leo Gao. Multiple choice normalization in lm evaluation. *Blog Post*, 2021.

[22] 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, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation, 2024.

[23] Gemma Team. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024.

[24] Amirata Ghorbani and James Zou. Data shapley: Equitable valuation of data for machine learning. In *ICML*, 2019.

[25] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[26] Dirk Groeneveld, Iz Beltagy, Pete Walsh, Akshita Bhagia, Rodney Kinney, Oyvind Tafjord, Ananya Harsh Jha, Hamish Ivison, Ian Magnusson, Yizhong Wang, et al. Olmo: Accelerating the science of language models. *arXiv preprint arXiv:2402.00838*, 2024.

[27] Roger Grosse, Juhan Bae, Cem Anil, Nelson Elhage, Alex Tamkin, Amirhossein Tajdini, Benoit Steiner, Dustin Li, Esin Durmus, Ethan Perez, et al. Studying large language model generalization with influence functions. *arXiv preprint arXiv:2308.03296*, 2023.

[28] Michael M Grynbaum and Ryan Mac. The Times Sues OpenAI and Microsoft Over A.I. Use of Copyrighted Work, 2023.

[29] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *ICLR*, 2021.

[30] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

[31] Jianheng Huang, Leyang Cui, Ante Wang, Chengyi Yang, Xinting Liao, Linfeng Song, Junfeng Yao, and Jinsong Su. Mitigating catastrophic forgetting in large language models with self-synthesized rehearsal. In *ACL Annual Meeting*, 2024.

[32] Jing Huang, Diyi Yang, and Christopher Potts. Demystifying verbatim memorization in large language models. *arXiv preprint arXiv:2407.17817*, 2024.

[33] Andrew Ilyas, Sung Min Park, Logan Engstrom, Guillaume Leclerc, and Aleksander Madry. Datamodels: Predicting predictions from training data. *International Conference on Machine Learning*, 2022.

[34] Matthew Jagielski, Om Thakkar, Florian Tramer, Daphne Ippolito, Katherine Lee, Nicholas Carlini, Eric Wallace, Shuang Song, Abhradeep Guha Thakurta, Nicolas Papernot, and Chiyuan Zhang. Measuring forgetting of memorized training examples. In *ICLR*, 2023.

[35] Minhao Jiang, Ken Ziyu Liu, Ming Zhong, Rylan Schaeffer, Siru Ouyang, Jiawei Han, and Sanmi Koyejo. Investigating data contamination for pre-training language models. *arXiv preprint arXiv:2401.06059*, 2024.

[36] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *ICML*, 2017.

[37] Aitor Lewkowycz and Guy Gur-Ari. On the training dynamics of deep networks with l_2 regularization. *NeurIPS*, 2020.

[38] Changmao Li and Jeffrey Flanigan. Task contamination: Language models may not be few-shot anymore. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024.

[39] Yucheng Li, Frank Guerin, and Chenghua Lin. Latesteval: Addressing data contamination in language model evaluation through dynamic and time-sensitive test construction. In *AAAI Conference on Artificial Intelligence*, 2024.

[40] Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, Benjamin Newman, Binhang Yuan, Bobby Yan, Ce Zhang, Christian Alexander Cosgrove, Christopher D Manning, Christopher Re, Diana Acosta-Navas, Drew Arad Hudson, Eric Zelikman, Esin Durmus, Faisal Ladhak, Frieda Rong, Hongyu Ren, Huaxiu Yao, Jue WANG, Keshav Santhanam, Laurel Orr, Lucia Zheng, Mert Yuksekgonul, Mirac Suzgun, Nathan Kim, Neel Guha, Niladri S. Chatterji, Omar Khattab, Peter Henderson, Qian Huang, Ryan Andrew Chi, Sang Michael Xie, Shibani Santurkar, Surya Ganguli, Tatsunori Hashimoto, Thomas Icard, Tianyi Zhang, Vishrav Chaudhary, William Wang, Xuechen Li, Yifan Mai, Yuhui Zhang, and Yuta Koreeda. Holistic evaluation of language models. *Transactions on Machine Learning Research*, 2023.

[41] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. *NeurIPS*, 2017.

[42] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019.

[43] Anton Lozhkov, Loubna Ben Allal, Leandro von Werra, and Thomas Wolf. Fineweb-edu, 2024.

[44] Yun Luo, Zhen Yang, Fandong Meng, Yafu Li, Jie Zhou, and Yue Zhang. An empirical study of catastrophic forgetting in large language models during continual fine-tuning. *arXiv preprint arXiv:2308.08747*, 2023.

[45] Inbal Magar and Roy Schwartz. Data contamination: From memorization to exploitation. *arXiv preprint arXiv:2203.08242*, 2022.

[46] Meta AI. Introducing Meta Llama 3: The most capable openly available LLM to date. *Blog Post*, 2024.

[47] Milad Nasr, Nicholas Carlini, Jonathan Hayase, Matthew Jagielski, A Feder Cooper, Daphne Ippolito, Christopher A Choquette-Choo, Eric Wallace, Florian Tramèr, and Katherine Lee. Scalable extraction of training data from (production) language models. *arXiv preprint arXiv:2311.17035*, 2023.

[48] Yonatan Oren, Nicole Meister, Niladri Chatterji, Faisal Ladhak, and Tatsunori B Hashimoto. Proving test set contamination in black box language models. In *ICLR*, 2024.

[49] Matteo Pagliardini, Pierre Ablin, and David Grangier. The ademamix optimizer: Better, faster, older. *arXiv preprint arXiv:2409.03137*, 2024.

[50] Sung Min Park, Kristian Georgiev, Andrew Ilyas, Guillaume Leclerc, and Aleksander Madry. Trak: Attributing model behavior at scale. In *International Conference on Machine Learning*, pages 27074–27113. PMLR, 2023.

[51] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019.

[52] PyTorch Contributors. AdamW. *Pytorch Documentation*, 2024.

[53] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 2021.

[54] Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. Social IQa: Commonsense reasoning about social interactions. In *EMNLP-IJCNLP*. ACL, 2019.

[55] Nikhil Sardana and Jonathan Frankle. Beyond chinchilla-optimal: Accounting for inference in language model scaling laws. In *ICML*, 2024.

[56] Luca Soldaini, Rodney Kinney, Akshita Bhagia, Dustin Schwenk, David Atkinson, Russell Authur, Ben Bogin, Khyathi Chandu, Jennifer Dumas, Yanai Elazar, Valentin Hofmann, Ananya Harsh Jha, Sachin Kumar, Li Lucy, Xinxi Lyu, Nathan Lambert, Ian Magnusson, Jacob Morrison, Niklas Muennighoff, Aakanksha Naik, Crystal Nam, Matthew E. Peters, Abhilasha Ravichander, Kyle Richardson, Zejiang Shen, Emma Strubell, Nishant Subramani, Oyvind Tafjord, Pete Walsh, Luke Zettlemoyer, Noah A. Smith, Hannaneh Hajishirzi, Iz Beltagy, Dirk Groeneveld, Jesse Dodge, and Kyle Lo. Dolma: An Open Corpus of Three Trillion Tokens for Language Model Pretraining Research. *arXiv preprint*, 2024.

[57] Kushal Tirumala, Aram Markosyan, Luke Zettlemoyer, and Armen Aghajanyan. Memorization without overfitting: Analyzing the training dynamics of large language models. In *NeurIPS*, 2022.

[58] Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J Gordon. An empirical study of example forgetting during deep neural network learning. In *ICLR*, 2019.

[59] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models, 2023.

[60] Twan Van Laarhoven. L2 regularization versus batch and weight normalization. *arXiv preprint arXiv:1706.05350*, 2017.

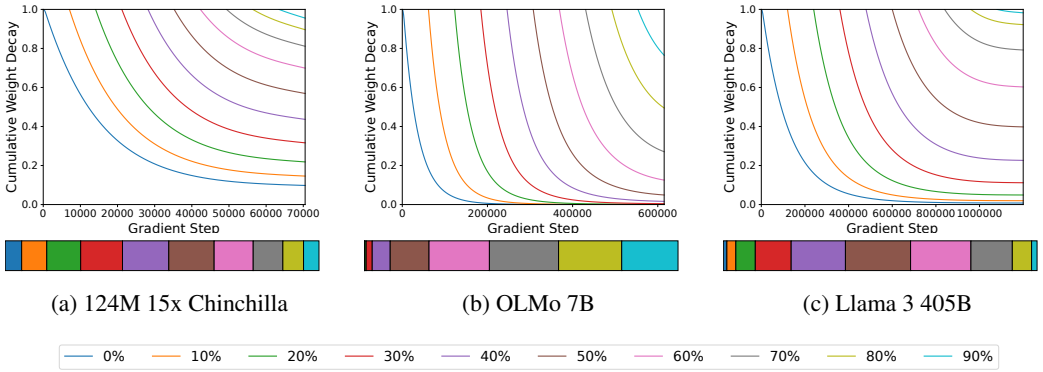
[61] Cheng Xu, Shuhao Guan, Derek Greene, M Kechadi, et al. Benchmark data contamination of large language models: A survey. *arXiv preprint arXiv:2406.04244*, 2024.

[62] Shuo Yang, Wei-Lin Chiang, Lianmin Zheng, Joseph E. Gonzalez, and Ion Stoica. Rethinking benchmark and contamination for language models with rephrased samples, 2023.

[63] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In *ACL*, 2019.

[64] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *ICLR*, 2017.

[65] Guodong Zhang, Chaoqi Wang, Bowen Xu, and Roger Grosse. Three mechanisms of weight decay regularization. *ICLR*, 2019.



(a) 124M 15x Chinchilla (b) OLMo 7B (c) Llama 3 405B

Figure 4: **Theoretical estimates of forgetting and approximate model weight composition at the end of training.** *Top Row:* The cumulative weight decay $w_{t_1}^{t_2}$ as defined in equation (3) for different training runs. The figures depict the decay of the gradient updates for every decile of the training run, indicated in different colors. *Bottom Row:* The approximate composition of the final model weights in terms of the gradient updates from different deciles of the training run. The deciles are indicated in colors depicted in the legend below the plot.

A What is the role of weight decay in forgetting?

Here, we show that the weight decay parameter and learning rate schedule of the AdamW optimizer play a key part in forgetting past training examples. This offers a novel perspective on the interplay between these two parameters, usually seen in terms of generalization and training stability [60, 65, 37, 3].

A.1 Weight Decay as a Mechanism for Forgetting Training Examples

Consider the parameter update of AdamW at gradient step $t \geq 1$. It consists of two decoupled updates [51, 52]: A weight decay update given by

$$\hat{\theta}_t = \theta_{t-1} - \gamma \lambda_t \theta_{t-1}, \tag{1}$$

and a gradient update given by

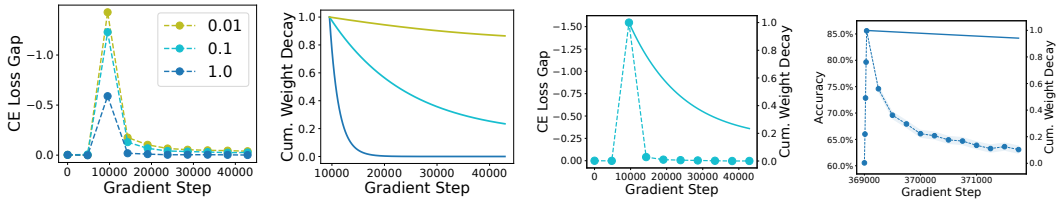
$$\theta_t = \hat{\theta}_t - \lambda_i \, \hat{m}_t / (\sqrt{\hat{v}_t + \epsilon}). \tag{2}$$

Here, θ_t are the model parameters, λ_t is the learning rate, γ is the weight decay parameter, and $\hat{m}_t$ and $\hat{v}_t$ are first- and second-order moment estimates of the gradient. Denoting the model weights at initialization by θ_0, and the adaptive gradient by $\hat{g}_t = \hat{m}_t / (\sqrt{\hat{v}_t + \epsilon})$, we can iterate (1) and (2) to obtain

$$\theta_T = w_0^T \, \theta_0 + \sum_{t=1}^{T} w_t^T \, \lambda_t \, \hat{g}_t \qquad \text{where} \quad w_{t_1}^{t_2} = \prod_{i=t_1}^{t_2} (1 - \lambda_i \gamma). \tag{3}$$

Here, the weights $w_{t_1}^{t_2}$ account for the *cumulative weight decay* between gradient step t_1 and t_2. Intuitively, the model weights after t gradient steps are a weighted average of the initial model weights and all the adaptive gradient updates up to time step t. This is not specific to the AdamW optimizer and applies to every optimizer with weight decay. Analyzing equation (3) reveals a critical factor influencing forgetting: The exponential decay of the $w_{t_1}^{t_2}$ with respect to increasing gap $t_2 - t_1$ in the optimization steps. In other words, the longer an update occurs in the past (i.e., the larger $t_2 - t_1$), the more the contribution of the update $\hat{g}_{t_1}$ is being scaled down due to the exponential decay of weights $w_{t_1}^{t_2}$. We can describe the evolution of these weights as the function of the time $T = t_2 - t_1$.

Proposition 1. *(The Decay of Past Gradients) The number of optimization steps $T = t_2 - t_1$ that are required to make the contribution of a model update at time t_1 small, that is $w_{t_1}^{t_2} \leq \epsilon$ for some small $\epsilon \in \mathbb{R}^+$, scales as $T \gtrsim \frac{log(1/\epsilon)}{\gamma \lambda_{avg}}$, where $\lambda_{avg} = \frac{1}{T} \sum_{t=t_1}^{t_2} \lambda_t$ is the average learning rate of the optimizer between t_1 and t_2.*



(a) Empirical (b) Theory (c) Empirical & Theory (d) Empirical & Theory

Figure 5: **The theoretical estimates provide an upper bound on empirical forgetting, which can occur much faster. (a)** Empirical forgetting for three different weight decay parameters. **(b)** The corresponding cumulative weight decay. **(c)** Empirical forgetting and theoretical estimate for the default weight decay of 0.1. The y-axis of the cumulative weight decay $w_{t_1}^{t_2}$ is calibrated to start at the peak of the empirically observed overfitting. **(d)** Empirical forgetting and theoretical estimate for OLMo-1B (HellaSwag).

We present a proof in the Supplement B.5. If $w_{t_2}^{t_1} \leq \epsilon$ for some sufficiently small ϵ, the term $w_{t_2}^{t_1} \lambda_{t_1} \hat{g}_{t_1}$ vanishes from the sum in (3). Intuitively, this is the same as saying that the gradient update at time step t_1 has been forgotten.

We now analyze the weight-decay mechanism of forgetting in different LLM training runs. Of course, the decay of past gradients described in Proposition 1 is only one effect that contributes to forgetting. Indeed, there is also the potential for different gradient updates in the sum (3) to cancel each other. However, because a term decayed to zero is definitively forgotten, we can consider the cumulative weight decay as an *upper bound* on forgetting. Figure 4 depicts the evolution of the forgetting term $w_{t_1}^{t_2}$ for the 124M parameter model from Section 4.2, OLMo 7B, and Llama 3 405B (where we assume the model trained with a weight decay of 0.1). For the training data at each decline of the training run, Figure 4 depicts *forgetting curves* that indicate how much the corresponding gradients decay as we continue training. For the 124M model depicted in Figure 4a, even the initialization is not completely decayed towards the end (the blue curve is still strictly larger than zero). In contrast, for OLMo-7B, the gradients of the first 40% of the training data decay to zero until the end of training, meaning that this data is forgotten (Figure 4b). Llama 3 405B also experiences significant decay of the early gradients (Figure 4c).

The interplay between the weight decay parameter and learning rate schedule of AdamW creates an interesting dynamic. Lower learning rates towards the end of training (1) slow down the forgetting of past training examples and (2) decrease the impact of later training examples on the final model weights. To better understand this dynamic, we plot the sum of the terms $\lambda_{t_1} w_{t_1}^{t_2}$ for each decile of the training run, normalized by the same sum over the entire training run. This is depicted in the bottom row of Figure 4 and can be thought of as a simple approximation of how the final model weights are composed by the gradients of different training deciles. Interestingly, this approximation suggests that the Llama 3 405B training run, where supposedly a lot of expertise has gone into the choice of the hyperparameters, results in a model where the approximate influence of different training steps is symmetrically distributed around the middle of training (Figure 4c). In contrast, the OLMo-7B training could seemingly benefit from further decaying the learning rate to give more weight to early versus late gradients.

While our analysis in this Section considers the mechanistic effect of individually decaying gradient updates, it does not model any interactions between different gradient updates. For example, if the weight updates at a later time step t_2 were aligned with past updates at t_1, then the model might not forget the information even if the effect of past updates vanishes from the sum. However, such complex interactions are avoided if the model updates from contaminated samples are orthogonal. Formalizing this observation leads to a more rigorous version of Proposition 1, presented in Supplement B.5. The argument is that under suitable gradient orthogonality conditions, the decay of past gradients can guarantee forgetting.

A.2 Practical forgetting occurs faster than what the theory predicts

In Section A.1, we have derived a simple theory of forgetting via cumulative weight decay. We now investigate how the theoretical estimates relate to the empirically observed forgetting.

The main parameter that controls the theoretical forgetting curves is the weight decay parameter. Therefore, we ask how forgetting changes empirically when we change the weight decay. Figure 5a depicts the result of repeating the forgetting experiment from Section 4.2 with three different choices for the weight decay parameter. From Figure 5a, we see that the weight decay parameter controls the impact of contamination at all time steps, where a larger weight decay parameter leads to more forgetting and a smaller weight decay parameter to less forgetting. This is consistent with the theoretical predictions depicted in Figure 5b, meaning there is a *qualitative alignment between the empirical results and our theoretical predictions*.

To better understand the quantitative relation between empirical forgetting and the theoretical estimates, we ask how the empirically forgotten fraction (of cross-entropy loss or accuracy) relates to the cumulative weight decay. Figure 5c depicts the empirical decay and corresponding theoretical prediction for the model from Section 4.2. We see that the theoretical estimate is somewhat pessimistic and that *forgetting occurs faster than what is predicted by the theory*. Figure 5d is similar to Figure 5c, except that we consider the OLMo-1B forgetting experiment from Section 4.3. Figure 5d depicts a case where *forgetting occurs much faster* than what is predicted by the theory. Interestingly, we also see that the empirical rate of forgetting, at least in this experiment, is not smaller for the larger model.

B Proofs and Additional Details

B.1 Additional Discussion of Data Contamination Assumptions and Setting

Here, we discuss our data contamination approach in a bit more detail.

In this paper, we consider only **exact contamination**. This means we contaminate the training data exactly with the text the model is later evaluated on. In the literature, it has been shown that non-exact contamination (re-worded questions, translation into a different language) can affect benchmark performance, too. For example, **(author?)** [62] have shown that a 13B parameter Llama 2 Model [59] can achieve an accuracy increase of over 20 percentage points after training on re-phrased benchmark questions. We decided against considering non-exact contamination for this paper because the models we train from scratch are much smaller than those for which non-exact contamination results have been shown. This means these models are less capable of making sense of related information, potentially leading us to underestimate the effect of non-exact contamination for realistic training runs.

In addition, we consider contamination with **individual benchmark questions**, inserted into the training data at **random** positions. We consider this setup because we are interested in contamination from the perspective of *leakage*, where individual benchmark questions may enter the training data via different documents (for example, as quotes in Wikipedia articles, a case described in **(author?)** [9]). This contrasts with the setup where a dataset is present in the training data as a long contiguous string, which we conjecture might have a similar impact but be easier detectable [48]. The fact that we contaminate with benchmark questions also sets us apart from related works that study data contamination and memorization for random strings and uniquely identified objects [12, 13]. It is worth highlighting that the results between these two setups might differ, especially considering the time it takes to forget an example.

We only consider pre-training.

B.2 Additional Details on Evaluation and How Contamination was Performed

Benchmark Questions and Evaluation. We use code from OLMo [26] to format the different benchmark questions. This code is again based in part on the EleutherAI Evaluation Harness [22]. The benchmark questions are multiple-choice, and the different options are presented to the model zero-shot as possible sentence continuations. The prediction is the sentence continuation with the largest likelihood. For the small GPT-3 models, we normalize by the number of tokens [21]. For OLMo, we rely on the evaluation framework that is part of the code repository.

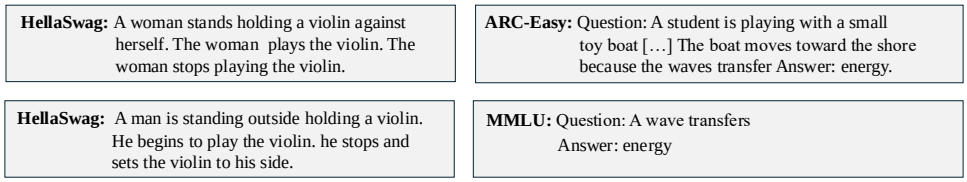


Figure 6: Language modeling benchmarks frequently contain near-duplicate questions. We perform extensive filtering for duplicates using fuzzy string matching. The figure depicts a near-duplicate from HellaSwag and a cross-benchmark duplicate from ARC-Easy/MMLU.

Inserting benchmark questions into the training data. A batch of LLM training data consists of B sequences of S tokens, resulting in a batch size of $B \times S$. For example, OLMo-1B is trained with $B = 2048$ and $S = 2048$; the batch for a single gradient step contains $\sim$4M tokens [26]. Individual sequences in a batch usually contain multiple texts separated by a special end-of-text token. We insert benchmark questions at random positions into the pre-training data, separated at the beginning and end with the end-of-text token.

B.3 Filtering Near-Duplicate Benchmark Questions

Our method requires that there are no side effects from contaminating the training data with one question on the evaluation of another question. However, upon closer inspection, it turns out that *all* the commonly used benchmarks from the literature contain questions that are either near-duplicates or where the context of one question contains the answer to another question (for example, because the same text document was used to create multiple questions). This is illustrated in Figure 6, which depicts two near-duplicate questions on HellaSwag and questions from ARC-Easy and MMLU that are cross-benchmark duplicates. To tackle this problem, we perform fuzzy string matching between the ground-truth options (that is, the potential contamination data) of all benchmark questions, randomly removing one question for every detected duplicate. We use the Python package rapidfuzz.

Summary Statistics. Table 2 depicts summary statistics about the different benchmarks, including the number of questions that were filtered during the duplicate-detection stage. We see that the number of filtered questions is significant. On some datasets, especially Social-i-QA, we had to apply very aggressive filtering to avoid any side-effects during contamination. Hence, the number of removed questions per dataset does not necessarily reflect the actual number of duplicates, but the level of filtering that had to be applied to remove all duplicate questions.

Experimental verification that filtering worked. We verify that our filtering procedure worked by training two models: One that is heavily contaminated (obtaining an accuracy of over 97%), and another model that did not see any contamination. We then evaluate both models on a set of 10,000 benchmark questions that are holdout even for the contaminated model. The contaminated model obtains an accuracy of 42.2%, (95%-CI: 41.2% - 43.2%) on the holdout, while the clean model obtains an accuracy of 41.9% (95%-CI: 41.0% -42.9%). Because the observed accuracy difference is small in absolute terms and lies within the confidence interval, we conclude that there are no significant side-effects in our evaluation procedure.

B.4 Proof of Proposition 1

Proposition 2. *(The Decay of Past Gradients) The number of optimization steps $T = t_2 - t_1$ that are required to make the contribution of a model update at time t_1 small, that is $w_{t_1}^{t_2} \leq \epsilon$ for some small $\epsilon \in \mathbb{R}^+$, scales as $T \gtrsim \frac{log(1/\epsilon)}{\gamma \lambda_{avg}}$, where $\lambda_{avg} = \frac{1}{T}\sum_{t=t_1}^{t_2} \lambda_t$ is the average learning rate of the optimizer between t_1 and t_2.*

Proof. Without loss of generality, mapping $t_1 = 1$ and $T = t_2$, we have from equation 3:

$$w_1^T = \prod_{i=1}^{T}(1 - \lambda_i \gamma)$$

Table 2: Overview of benchmarks used in the paper. This table documents the experiments with GPT-3 models. The first two rows provide the dataset split and corresponding number of benchmark questions. The third row provides the number of questions that were removed from the dataset after filtering each dataset for near-duplicate questions. The fourth row provides the number of questions that were removed after additionally filtering for near-duplicate questions across all the different datasets combined. The fifth row provides the dataset's weight in the dataset splits used in the experiments.

	HellaSwag	PiQA	Social-i-QA	BoolQ	MMLU	WinoGrande	ARC-Easy
Split	Validation	Train	Train	Validation	Test	XL, Train	All
Size	10,042	16,113	33,410	3,269	14,042	40,398	5,197
Filtered	1,416	7,386	29,756	409	4,423	21,944	2,568
Cross-Filtered	3	6	10	2	15	0	13
Weight	19.58%	19.77%	8.27%	6.48%	21.82%	18.16%	5.92%

Assigning the forgetting ratio to be less than ϵ according to our criteria, we have:

$$\prod_{i=1}^{T}(1-\lambda_i\gamma) \leq \epsilon$$

$$\sum_{i=1}^{T}\log(1-\lambda_i\gamma) \leq \log\epsilon$$

$$\sum_{i=1}^{T}(-\lambda_i\gamma) \lesssim \log\epsilon \quad (\log(1-x)\approx -x \text{ for small } x)$$

$$T\times\left(\frac{1}{T}\sum_{i=1}^{T}\lambda_i\right)\times\gamma \gtrsim \log\frac{1}{\epsilon}$$

Re-arranging this equation gives us the desired result, where $\lambda_{avg} = (\frac{1}{T}\sum_i \lambda_i)$. □

B.5 Extended Analysis of Forgetting & Gradient Alignment

To understand the effect of weight decay on forgetting, we analyze two different stages of optimization: (1) the contamination stage, where the training set consists of only the contaminated samples, and (2) the forgetting stage, where the training set consists only of clean samples. Here, we consider the SGD learning algorithm, but the resulting analysis also applies to SGD with momentum. In particular, to illustrate the effect of weight decay, we assume the usage of SGD for the contamination stage and SGD with weight decay for the forgetting stage.

We now introduce some notation. Let $\theta \in \mathbb{R}^D$ be the weights of the model, and let $\mathcal{X}_{\text{cont}} = \{\mathbf{x}_1^{\text{cont}}, \mathbf{x}_2^{\text{cont}}, ...\mathbf{x}_{N_{\text{cont}}}^{\text{cont}}\}$ be the contamination set and $\mathcal{X}_{\text{clean}} = \{\mathbf{x}_1^{\text{clean}}, \mathbf{x}_2^{\text{clean}}, ...\mathbf{x}_{N_{\text{clean}}}^{\text{clean}}\}$ be the clean pre-training set. The training data used for the contamination stage is thus $\mathcal{X}_{\text{cont}}$, and the training data for the forgetting stage is $\mathcal{X}_{\text{clean}}$. Let $\ell(\mathbf{x}_i) \in \mathbb{R}$ be the loss associated with sample $\mathbf{x}_i$. Let the model be initialized at the contamination stage with $\theta = \theta_{\text{init}}$. The learning algorithm is (single batch size) SGD, which is run for a total of N_{cont} steps for the contamination stage and N_{clean} steps for the forgetting stage. First, we observe that the weights at the end of the contamination stage are:

$$\theta' = \theta^{\text{init}} - \underbrace{\sum_{i=1}^{N_{cont}} \lambda_i \nabla_{\theta i}\ell(\mathbf{x}_i^{\text{cont}})}_{\theta^{\text{cont}}}$$

We thus denote the weights at the end of the contamination stage as $\theta' = \theta^{\text{init}} + \theta^{\text{cont}}$. For the subsequent fine-tuning stage to forget information regarding samples $\mathcal{X}_{\text{cont}}$, we first define a criterion to identify forgetting based on the angle between a weight vector and the contaminated part identified above.

Forgetting Criteria. A model with weights θ is said to have "forgetten" information contained in θ^{cont} if $\frac{|\theta^\top \theta^{\text{cont}}|}{\|\theta^{\text{cont}}\|_2^2} \leq \epsilon$, which we call the "forgetting ratio". Here, $\epsilon \in \mathbb{R}^+$ is a small constant.

We now proceed with an analysis of the forgetting stage. To enable this, we make the following important assumption that the gradients of clean and contaminated samples are orthogonal for all clean and contaminated samples across all optimization steps. This is a relatively strong assumption, and it quantifies the intuition that model updates required by SGD to memorize clean samples and contaminated samples are distinct.

Assumption 1. (Gradient Orthogonality) $\nabla_{\theta t_1} \ell(\mathbf{x}_i^{\text{clean}})^\top \nabla_{\theta t_2} \ell(\mathbf{x}_j^{\text{cont}}) = 0, \forall i \in [1, N_{\text{clean}}], \forall j \in [1, N_{\text{cont}}]$ and $\forall$ steps t_1, t_2.

We also make another minor simplifying assumption that the weight initialization is orthogonal to both these quantities.

Assumption 2. (Gradient-Initialization Orthogonality) $\nabla_{\theta t} \ell(\mathbf{x}_i)^\top \theta_{init} = 0, \forall i$ and $\forall$ steps t.

Given these assumptions, we are ready to state our result.

Proposition 3. *(Forgetting Time) The number of optimization steps T_{forget} in the forgetting stage, such that the weights $\theta_{T_{forget}}$ satisfy the ϵ-forgetting criteria is given by: $T_{forget} \gtrsim \frac{\log(1/\epsilon)}{\lambda_{avg}\gamma}$, where $\lambda_{avg} = \frac{1}{T}\sum_{i=1}^{T} \lambda_i$ is the average learning rate of the optimizer.*

Proof. We first compute the forgetting ratio at $\theta = \theta'$, and as a consequence of Assumption 2, verify that the forgetting ratio is equal to one.

Let us now denote these weights as $\theta'_0 = \theta'$, used as initialization for the forgetting stage. Analyzing the first optimization step, and the subsequent forgetting ratio, we have:

$$
\begin{aligned}
\text{(Optimization Step)} \quad \theta'_1 &= \theta'_0 - \lambda_0 \nabla_{\theta 0} \ell(\mathbf{x}_0^{\text{clean}}) - \lambda_0 \gamma \theta'_0 \\
\text{(Forgetting ratio)} \quad \frac{|{\theta'_1}^\top \theta^{\text{cont}}|}{\|\theta^{\text{cont}}\|_2^2} &= \frac{|(\theta'_0 - \lambda_0 \nabla_{\theta 0} \ell(\mathbf{x}_0^{\text{clean}}) - \lambda_0 \gamma \theta'_0)^\top \theta^{\text{cont}}|}{\|\theta^{\text{cont}}\|_2^2} \\
&= \frac{|((\theta^{\text{init}} + \theta^{\text{cont}}) - \lambda_0 \nabla_{\theta 0} \ell(\mathbf{x}_0^{\text{clean}}) - \lambda_0 \gamma((\theta^{\text{init}} + \theta^{\text{cont}})))^\top \theta^{\text{cont}}|}{\|\theta^{\text{cont}}\|_2^2} \\
&= (1 - \lambda_0 \gamma) \qquad \text{(From Assumptions 1 \& 2)}
\end{aligned}
$$

We can similarly analyze the subsequent optimization steps to compute the forgetting ratio, which for some step $t+1$ is:

$$
\begin{aligned}
\underbrace{\frac{|{\theta'_{t+1}}^\top \theta^{\text{cont}}|}{\|\theta^{\text{cont}}\|_2^2}}_{\text{Forgetting ratio at } t+1} &= \frac{|(\theta'_t - \lambda_t \nabla_{\theta t} \ell(\mathbf{x}_t^{\text{clean}}) - \lambda_t \gamma \theta'_t)^\top \theta^{\text{cont}}|}{\|\theta^{\text{cont}}\|_2^2} \\
&= \underbrace{\frac{|{\theta'_t}^\top \theta^{\text{cont}}|}{\|\theta^{\text{cont}}\|_2^2}}_{\text{Forgetting ratio at } t} (1 - \lambda_t \gamma)
\end{aligned}
$$

Unrolling the recurrence till step T, we have that:

$$
\frac{|{\theta'_T}^\top \theta^{\text{cont}}|}{\|\theta^{\text{cont}}\|_2^2} = \underbrace{\frac{|{\theta'_0}^\top \theta^{\text{cont}}|}{\|\theta^{\text{cont}}\|_2^2}}_{=1} \times \prod_{i=1}^{T} (1 - \lambda_i \gamma)
$$

Assigning the forgetting ratio to be less than ϵ according to our criteria, we have:

$$\prod_{i=1}^{T}(1-\lambda_i\gamma) \leq \epsilon$$

$$\sum_{i=1}^{T}\log(1-\lambda_i\gamma) \leq \log\epsilon$$

$$\sum_{i=1}^{T}(-\lambda_i\gamma) \lesssim \log\epsilon \quad (\log(1-x)\approx -x \text{ for small } x)$$

$$T\times\left(\frac{1}{T}\sum_{i=1}^{T}\lambda_i\right)\times\gamma \gtrsim \log\frac{1}{\epsilon}$$

Re-arranging this equation gives us the desired result, where $\lambda_{avg} = (\frac{1}{T}\sum_i \lambda_i)$. □

B.6 Reproducibility Statement

The results in this paper were obtained using the OLMo codebase, available at `https://github.com/allenai/OLMo`, and the llm.c codebase, available at `https://github.com/karpathy/llm.c`. Our code is fully reproducible, including the random positions at which benchmark questions were inserted into the training data. Our code is available at `https://github.com/tml-tuebingen/forgetting-contamination/`. We trained on the 100BT split of the FineWeb-Edu dataset, available at `https://huggingface.co/datasets/HuggingFaceFW/fineweb-edu`.