Causal Estimation of Memorisation Profiles

Pietro Lesci,[UC] Clara Meister,[ETH] Thomas Hofmann,[ETH] Andreas Vlachos,[UC] Tiago Pimentel[ETH]

[UC]University of Cambridge, [ETH]ETH Zürich

{pl487, av308}@cam.ac.uk

{clara.meister, thomas.hofmann, tiago.pimentel}@inf.ethz.ch

Abstract

Understanding memorisation in language models has practical and societal implications, e.g., studying models' training dynamics or preventing copyright infringements. Prior work defines memorisation as the causal effect of training with an instance on the model's ability to predict that instance. This definition relies on a counterfactual: the ability to observe what would have happened had the model not seen that instance. Existing methods struggle to provide computationally efficient and accurate estimates of this counterfactual. Further, they often estimate memorisation for a model architecture rather than for a specific model instance. This paper fills an important gap in the literature, proposing a new, principled, and efficient method to estimate memorisation based on the difference-in-differences design from econometrics. Using this method, we characterise a model's memorisation profile—its memorisation trends across training—by only observing its behaviour on a small set of instances throughout training. In experiments with the Pythia model suite, we find that memorisation (i) is stronger and more persistent in larger models, (ii) is determined by data order and learning rate, and (iii) has stable trends across model sizes, thus making memorisation in larger models predictable from smaller ones.

🔗 | pietrolesci/memorisation-profiles

1 Introduction

Large language models (LMs) are often pretrained with a single pass on web-scale datasets (Raffel et al., 2020; Gao et al., 2020; Penedo et al., 2023, *inter alia*). Given the colossal size of these training sets, one may expect each individual instance to have little impact on the final model. Yet, LMs can still reproduce entire sequences from their training set verbatim (Carlini et al., 2021), suggesting that models can store, or *memorise*, precise knowledge about individual training instances. In the era of
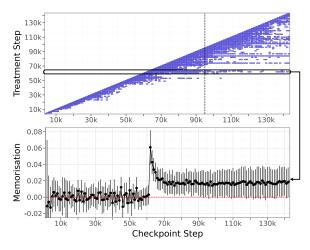

Figure 1: Memorisation profile (top) and path (bottom) of Pythia 6.9B. Each entry represents the expected counterfactual memorisation of instances trained on at a specific timestep ("Treatment Step") across model checkpoints ("Checkpoint Step"). The dashed vertical line indicates the end of the first epoch.

large LMs, measuring memorisation is crucial for NLP practitioners; it has implications for copyright and data protection (Hu et al., 2022; Vyas et al., 2023; Lee et al., 2023), for how models encode factual information (Cao et al., 2022; Tirumala et al., 2022), and for understanding their training dynamics (Arpit et al., 2017; Chang and Bergen, 2024).

One line of prior work has adopted a causal definition of **memorisation**: it is the causal effect of observing an instance during training on a model's ability to correctly predict that instance (Feldman, 2020). Despite being an intuitive concept, quantification of this definition is not straightforward as it requires knowledge of a *counterfactual*:[1] we must know how our model would have performed on a training instance had the model not been trained on it. To overcome this challenge, prior work has proposed a variety of methods to estimate memorisa-

---

[1]Counterfactuals are thought experiments about what would have happened if a present condition were changed while keeping everything else unchanged (McCloskey, 2016).

tion. Some estimate it by training a model multiple times on different subsets of the data (e.g., Feldman and Zhang, 2020; Zheng and Jiang, 2022), while others implicitly assume this counterfactual's value to be negligible (Carlini et al., 2021). Both these approaches, however, have drawbacks: the first computes memorisation for an architecture rather than a specific model, while the second relies on a strong assumption (we discuss this in detail in §5).

In this paper, we first formalise **counterfactual memorisation** as the difference between two potential outcomes; notably, our formalisation generalises prior definitions of memorisation, allowing us to compare them within a unified framework. We then draw from the econometrics literature (Callaway and Sant'Anna, 2021) and propose a new method which estimates memorisation using only observational data; our method simply needs a model's performance measurements (e.g., log-likelihood) on a subset of the training data throughout training. The output of our method is what we term a **memorisation profile**: a model's memorisation of training batches over the course of training.

Empirically, we use this method to analyse memorisation for models in the Pythia suite (Biderman et al., 2023b) and characterise their memorisation profiles; e.g., Fig. 1 reports the memorisation profile of Pythia 6.9B. By studying these memorisation profiles we find that memorisation is stronger and more persistent in larger models. Furthermore, both the learning rate and the position of an instance in the training set considerably impact how strongly that instance is memorised. Finally, memorisation profiles are stable across model sizes; thus, we can predict memorisation in larger models from the memorisation observed in smaller ones.

2 Background

In this section, we introduce some background on language modelling and on causal analysis which will be required throughout our paper.

2.1 Language Modelling

We start by providing some background on language modelling.[2] Let $p_{\theta}(x)$ be a language model with parameters $\theta \in \mathbb{R}^d$. This model defines a probability distribution over $x \in \mathcal{V}^*$, the set of all finite sequences that can be constructed from elements in the alphabet $\mathcal{V}$. To train this model,

we start with a set of randomly selected initial parameters, θ_0. We then learn parameters θ using a dataset $\mathcal{D}$ and an optimisation procedure defined with respect to a loss function $\mathcal{L}$.

Specifically, let $\mathcal{D} = \{x_n\}_{n=1}^N$ be a dataset whose **instances** x are sequences drawn from a target (unknown) distribution $p(x)$. These instances are typically assumed to be sampled i.i.d., and are shuffled using a permutation function $\sigma \colon \{1, ..., N\} \to \{1, ..., N\}$. For a given batch size B, we then split this dataset into $T \leq \lfloor N/B \rfloor$ batches $\mathcal{B}_t$. We iterate through these batches performing gradient updates on the model parameters:

$$\theta_t = \theta_{t-1} - \eta \nabla_{\theta} \mathcal{L}(\theta_{t-1}, \mathcal{B}_t) \qquad (1)$$

where $\eta \in \mathbb{R}$ is a learning rate.[3] Notably, this procedure consists of a single pass on the training set and is standard for recent LMs (e.g., Touvron et al., 2023; Jiang et al., 2023; Dey et al., 2023).

At each iteration, we use a batch $\mathcal{B}_t$ to obtain a new model checkpoint, θ_t. We introduce a notation which distinguishes the indexing of checkpoints and batches. We use $c \in \{0, 1, ..., T\}$ to denote a **checkpoint step** (e.g., θ_c). Further, we use $g \in \{1, ..., T\} \cup \{\infty\}$ to denote the timestep at which a batch is used for training (e.g., $\mathcal{B}_g$); we term this a **treatment step**, borrowing this terminology from the econometrics literature. We denote as $g = \infty$ a batch composed of instances that are not used for training and which form a validation set.

2.2 Causal Analysis

Causal estimation is typically split into three steps. First, we define a **causal estimand**, the target quantity we want to estimate. Second, we state the assumptions needed to rewrite this causal estimand in terms of observable data, thus defining a **statistical estimand**; this process is called identification. Finally, we define an **estimator**, a statistical procedure to approximate the statistical estimand.

To formally define memorisation as a causal estimand, we will use the **potential outcomes** framework of Rubin (1974, 2005).[4] This framework allows us to formally describe the causal effect of an intervention, or **treatment**, on some target quantity, or **outcome**. In §1, we defined memorisation as the causal effect of *training on* an instance on a

---

[2]We frame our exposition in terms of language models, but our framework can be trivially applied to any neural model and input modality (e.g., images).

[3]The learning rate can be a function of other quantities (e.g., Duchi et al., 2011; Kingma and Ba, 2015, *inter alia*).

[4]For a comparison of causal frameworks, see Ibeling and Icard (2023). For an introduction to causal inference, see Pearl (2009) and Imbens and Rubin (2015).

model's ability to *predict* it. Thus, the act of training on $\boldsymbol{x}$ defines the treatment, while the model's ability to predict an instance defines the outcome.

Since training is performed iteratively over batches, instances are treated at different timesteps. Thus, we use a **treatment assignment variable** $G(\boldsymbol{x})$ to denote the step g an instance is trained on. Further, to quantify the ability of a model with parameters $\boldsymbol{\theta}_c$ to predict $\boldsymbol{x}$, we use a performance function γ. We then define the **outcome variable** as $Y_c(\boldsymbol{x}) \stackrel{\text{def}}{=} \gamma(\boldsymbol{\theta}_c, \boldsymbol{x})$, and, unless noted otherwise, we set this performance function to be the log-likelihood of $\boldsymbol{x}$ under $p_{\boldsymbol{\theta}_c}$: $\gamma(\boldsymbol{\theta}_c, \boldsymbol{x}) = \log p_{\boldsymbol{\theta}_c}(\boldsymbol{x})$.[5]

To define memorisation we need to represent both observed—i.e., $Y_c(\boldsymbol{x})$—and counterfactual outcomes—i.e., the performance of the model on $\boldsymbol{x}$ had we not trained on it. The potential outcomes notation (Splawa-Neyman, 1923) enables us to represent both types of outcomes consistently.

Definition 1. *The **potential outcome** of an instance $\boldsymbol{x}$ at timestep c under treatment assignment g, denoted as $Y_c(\boldsymbol{x}; g)$, is the value that the outcome would have taken if $G(\boldsymbol{x})$ was equal to g.*

Since we only observe a single permutation of the data, we only see one specific treatment step for each instance, i.e., $G(\boldsymbol{x})$. Thus, the potential outcome of an instance is observed only for the actual treatment assignment $g = G(\boldsymbol{x})$. In this case, we can equate potential and observed outcomes, that is $Y_c(\boldsymbol{x}; g) = Y_c(\boldsymbol{x})$, a property called consistency (Cole and Frangakis, 2009). For any other treatment step $g \neq G(\boldsymbol{x})$, the potential outcome is counterfactual and, thus, unobservable from the data.

3 Counterfactual Memorisation

Intuitively, counterfactual memorisation can be understood as the answer to the question: how would the model's performance on instance $\boldsymbol{x}$ at timestep c be different if we had not trained on it at timestep g? Using the potential outcomes notation, we formalise this definition as follows.

Definition 2. *Counterfactual memorisation is the causal effect of using instance $\boldsymbol{x}$ for training at the observed timestep $G(\boldsymbol{x}) = g$ on the model's performance on this same instance at timestep c:*

$$\tau_{\boldsymbol{x},c} \stackrel{\text{def}}{=} \underbrace{Y_c(\boldsymbol{x}; g)}_{\substack{\text{performance on } \boldsymbol{x} \\ \text{when trained with } \boldsymbol{x}}} - \underbrace{Y_c(\boldsymbol{x}; \infty)}_{\substack{\text{performance on } \boldsymbol{x} \\ \text{when not trained with } \boldsymbol{x}}} \quad (2)$$

---

[5]We experiment with other functions in the appendix.

In econometrics, eq. (2) is called an individual treatment effect (ITE). Notably, the first potential outcome in this equation, $Y_c(\boldsymbol{x}; g)$, can be observed from the data since, by definition, we trained on $\boldsymbol{x}$ at timestep $G(\boldsymbol{x}) = g$. However, the second term, $Y_c(\boldsymbol{x}; \infty)$, is counterfactual. To compute the ITE, we would need to estimate this counterfactual outcome for a specific instance, which is challenging due to unobserved factors and heterogeneity[6] (Lu et al., 2018). While we would ideally estimate memorisation at the instance level, we focus on average effects instead, as is common in the econometrics literature (Angrist and Pischke, 2015).

Definition 3. *Expected counterfactual memorisation is the average causal effect of using instances for training at timestep g on the model's performance on these same instances at timestep c:*[7]

$$\tau_{g,c} \stackrel{\text{def}}{=} \mathbb{E}_{\boldsymbol{x}}\left[Y_c(\boldsymbol{x}; g) - Y_c(\boldsymbol{x}; \infty) \mid G(\boldsymbol{x}) = g\right] \quad (3)$$

Together, the $\tau_{g,c}$ form a matrix which we term the model's **memorisation profile**; each row therein is the **memorisation path** of a batch. Memorisation profiles and paths allow us to analyse a model's memorisation patterns across different treatment and checkpoint steps. Notably, there cannot be memorisation whenever $c < g$, as the instances have not been seen by the model yet, so $\tau_{g,c} = 0$ in those cases. We term $\tau_{g,c}$ as **instantaneous memorisation** when $c = g$, as **persistent memorisation** when $c > g$, and as **residual memorisation** when $c = T$.

4 Estimating Memorisation

The practical implication of defining memorisation at a treatment level g is that we can only make causal claims for groups of instances treated at the same timestep (i.e., a batch $\mathcal{B}_g$), rather than for individual instances. However, even though this approach simplifies the problem, estimating eq. (3) still poses major challenges as it contains a counterfactual. A simple decomposition makes

---

[6]I.e., non-random variability across instances.

[7]In econometrics, eq. (3) is called an average treatment effect on the treated (ATT), as it is defined in terms of an expectation over $\boldsymbol{x} \sim p(\boldsymbol{x} \mid G(\boldsymbol{x}) = g)$. In other words, this expectation is taken with respect to the instance distribution conditioned on it being selected for training at step g. Assuming the training set is sampled i.i.d. and that its permutation is random (as discussed in §2.1), then $p(\boldsymbol{x} \mid G(\boldsymbol{x}) = g) = p(\boldsymbol{x})$. Given these assumptions, eq. (3) would also be an average treatment effect (ATE), which would allow us to make causal claims about the entire population.

this counterfactual explicit:

$$\tau_{g,c} = \tag{4}$$
$$\underbrace{\mathbb{E}_{\boldsymbol{x}}\big[Y_c(\boldsymbol{x}; g) \mid G(\boldsymbol{x}) = g\big]}_{\text{①}} - \underbrace{\mathbb{E}_{\boldsymbol{x}}\big[Y_c(\boldsymbol{x}; \infty) \mid G(\boldsymbol{x}) = g\big]}_{\text{②}}$$

Expectation ① can be directly estimated from the data because batch $\mathcal{B}_g$ contains a set of examples $\boldsymbol{x}$ for which $G(\boldsymbol{x}) = g$ and, thus, we can invoke the consistency assumption to equate $Y_c(\boldsymbol{x}; g)$ with the observed outcome $Y_c(\boldsymbol{x})$. Let us define the mean across instances in a batch as:

$$\overline{Y}_c(g) \stackrel{\text{def}}{=} \frac{1}{|\mathcal{B}_g|} \sum_{\boldsymbol{x} \in \mathcal{B}_g} Y_c(\boldsymbol{x}) \tag{5}$$

We can thus use eq. (5) as an estimator for expectation ①. Expectation ②, however, is counterfactual: we cannot observe the potential outcome $Y_c(\boldsymbol{x}; \infty)$ for instances treated at timestep $g \neq \infty$. The presence of counterfactual potential outcomes in causal estimands creates challenges for their estimation, being known as the fundamental problem of causal inference (Holland, 1986). The goal of causal methods is then to estimate these counterfactual outcomes from observed ones, using comparable groups of instances. Thus far, we have defined our causal estimand. In this section, we perform steps two and three of causal estimation (§2.2): we derive two statistical estimands for our causal estimand (identifying it under specific assumptions), and provide concrete estimators for them.

4.1 The Difference Estimator

Our first approach to estimate memorisation is straightforward and only requires the observed outcomes of a held-out validation set. However, it relies on a strong identification assumption.

Assumption 1 (I.i.d. Dataset Sampling). *Instances $\boldsymbol{x}$ are independently and identically distributed, following $p(\boldsymbol{x})$, and are randomly assigned to treatment groups g.*

Under this assumption, we have that $p(\boldsymbol{x} \mid G(\boldsymbol{x}) = g) = p(\boldsymbol{x} \mid G(\boldsymbol{x}) = \infty) = p(\boldsymbol{x})$. Thus, the following statistical estimand identifies $\tau_{g,c}$:

$$\tau_{g,c}^{\text{diff}} = \mathbb{E}_{\boldsymbol{x}}\big[Y_c(\boldsymbol{x}; g) \mid G(\boldsymbol{x}) = g\big] \tag{6}$$
$$- \mathbb{E}_{\boldsymbol{x}}\big[Y_c(\boldsymbol{x}; \infty) \mid G(\boldsymbol{x}) = \infty\big]$$

(See Lemma 1 in App. A.1 for a proof.) Note that, unlike eq. (3), the second term in this estimand is not counterfactual: it is the expected observed outcome of validation instances, $\mathcal{B}_\infty$. This statistical estimand can then be estimated as follows.

Estimator 1. *The **difference estimator**, defined as:*

$$\widehat{\tau}_{g,c}^{\text{diff}} = \overline{Y}_c(g) - \overline{Y}_c(\infty) \tag{7}$$

is an unbiased estimator of $\tau_{g,c}$ under Assump. 1.

Proof. See Lemma 2 in App. A.1 for a proof. □

Notably, Assump. 1 is satisfied by the training procedure we described in §2.1, and is commonly true in machine learning. However, it might not hold in general, as the train and validation distributions may not match exactly. For example, NLP practitioners might deduplicate their training set but not validation (Biderman et al., 2023b) or might use challenge sets for validation (Kiela et al., 2021). Moreover, even when Assump. 1 holds, eq. (7) is low-variance only if we have large enough samples to compute $\overline{Y}_c(g)$ and $\overline{Y}_c(\infty)$. Unfortunately, given the size of state-of-the-art LMs and their datasets, it can be expensive—both in terms of computation and memory usage—to extract the performance measures $Y_c(\boldsymbol{x})$ for all instances $\boldsymbol{x}$ and checkpoints c. Even with unlimited compute, $\overline{Y}_c(g)$ can only be estimated using instances in $\mathcal{B}_g$, which lower bounds the variance of this estimator.

While the difference estimator is a first step towards a principled estimator of counterfactual memorisation, we can do better. In the next section, we describe an estimator that has lower variance and requires weaker assumptions.

4.2 The Difference-in-Differences Estimator

We now introduce another causal estimator based on the difference-in-differences (DiD) design. The intuition behind DiD is to use the time dimension to help with identification; DiD identifies a causal estimand using the difference in the *trends* over time of the outcome on treated vs. untreated instances. In our specific setting, DiD relies on the assumption that changes in model performance over time would follow similar trends in different batches if they had not been used for training. We formalise this assumption as follows.

Assumption 2 (Parallel Trends). *In the absence of training, the expected change in model performance across checkpoints would be the same regardless of treatment. That is, for all $c, c' \geq g - 1$:*

$$\mathbb{E}_{\boldsymbol{x}}\big[Y_c(\boldsymbol{x}; \infty) - Y_{c'}(\boldsymbol{x}; \infty) \mid G(\boldsymbol{x}) = g\big] \tag{8}$$
$$= \mathbb{E}_{\boldsymbol{x}}\big[Y_c(\boldsymbol{x}; \infty) - Y_{c'}(\boldsymbol{x}; \infty) \mid G(\boldsymbol{x}) = \infty\big]$$

We need a second assumption before we can apply the DiD design to our setting.

Assumption 3 (No Anticipation). *Training has no effect before it happens. That is, for all $c < g$:*

$$\mathbb{E}_{\boldsymbol{x}} \left[Y_c(\boldsymbol{x}; g) \mid G(\boldsymbol{x}) = g \right] \tag{9}$$
$$= \mathbb{E}_{\boldsymbol{x}} \left[Y_c(\boldsymbol{x}; \infty) \mid G(\boldsymbol{x}) = g \right]$$

Given these two assumptions, we can now follow [Callaway and Sant'Anna (2021)](#) in identifying our target statistical estimand.[8] Combined, these assumptions allow us to rewrite expectation ② in eq. (4) as a function of potential outcomes that are observable: $Y_c(\boldsymbol{x}; \infty)$, $Y_{g-1}(\boldsymbol{x}; \infty)$ given $G(\boldsymbol{x}) = \infty$ are observable on a held-out validation set, while $Y_{g-1}(\boldsymbol{x}; g)$ given $G(\boldsymbol{x}) = g$ is observable on the training set. The following statistical estimand thus identifies $\tau_{g,c}$:

$$\tau_{g,c}^{\texttt{did}} = \mathbb{E}_{\boldsymbol{x}}[Y_c(\boldsymbol{x}; g) - Y_{g-1}(\boldsymbol{x}; g) \mid G(\boldsymbol{x}) = g] \tag{10}$$
$$- \mathbb{E}_{\boldsymbol{x}}[Y_c(\boldsymbol{x}; \infty) - Y_{g-1}(\boldsymbol{x}; \infty) \mid G(\boldsymbol{x}) = \infty]$$

(See Lemma 3 in App. A.2 for a proof.) This leads to the following DiD estimator.

Estimator 2. *The **difference-in-differences estimator (DiD)**, defined as:*

$$\widehat{\tau}_{g,c}^{\texttt{did}} = \underbrace{\left(\overline{Y}_c(g) - \overline{Y}_{g-1}(g) \right)}_{\text{diff in trained}} - \underbrace{\left(\overline{Y}_c(\infty) - \overline{Y}_{g-1}(\infty) \right)}_{\text{diff in untrained}}$$
$$\tag{11}$$

is an unbiased estimator of $\tau_{g,c}$ under Assumps. 2 and 3.

Proof. See Lemma 4 in App. A.2 for a proof. □

The DiD estimator depends on weaker assumptions and has a lower variance (under mild assumptions, see App. A.3) than the difference estimator in eq. (7). Specifically, the parallel trends assumption (Assump. 2) is strictly weaker than the i.i.d. one (Assump. 1): if $p(\boldsymbol{x} \mid G(\boldsymbol{x}) = g) = p(\boldsymbol{x} \mid G(\boldsymbol{x}) = \infty)$, then it is trivial that performances should present parallel trends. Moreover, Assump. 2 only requires that the training and validation sets follow similar trends on average, which

---

[8]The DiD design was originally proposed for the case with only two treatment and checkpoint steps (i.e., $g \in \{1, \infty\}$ and $c \in \{0, 1\}$). Previous work has shown the challenges of extending DiD to multiple timesteps, especially when allowing for heterogeneous treatment effects ([Roth et al., 2023](#)). [Callaway and Sant'Anna (2021)](#) propose an extension which identifies eq. (3) while allowing for treatment effect heterogeneity across checkpoint and treatment steps.

might be true even in the case of, e.g., challenge validation sets or deduplicated training data. Therefore, the assumptions underpinning DiD are more likely to hold in practice and we will use it to estimate memorisation here.

In practice, the difference-in-differences estimation procedure includes two steps. First, we compute the model's performance on a subset of analysed instances—i.e., samples from the training and validation sets—using the available checkpoints; thus forming a **panel** of observed outcomes, as it is usually termed in econometrics. Then, we use this panel to compute the estimates in eq. (11).

5 Prior Notions of Memorisation

Memorisation has recently received much attention ([Arpit et al., 2017](#); [Carlini et al., 2019](#), [2021](#); [Anagnostidis et al., 2023](#), *inter alia*).[9] Prior work has studied how model architecture and training choices influence memorisation ([Tirumala et al., 2022](#); [Kandpal et al., 2022](#); [Lee et al., 2022](#); [Biderman et al., 2023a](#)), and where memorised instances are stored within a model ([Maini et al., 2023](#); [Stoehr et al., 2024](#)). In this section, we use our framework to discuss three prior notions of memorisation which we consider most relevant to our paper: previous operationalisations of counterfactual memorisation (e.g., [Feldman, 2020](#)), influence functions ([Zheng and Jiang, 2022](#)), and extractable memorisation ([Carlini et al., 2023](#)).

5.1 Previous Operationalisations of Counterfactual Memorisation

As mentioned before, estimating an instance's memorisation $\tau_{\boldsymbol{x},c}$ is non-trivial due to the counterfactual component in its definition. We avoid this issue by estimating expected memorisation $\tau_{g,c}$ instead. Prior work ([Feldman, 2020](#); [Feldman and Zhang, 2020](#); [Zhang et al., 2023](#); [Lukasik et al., 2024](#)) takes a different approach, comparing the performance of models trained with and without that instance. In doing so, they average performance across training runs, measuring what we term **architectural counterfactual memorisation**.

Formally, let $\boldsymbol{\psi}$ be a vector of variables responsible for training variance. This includes, e.g., the data permutation induced by σ and the initial model parameters $\boldsymbol{\theta}_0$. By defining a distribution $p(\boldsymbol{\psi})$ over these variables, architectural memorisation can be defined as follows.

---

[9]See [Hartmann et al. (2023)](#) or [Ishihara (2023)](#) for surveys.

Definition 4. *Architectural counterfactual memorisation is the counterfactual memorisation $\tau_{x,T}$ when marginalising over training variables ψ:*

$$\tau_{x,p(\psi)} \stackrel{\text{def}}{=} \mathop{\mathbb{E}}_{\psi}\left[\tau_{x,T} \,|\, G(x) \neq \infty\right] \quad (12)$$

$$= \mathop{\mathbb{E}}_{\psi}\left[Y_T(x; G(x)) - Y_T(x; \infty) \,|\, G(x) \neq \infty\right]$$

where $G(x)$ in the first potential outcome depends on which batch the shuffling function σ puts x in.

Prior work has proposed a number of methods to estimate this value (Bachmann et al., 2022; Lin et al., 2022; Ilyas et al., 2022; Park et al., 2023). The simplest of these is to train several models while including or not x in the training set; these models are then used to approximate the expectation above. We describe a statistical estimand and estimator for $\tau_{x,p(\psi)}$ in App. B.1, discussing the assumptions needed by this approach.

Notably, this operationalisation has the advantage of estimating memorisation at the instance level. However, it also has drawbacks—beyond just being computationally expensive to estimate. These become apparent upon closer inspection of the definition of $\tau_{x,p(\psi)}$. First, it does not provide insights into the effect of the checkpoint step or treatment step on memorisation; this is because T is hard-coded into $\tau_{x,p(\psi)}$'s definition and because it marginalises over permutations of the data. While it is trivial to generalise this definition to other checkpoint steps c or to specific g, prior work has mainly focused on these choices, overlooking the impact of training dynamics on memorisation. Further, and perhaps more importantly, marginalising over $p(\psi)$ means that this metric quantifies memorisation for a model architecture, rather than for a specific model.

5.2 Influence Functions

Influence functions (Hampel, 1974; Cook and Weisberg, 1980) estimate—without re-training a model—how its parameters would change if an instance x was removed from the training set. Specifically, the new set of parameters can be approximated as follows (Koh and Liang, 2017):

$$\boldsymbol{\theta}_{-x,T} \approx \boldsymbol{\theta}_T + \nicefrac{1}{N}\,\mathrm{H}_{\boldsymbol{\theta}}^{-1}\,\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}_T, x) \quad (13)$$

where $\mathrm{H}_{\boldsymbol{\theta}}$ is the hessian of $\mathcal{L}$ evaluated at $\boldsymbol{\theta}_T$. This approximation is based on a first-order Taylor expansion of the training objective around $\boldsymbol{\theta}_T$, and should lead to small errors under the following

assumptions: (i) the loss function is strictly convex in $\boldsymbol{\theta}$, (ii) $\mathrm{H}_{\boldsymbol{\theta}}$ is a positive-definite matrix, and (iii) the model has converged (i.e., the gradient is zero). Given these assumptions, influence functions can be used to efficiently estimate the counterfactual term $Y_T(x; \infty)$ in the definition of $\tau_{x,T}$. Specifically, we can measure the model's performance using the updated parameters, $Y_{-x,T}(x) \stackrel{\text{def}}{=} \gamma(\boldsymbol{\theta}_{-x,T}; x)$, and equate $Y_T(x; \infty) = Y_{-x,T}(x)$. The influence function estimator of memorisation can then be written as:

$$\hat{\tau}_{x,T}^{\text{infl}} = Y_T(x) - Y_{-x,T}(x) \quad (14)$$

We formalise this estimator and its statistical estimand in App. B.2. Notably, Zheng and Jiang (2022) use a similar approach to estimate memorisation in a classification setting.

Influence functions thus provide a computationally efficient method to estimate instance-level counterfactual memorisation. However, none of the required assumptions above is typically satisfied for LMs, which can lead to strong biases in this estimator (Basu et al., 2020; Bae et al., 2022; Schioppa et al., 2023). Moreover, assumptions (ii) and (iii) require $\boldsymbol{\theta}_T$ to be locally optimal, meaning that this approach is not applicable for studying $c < T$. We therefore cannot use it to study how memorisation interacts with training dynamics.

5.3 Extractable Memorisation

Carlini et al. (2023) defines memorisation as (k,ℓ)-extractability; a string is (k,ℓ)-extractable if the model correctly predicts ℓ of its tokens given a prefix of k tokens. This definition has recently gained much interest because of its relevance to copyright infringement and data protection. Despite being seemingly different, we argue that extractable memorisation is an estimator for counterfactual memorisation. Concretely, let performance γ be measured as whether a string is (k,ℓ)-extractable. Now, assume that a string is not (k,ℓ)-extractable in the absence of training, i.e., $Y_c(x; \infty) = 0$. Given this assumption, we can define an extractable memorisation estimator as:

$$\hat{\tau}_{x,c}^{\text{extr}} = Y_c(x) \quad (15)$$

We formalise this estimator and its statistical estimand in App. B.3.

Extractable memorisation thus implicitly assumes that $Y_c(x; \infty) = 0$. Counterfactual memorisation, on the other hand, controls for this counterfactual. Notably, assuming $Y_c(x; \infty) = 0$ may

be reasonable when a string is long and complex; in this case, the chance that x would be in the model's top-1 beam (the output of greedy decoding) approaches zero. However, this assumption may not be reasonable for shorter or less complex sequences. Rather, it may cause the resulting estimate to conflate memorisation with the intrinsic difficulty of predicting a string.

6 Experiments

While our method applies to any model trained with a single pass on its training data, we focus on quantifying memorisation in pretrained LMs, which are characterised by the use of architectures with a large number of parameters and large datasets. Due to the costs of training such models from scratch, we take advantage of open-source pretrained models whose intermediate checkpoints and preprocessed data are publicly available. In this section, we detail the models and data used and describe how we collect the observed outcomes $Y_c(x)$.

The Pythia Suite. We use the publicly available Pythia model suite[10] (Biderman et al., 2023b), composed of 8 transformers of sizes ranging from 70M to 12B parameters. These models were trained on the Pile dataset (Gao et al., 2020; Biderman et al., 2022), a 300B-token curated collection of English documents. All models are trained using the same data. Specifically, the dataset is shuffled and "packed" into sequences of 2,049 tokens;[11] each of these sequences corresponds to an instance x. Training was performed using a cosine learning rate schedule with warm-up, and using a batch size of 1,024 sequences, resulting in exactly 143k optimisation steps. We use the model versions trained on the deduplicated Pile dataset to reduce the risk of finding spurious memorisation patterns due to duplication. The deduplicated dataset has 207B tokens, thus models using this version are trained for circa 1.5 epochs to keep an equal token count relative to the non-deduplicated versions. We consider the checkpoints relative to the first epoch (i.e., up to step 95k).[12] More details are in App. C.

Constructing the Panel. Ideally, we would collect performance metrics for each instance x and

for every checkpoint step c. However, given the size of the Pile dataset, it is computationally infeasible to collect evaluations for all instances; thus we resort to subsampling this data. Furthermore, the granularity of the available checkpoints (i.e., every 1k timesteps) does not allow us to consider each timestep; thus we consider timesteps $c \in \{0, 1\text{k}..., 95\text{k}\}$ and treatment timesteps $g \in \{1\text{k}, 2\text{k}..., 95\text{k}\}$. To match the checkpoint frequency, we consider all instances between two checkpoints (i.e., 1k batches) as if they were seen by the model at the same timestep. We term these groups of batches **macro-batches**[13] and formally define them as $\mathcal{G}_g = \bigcup_{g-1\text{k}<t\leq g} \mathcal{B}_t$. To obtain enough evaluations for each macro-batch, we sample instances from the training set in two steps: we randomly choose 10 batches for each macro-batch and sample 10 instances from each. This process results in 14.3k analysed training instances. Additionally, we sample 2k instances from the validation set to create $\mathcal{G}_\infty$. This process returns a panel of 16.3k instances evaluated at 96 timesteps.[14] As our performance metric we use the sequence-level log-likelihood: $\gamma(\boldsymbol{\theta}, x) = \log p_{\boldsymbol{\theta}}(x)$.[15]

Statistical Inference. To compute statistical significance, we use the Simple Multiplier Bootstrap procedure of Callaway and Sant'Anna (2021) which returns simultaneous confidence intervals for all memorisation estimates, accounting for dependencies across macro-batches and checkpoint steps and thus avoiding multiple-testing issues.

7 Results

We report the memorisation profiles of all Pythia sizes in Fig. 2. Below, we use these memorisation profiles to describe different types of memorisation.

Instantaneous Memorisation. Instantaneous memorisation estimates (defined in §3 as $\tau_{g,c}$ when $g = c$) are depicted as the diagonal entries in the memorisation profiles in Fig. 2, and are also presented in Fig. 3. From these estimates, we can clearly observe the effect of the treatment step on memorisation: instantaneous memorisation is stronger earlier in training than later. Interestingly (but perhaps unsurprisingly), instantaneous memorisation correlates with the cosine learning rate

---

[10]Both preprocessed data and intermediate checkpoints are publicly available at github.com/EleutherAI/pythia.

[11]Since target tokens are the right-shifted input tokens, to compute the loss on 2,048 tokens the Pythia authors added a token to the context.

[12]For completeness, we report the second half-epoch (steps 96k-143k) analysis in App. D.

[13]In econometrics group of instances that undergo treatment at the same time are typically termed *cohorts*.

[14]Our data and experimental artefacts are publicly available at huggingface.co/collections/pietrolesci/memorisation-profiles.

[15]Results using different metrics are reported in App. D.
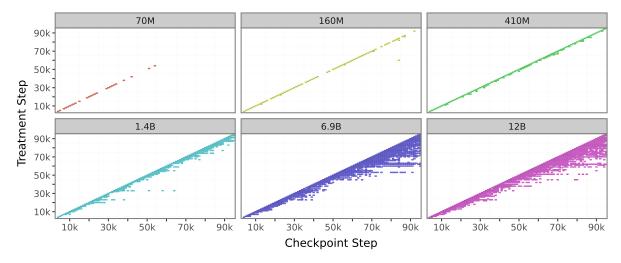
Figure 2: Memorisation profiles ($\tau_{g,c}$). We only show statistically significant entries.
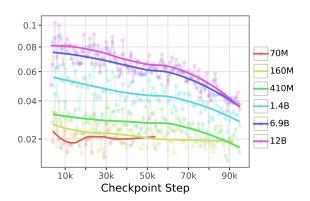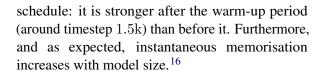

Figure 3: Instantaneous memorisation ($\tau_{g,c}$ for $g = c$). We only show statistically significant estimates.

Figure 4: Average persistent memorisation ($\tau_{g,c}$ averaged per timestep after treatment, i.e., $c - g$). We only show statistically significant estimates.

schedule: it is stronger after the warm-up period (around timestep 1.5k) than before it. Furthermore, and as expected, instantaneous memorisation increases with model size.[16]

Persistent Memorisation. Persistent memorisation estimates (defined in §3 as $\tau_{g,c}$ when $g > c$) are depicted as the off-diagonal entries in the memorisation profiles in Fig. 2. Fig. 4 shows the average persistent memorisation at a specific number of timesteps after treatment; in this figure, $\tau_{g,c}$ were averaged across macro-batches for each $c - g$.[17] This way of aggregating the memorisation profile allows us to summarise the general memorisation patterns of a model. Smaller models have lower per-
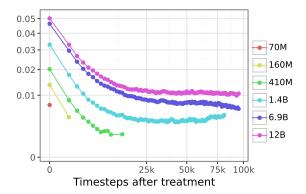
sistent memorisation, with 70M having no persistent memorisation. Interestingly, persistent memorisation plateaus after 25k timesteps. This result has implications for data ordering during training. For example, if there are particular instances that we do not want the model to memorise, but which we still want to use during training, they should be included in earlier batches.[18]

Residual Memorisation. Residual memorisation estimates (defined in §3 as $\tau_{g,c}$ when $c = T$) are depicted as the final-column entries in Fig. 2, and are also presented in Fig. 5 (we consider T to be the end of the first epoch here, i.e., timestep 95k). Interestingly, while all macro-batches undergo some degree of instantaneous memorisation, it appears

---

[16]Notably, we expect instantaneous memorisation to always be present in normally-trained LMs (albeit with a potentially small value). It could thus be used for power analysis (Cohen, 1992): choosing the number of instances to sample per macro-batch which provides sufficient statistical power to correctly detect memorisation.

[17]By averaging across macro-batches, variance is lower and more estimates become statistically significant.

[18]We note that our results differ from Biderman et al.'s (2023b), who find no differences in memorisation due to an instance's treatment step. We hypothesise that this discrepancy stems from the differences in metrics used to quantify memorisation and the statistical approaches adopted.
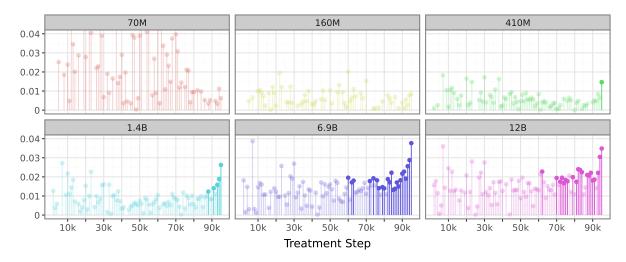
Figure 5: Residual memorisation ($\tau_{g,c}$ for $c = T = 95k$). Stronger colour intensity indicates statistical significance.
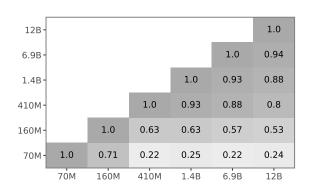

Figure 6: Pearson correlation between the memorisation profile of different models.

that many are then forgotten by the end of the first epoch, as shown by the statistically insignificant residual memorisation estimates. Furthermore, in line with our persistent memorisation results, residual memorisation shows a recency effect: the final macro-batches are the most memorised. We hypothesise that this recency effect can be explained by the learning rate schedule. Specifically, when the learning rate is high, the optimisation process moves model parameters further in the locally optimal direction, thus "overwriting" previous information with new information; this results in higher instantaneous and lower residual memorisation. On the contrary, towards the end of the training process, when the learning rate is lower, previous information is "forgotten" less as the updates are smaller (in expectation), resulting in higher residual and lower instantaneous memorisation.

Memorisation Across Scales. Due to the cost of training large LMs, it is highly desirable to be able to make predictions about a trained model's

characteristics before undertaking training. One strategy is to derive insights from smaller models to inform the design of larger ones (Rae et al., 2021; Black et al., 2022; Le Scao et al., 2022).[19] Predictability across scales is visually apparent in Fig. 3 and 4 where there are similar trends across model sizes. We formalise this intuition in Fig. 6, where we report the Pearson correlation between the memorisation profiles of different models. Interestingly, memorisation for larger models (e.g., 12B) is predictable from smaller ones (e.g., 410M). We note that 70M and 160M are less predictive of the memorisation in 12B. However, prior work has shown that both these models suffer from training instability (Godey et al., 2024); the reduction in predictability with these smaller models might thus be specific to the Pythia suite.

8 Conclusions

The memorisation of training data by neural networks has critical implications for privacy, copyright, and security. Thus, well-founded quantifications of memorisation, and corresponding accurate and efficient methods for their estimation are of great importance. This work presents one such quantification and builds on the econometrics literature to derive an unbiased and efficient estimator of memorisation based on the difference-in-differences design. We use this estimator to study the memorisation profiles of the Pythia model suite and find that memorisation is stronger and more persistent in larger models, determined by data order and learning rate, and stable across model sizes.

---

[19]Scaling laws for other notions of memorisation (§5) have been studied in Biderman et al. (2023a).

Limitations

This work estimates counterfactual memorisation in pretrained LMs. Unfortunately, due to the costs associated with running large pretrained LMs—even in inference mode—we experimented with a limited number of models (the Pythia suite) trained in a single language (English). Investigating whether other model architectures, training procedures, and natural languages result in similar memorisation profiles would be important to strengthen our conclusions. Furthermore, when collecting the panel data needed to estimate memorisation, we subsampled the number of evaluated instances; this can significantly increase our estimators' variance.

Acknowledgements

References

Sotiris Anagnostidis, Gregor Bachmann, Lorenzo Noci, and Thomas Hofmann. 2023. The curious case of benign memorization. In *The Eleventh International Conference on Learning Representations*.

Joshua Angrist and Jörn-Steffen Pischke. 2015. *Mastering 'Metrics: The Path from Cause to Effect*. Princeton University Press.

Devansh Arpit, Stanisław Jastrzundefinedbski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S. Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, and Simon Lacoste-Julien. 2017. A closer look at memorization in deep networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 233–242. JMLR.org.

Gregor Bachmann, Thomas Hofmann, and Aurelien Lucchi. 2022. Generalization through the lens of leave-one-out error. In *International Conference on Learning Representations*.

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

Samyadeep Basu, Phil Pope, and Soheil Feizi. 2020. Influence functions in deep learning are fragile. In *International Conference on Learning Representations*.

Stella Biderman, Kieran Bicheno, and Leo Gao. 2022. Datasheet for the Pile. *arXiv preprint 2201.07311*.

Stella Biderman, Usvsn Prashanth, Lintang Sutawika, Hailey Schoelkopf, Quentin Anthony, Shivanshu Purohit, and Edward Raff. 2023a. Emergent and predictable memorization in large language models. *Advances in Neural Information Processing Systems*, 36:28072–28090.

Stella Biderman, Hailey Schoelkopf, Quentin Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar Van Der Wal. 2023b. Pythia: A suite for analyzing large language models across training and scaling. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23.

Sidney Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, Michael Pieler, Usvsn Sai Prashanth, Shivanshu Purohit, Laria Reynolds, Jonathan Tow, Ben Wang, and Samuel Weinbach. 2022. GPT-NeoX-20B: An open-source autoregressive language model. In *Proceedings of BigScience Episode #5 – Workshop on Challenges & Perspectives in Creating Large Language Models*, pages 95–136, virtual+Dublin. Association for Computational Linguistics.

Brantly Callaway and Pedro H. C. Sant'Anna. 2021. Difference-in-Differences with multiple time periods. *Journal of Econometrics*, 225(2):200–230.

Yuan Cao, Zixiang Chen, Misha Belkin, and Quanquan Gu. 2022. Benign overfitting in two-layer convolutional neural networks. *Advances in Neural Information Processing Systems*, 35:25237–25250.

Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramer, and Chiyuan Zhang. 2023. Quantifying memorization across neural language models. In *The Eleventh International Conference on Learning Representations*.

Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. 2019. The Secret Sharer: Evaluating and testing unintended memorization in neural networks. In *Proceedings of the 28th USENIX Conference on Security Symposium*, SEC'19, pages 267—284, USA. USENIX Association.

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

Tyler A. Chang and Benjamin K. Bergen. 2024. Language model behavior: A comprehensive survey. *Computational Linguistics*, pages 1–58.

Jacob Cohen. 1992. Statistical power analysis. *Current Directions in Psychological Science*, 1(3):98–101.

Stephen R. Cole and Constantine E. Frangakis. 2009. The consistency statement in causal inference: A definition or an assumption? *Epidemiology*, 20(1).

R. Dennis Cook and Sanford Weisberg. 1980. Characterizations of an Empirical Influence Function for Detecting Influential Cases in Regression. *Technometrics*, 22(4):495–508.

Nolan Dey, Gurpreet Gosal, Zhiming, Chen, Hemant Khachane, William Marshall, Ribhu Pathria, Marvin Tom, and Joel Hestness. 2023. Cerebras-GPT: Open compute-optimal language models trained on the cerebras wafer-scale cluster. *arXiv preprint 2304.03208*.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(61):2121–2159.

Vitaly Feldman. 2020. Does learning require memorization? A short tale about a long tail. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2020, pages 954–959, New York, NY, USA. Association for Computing Machinery.

Vitaly Feldman and Chiyuan Zhang. 2020. What neural networks memorize and why: Discovering the long tail via influence estimation. In *Advances in Neural Information Processing Systems*, volume 33, pages 2881–2891. Curran Associates, Inc.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. 2020. The Pile: An 800GB dataset of diverse text for language modeling. *arXiv preprint 2101.00027*.

Nathan Godey, Éric de la Clergerie, and Benoît Sagot. 2024. Why do small language models underperform? Studying language model saturation via the softmax bottleneck. *arXiv preprint 2404.07647*.

Frank R. Hampel. 1974. The influence curve and its role in robust estimation. *Journal of the American Statistical Association*, 69(346):383–393.

Valentin Hartmann, Anshuman Suri, Vincent Bindschaedler, David Evans, Shruti Tople, and Robert West. 2023. SoK: Memorization in general-purpose large language models. *arXiv preprint 2310.18362*.

Paul W. Holland. 1986. Statistics and causal inference. *Journal of the American Statistical Association*, 81(396):945–960.

Hongsheng Hu, Zoran Salcic, Lichao Sun, Gillian Dobbie, Philip S. Yu, and Xuyun Zhang. 2022. Membership inference attacks on machine learning: A survey. *ACM Comput. Surv.*, 54(11s).

Duligur Ibeling and Thomas Icard. 2023. Comparing causal frameworks: Potential outcomes, structural models, graphs, and abstractions. *Advances in Neural Information Processing Systems*, 36:80130–80141.

Andrew Ilyas, Sung Min Park, Logan Engstrom, Guillaume Leclerc, and Aleksander Madry. 2022. Datamodels: Understanding predictions with data and data with predictions. In *Proceedings of the 39th International Conference on Machine Learning*, pages 9525–9587. PMLR.

Guido W. Imbens and Donald B. Rubin. 2015. *Causal Inference for Statistics, Social, and Biomedical Sciences: An Introduction*. Cambridge University Press.

Shotaro Ishihara. 2023. Training data extraction from pre-trained language models: A survey. In *Proceedings of the 3rd Workshop on Trustworthy Natural Language Processing (TrustNLP 2023)*, pages 260–275, Toronto, Canada. Association for Computational Linguistics.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7B. *arXiv preprint 2310.06825*.

Nikhil Kandpal, Eric Wallace, and Colin Raffel. 2022. Deduplicating training data mitigates privacy risks in language models. In *International Conference on Machine Learning*, pages 10697–10707. PMLR.

Douwe Kiela, Max Bartolo, Yixin Nie, Divyansh Kaushik, Atticus Geiger, Zhengxuan Wu, Bertie Vidgen, Grusha Prasad, Amanpreet Singh, Pratik Ringshia, Zhiyi Ma, Tristan Thrush, Sebastian Riedel, Zeerak Waseem, Pontus Stenetorp, Robin Jia, Mohit Bansal, Christopher Potts, and Adina Williams. 2021. Dynabench: Rethinking benchmarking in NLP. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4110–4124, Online. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations*.

Pang Wei Koh and Percy Liang. 2017. Understanding black-box predictions via influence functions. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1885–1894. PMLR.

Teven Le Scao, Thomas Wang, Daniel Hesslow, Stas Bekman, M Saiful Bari, Stella Biderman, Hady Elsahar, Niklas Muennighoff, Jason Phang, Ofir Press, Colin Raffel, Victor Sanh, Sheng Shen, Lintang Sutawika, Jaesung Tae, Zheng Xin Yong, Julien Launay, and Iz Beltagy. 2022. What language model to train if you have one million GPU hours? In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 765–782, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Jooyoung Lee, Thai Le, Jinghui Chen, and Dongwon Lee. 2023. Do language models plagiarize? In *Proceedings of the ACM Web Conference 2023*, WWW '23, page 3637–3647, New York, NY, USA. Association for Computing Machinery.

Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. 2022. Deduplicating training data makes language models better. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8424–8445, Dublin, Ireland. Association for Computational Linguistics.

Jinkun Lin, Anqi Zhang, Mathias Lécuyer, Jinyang Li, Aurojit Panda, and Siddhartha Sen. 2022. Measuring the effect of training data on deep learning predictions via randomized experiments. In *Proceedings of the 39th International Conference on Machine Learning*, pages 13468–13504. PMLR.

Min Lu, Saad Sadiq, Daniel J Feaster, and Hemant Ishwaran. 2018. Estimating individual treatment effect in observational data using random forest methods. *Journal of Computational and Graphical Statistics*, 27(1):209–219.

Michal Lukasik, Vaishnavh Nagarajan, Ankit Singh Rawat, Aditya Krishna Menon, and Sanjiv Kumar. 2024. What do larger image classifiers memorise? *Transactions on Machine Learning Research*.

Pratyush Maini, Michael C. Mozer, Hanie Sedghi, Zachary C. Lipton, J. Zico Kolter, and Chiyuan Zhang. 2023. Can neural network memorization be localized? In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org.

Donald N. McCloskey. 2016. *Counterfactuals*, pages 1–5. Palgrave Macmillan UK, London.

Sung Min Park, Kristian Georgiev, Andrew Ilyas, Guillaume Leclerc, and Aleksander Madry. 2023. TRAK: Attributing model behavior at scale. In *Proceedings of the 40th International Conference on Machine Learning*, pages 27074–27113. PMLR.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Judea Pearl. 2009. *Causality: Models, Reasoning and Inference*, 2nd edition. Cambridge University Press, USA.

Guilherme Penedo, Quentin Malartic, Daniel Hesslow, Ruxandra Cojocaru, Alessandro Cappelli, Hamza Alobeidli, Baptiste Pannier, Ebtesam Almazrouei, and Julien Launay. 2023. The RefinedWeb dataset for Falcon LLM: Outperforming curated corpora with web data, and web data only. *arXiv preprint 2306.01116*.

Jack W Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, et al. 2021. Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint 2112.11446*.

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

Jonathan Roth, Pedro H. C. Sant'Anna, Alyssa Bilinski, and John Poe. 2023. What's trending in difference-in-differences? A synthesis of the recent econometrics literature. *Journal of Econometrics*, 235(2):2218–2244.

Donald B. Rubin. 1974. Estimating causal effects of treatments in randomized and nonrandomized studies. *Journal of Educational Psychology*, 66(5):688–701.

Donald B. Rubin. 2005. Causal inference using potential outcomes. *Journal of the American Statistical Association*, 100(469):322–331.

Andrea Schioppa, Katja Filippova, Ivan Titov, and Polina Zablotskaia. 2023. Theoretical and practical perspectives on what influence functions do. *Advances in Neural Information Processing Systems*, 36:27560–27581.

Jerzy Splawa-Neyman. 1923. On the application of probability theory to agricultural experiments. Essay on principles. *Annals of Agricultural Sciences*, pages 1–51.

Niklas Stoehr, Mitchell Gordon, Chiyuan Zhang, and Owen Lewis. 2024. Localizing paragraph memorization in language models. *arxiv preprint 2403.19851*.

Kushal Tirumala, Aram H. Markosyan, Luke Zettlemoyer, and Armen Aghajanyan. 2022. Memorization without overfitting: Analyzing the training dynamics of large language models. In *Advances in Neural Information Processing Systems*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint 2307.09288*.

Nikhil Vyas, Sham M. Kakade, and Boaz Barak. 2023. On provable copyright protection for generative models. In *International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 35277–35299. PMLR.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Chiyuan Zhang, Daphne Ippolito, Katherine Lee, Matthew Jagielski, Florian Tramèr, and Nicholas Carlini. 2023. Counterfactual memorization in neural language models. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Xiaosen Zheng and Jing Jiang. 2022. An empirical study of memorization in NLP. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6265–6278, Dublin, Ireland. Association for Computational Linguistics.

A   Proofs

A.1   Difference Estimand and Estimator

Lemma 1 (Identification of the Difference Estimand)*. The difference estimand, defined in eq. (6), identifies expected counterfactual memorisation (i.e., the causal estimand in eq. (3)) under Assump. 1.*

Proof. For this proof, we start with the definition of the difference estimand and, via algebraic manipulation, show its equality to expected counterfactual memorisation:

$$\tau_{g,c}^{\texttt{diff}} = \mathbb{E}_{\boldsymbol{x}} \left[Y_c(\boldsymbol{x}; g) \mid G(\boldsymbol{x}) = g \right] - \mathbb{E}_{\boldsymbol{x}} \left[Y_c(\boldsymbol{x}; \infty) \mid G(\boldsymbol{x}) = \infty \right] \tag{16a}$$

$$= \mathbb{E}_{\boldsymbol{x}} \left[Y_c(\boldsymbol{x}; g) \mid G(\boldsymbol{x}) = g \right] - \mathbb{E}_{\boldsymbol{x}} \left[Y_c(\boldsymbol{x}; \infty) \mid G(\boldsymbol{x}) = g \right] \qquad \text{By Assump. 1} \tag{16b}$$

$$= \mathbb{E}_{\boldsymbol{x}} \left[Y_c(\boldsymbol{x}; g) - Y_c(\boldsymbol{x}; \infty) \mid G(\boldsymbol{x}) = g \right] \qquad \text{By linearity of expectations} \tag{16c}$$

$$= \tau_{g,c} \tag{16d}$$

This completes the proof. □

Lemma 2 (Unbiasedness of the Difference Estimator)*. The difference estimator, defined in Estimator 1, is an unbiased estimator of expected counterfactual memorisation $\tau_{g,c}$ under Assump. 1.*

Proof. To prove this estimator is unbiased, let us first define the probability of sampling a batch $\mathcal{B}_g$:

$$p(\mathcal{B}_g) = \prod_{\boldsymbol{x} \in \mathcal{B}_g} p(\boldsymbol{x} \mid G(\boldsymbol{x}) = g) \tag{17}$$

Taking the expectation of our estimator with respect to the batches used for its estimation we see that:

$$\mathbb{E}_{\mathcal{B}_g, \mathcal{B}_\infty} \left[\widehat{\tau}_{g,c}^{\texttt{diff}} \right] = \mathbb{E}_{\mathcal{B}_g, \mathcal{B}_\infty} \left[\overline{Y}_c(g) - \overline{Y}_c(\infty) \right] \tag{18a}$$

$$= \mathbb{E}_{\mathcal{B}_g, \mathcal{B}_\infty} \left[\frac{1}{|\mathcal{B}_g|} \sum_{\boldsymbol{x} \in \mathcal{B}_g} Y_c(\boldsymbol{x}) - \frac{1}{|\mathcal{B}_\infty|} \sum_{\boldsymbol{x} \in \mathcal{B}_\infty} Y_c(\boldsymbol{x}) \right] \tag{18b}$$

$$= \mathbb{E}_{\mathcal{B}_g} \left[\frac{1}{|\mathcal{B}_g|} \sum_{\boldsymbol{x} \in \mathcal{B}_g} Y_c(\boldsymbol{x}) \right] - \mathbb{E}_{\mathcal{B}_\infty} \left[\frac{1}{|\mathcal{B}_\infty|} \sum_{\boldsymbol{x} \in \mathcal{B}_\infty} Y_c(\boldsymbol{x}) \right] \tag{18c}$$

$$= \mathbb{E}_{\boldsymbol{x}} \left[Y_c(\boldsymbol{x}; g) \mid G(\boldsymbol{x}) = g \right] - \mathbb{E}_{\boldsymbol{x}} \left[Y_c(\boldsymbol{x}; \infty) \mid G(\boldsymbol{x}) = \infty \right] \tag{18d}$$

$$= \tau_{g,c}^{\texttt{diff}} \tag{18e}$$

where eq. (18c) follows because the sampling of $\mathcal{B}_g$ and $\mathcal{B}_\infty$ are independent; eq. (18d) holds due to eq. (17) and the unbiasedness of Monte Carlo estimators. We can now invoke Lemma 1, which states that $\tau_{g,c}^{\texttt{diff}}$ identifies $\tau_{g,c}$ under the i.i.d. assumption (Assump. 1). Thus, we have that the expected value of our estimator is equal to $\tau_{g,c}$, finishing the proof. □

A.2   Difference-in-Differences Estimand and Estimator

Lemma 3 (Identification of the Difference-in-Differences Estimand)*. The DiD estimand, defined in eq. (10), identifies expected counterfactual memorisation (i.e., the causal estimand in eq. (3)) under Assumps. 2 and 3 for all $c \geq g$.*

Proof. To prove this lemma, we first note that by the no anticipations assumption (Assump. 3):

$$\mathbb{E}_{\boldsymbol{x}} \left[Y_{g-1}(\boldsymbol{x}; \infty) \mid G(\boldsymbol{x}) = g \right] = \mathbb{E}_{\boldsymbol{x}} \left[Y_{g-1}(\boldsymbol{x}; g) \mid G(\boldsymbol{x}) = g \right] \tag{19}$$

Furthermore, by the parallel trends assumption (Assump. 2) and linearity of expectations:

$$\underset{\boldsymbol{x}}{\mathbb{E}}\left[Y_c(\boldsymbol{x};\infty) - Y_{c'}(\boldsymbol{x};\infty) \mid G(\boldsymbol{x}) = g\right] = \underset{\boldsymbol{x}}{\mathbb{E}}\left[Y_c(\boldsymbol{x};\infty) - Y_{c'}(\boldsymbol{x};\infty) \mid G(\boldsymbol{x}) = \infty\right] \tag{20}$$

$$\implies \underset{\boldsymbol{x}}{\mathbb{E}}\left[Y_c(\boldsymbol{x};\infty) \mid G(\boldsymbol{x}) = g\right] = \underset{\boldsymbol{x}}{\mathbb{E}}\left[Y_{c'}(\boldsymbol{x};\infty) \mid G(\boldsymbol{x}) = g\right] - \underset{\boldsymbol{x}}{\mathbb{E}}\left[Y_c(\boldsymbol{x};\infty) - Y_{c'}(\boldsymbol{x};\infty) \mid G(\boldsymbol{x}) = \infty\right]$$

As in Lemma 1, we now start with the definition of the DiD estimand and, via algebraic manipulation, show its equivalence to expected counterfactual memorisation:

$$\tau_{g,c}^{\texttt{did}}$$

$$= \mathbb{E}\left[Y_c(\boldsymbol{x};g) - Y_{g-1}(\boldsymbol{x};g) \mid G(\boldsymbol{x}) = g\right] - \mathbb{E}\left[Y_c(\boldsymbol{x};\infty) - Y_{g-1}(\boldsymbol{x};\infty) \mid G(\boldsymbol{x}) = \infty\right] \tag{21a}$$

$$= \mathbb{E}\left[Y_c(\boldsymbol{x};g) \mid G(\boldsymbol{x}) = g\right] - \underbrace{\mathbb{E}\left[Y_{g-1}(\boldsymbol{x};g) \mid G(\boldsymbol{x}) = g\right]}_{\text{no anticipation}} - \mathbb{E}\left[Y_c(\boldsymbol{x};\infty) - Y_{g-1}(\boldsymbol{x};\infty) \mid G(\boldsymbol{x}) = \infty\right]$$

$$\tag{21b}$$

$$= \mathbb{E}\left[Y_c(\boldsymbol{x};g) \mid G(\boldsymbol{x}) = g\right] - \underbrace{\mathbb{E}\left[Y_{g-1}(\boldsymbol{x};\infty) \mid G(\boldsymbol{x}) = g\right] - \mathbb{E}\left[Y_c(\boldsymbol{x};\infty) - Y_{g-1}(\boldsymbol{x};\infty) \mid G(\boldsymbol{x}) = \infty\right]}_{\text{parallel trends}}$$

$$\tag{21c}$$

$$= \mathbb{E}\left[Y_c(\boldsymbol{x};g) \mid G(\boldsymbol{x}) = g\right] - \mathbb{E}\left[Y_c(\boldsymbol{x};\infty) \mid G(\boldsymbol{x}) = g\right] \tag{21d}$$

$$= \mathbb{E}\left[Y_c(\boldsymbol{x};g) - Y_c(\boldsymbol{x};\infty) \mid G(\boldsymbol{x}) = g\right] \tag{21e}$$

$$= \tau_{g,c} \tag{21f}$$

where we replace the terms in eqs. (21b) and (21c) using their equivalences given in eq. (19) and eq. (20), respectively. This completes the proof. We note that a similar proof is available in Lemma A.1 in Callaway and Sant'Anna (2021). □

Lemma 4 (Unbiasedness of the Difference-in-Differences Estimator). *The difference-in-differences estimator, defined in Estimator 2, is an unbiased estimator of expected counterfactual memorisation $\tau_{g,c}$ under Assumps. 2 and 3.*

Proof. We can follow the same logic as in Lemma 2 because the same properties hold (i.e., the sampling of $\mathcal{B}_g$ and $\mathcal{B}_\infty$ are independent, the joint probability of a set is the product of the probability of sampling individual instances, and the unbiasedness of Monte Carlo estimators). We then arrive at the following equivalence:

$$\underset{\mathcal{B}_g,\mathcal{B}_\infty}{\mathbb{E}}\left[\widehat{\tau}_{g,c}^{\texttt{did}}\right] = \underset{\mathcal{B}_g,\mathcal{B}_\infty}{\mathbb{E}}\left[\overline{Y}_c(g) - \overline{Y}_{g-1}(g) - \overline{Y}_c(\infty) + \overline{Y}_{g-1}(\infty)\right] \tag{22a}$$

$$= \underset{\mathcal{B}_g,\mathcal{B}_\infty}{\mathbb{E}}\left[\frac{1}{|\mathcal{B}_g|}\sum_{\boldsymbol{x}\in\mathcal{B}_g}\left(Y_c(\boldsymbol{x}) - Y_{g-1}(\boldsymbol{x})\right) - \frac{1}{|\mathcal{B}_\infty|}\sum_{\boldsymbol{x}\in\mathcal{B}_\infty}\left(Y_c(\boldsymbol{x}) - Y_{g-1}(\boldsymbol{x})\right)\right] \tag{22b}$$

$$= \underset{\mathcal{B}_g}{\mathbb{E}}\left[\frac{1}{|\mathcal{B}_g|}\sum_{\boldsymbol{x}\in\mathcal{B}_g}\left(Y_c(\boldsymbol{x}) - Y_{g-1}(\boldsymbol{x})\right)\right] - \underset{\mathcal{B}_\infty}{\mathbb{E}}\left[\frac{1}{|\mathcal{B}_\infty|}\sum_{\boldsymbol{x}\in\mathcal{B}_\infty}\left(Y_c(\boldsymbol{x}) - Y_{g-1}(\boldsymbol{x})\right)\right] \tag{22c}$$

$$= \underset{\boldsymbol{x}}{\mathbb{E}}\left[Y_c(\boldsymbol{x};g) - Y_{g-1}(\boldsymbol{x};g) \mid G(\boldsymbol{x}) = g\right] - \underset{\boldsymbol{x}}{\mathbb{E}}\left[Y_c(\boldsymbol{x};\infty) - Y_{g-1}(\boldsymbol{x};\infty) \mid G(\boldsymbol{x}) = \infty\right] \tag{22d}$$

$$= \tau_{g,c}^{\texttt{did}} \tag{22e}$$

Finally, we invoke Lemma 3 which proves that $\tau_{g,c}^{\texttt{did}}$ identifies $\tau_{g,c}$ under Assumps. 2 and 3. □

A.3 Variances of Estimators

We assume that all potential outcomes $Y_c(\boldsymbol{x}; g)$ have the same variance σ^2. We now first look at the variance of the difference estimator. To this end, let's consider the variance of $\overline{Y}_c(g)$:

$$\text{Var}\left(\overline{Y}_c(g)\right) = \text{Var}\left(\frac{1}{|\mathcal{B}_g|} \sum_{\boldsymbol{x} \in \mathcal{B}_g} Y_c(\boldsymbol{x})\right) = \frac{1}{|\mathcal{B}_g|^2} \sum_{\boldsymbol{x} \in \mathcal{B}_g} \text{Var}\left(Y_c(\boldsymbol{x}; g) \mid G(\boldsymbol{x}) = g\right) = \frac{|\mathcal{B}_g| \sigma^2}{|\mathcal{B}_g|^2} = \frac{\sigma^2}{|\mathcal{B}_g|} \tag{23}$$

This is simply the variance of estimating an expectation using the mean of $|\mathcal{B}_g|$ i.i.d. random variables, each with variance σ^2. We can similarly derive the variance of $\overline{Y}_c(\infty)$. The variance of $\widehat{\tau}_{g,c}^{\texttt{diff}}$ is then:

$$\text{Var}(\widehat{\tau}_{g,c}^{\texttt{diff}}) = \frac{\sigma^2}{|\mathcal{B}_g|} + \frac{\sigma^2}{|\mathcal{B}_\infty|} - 2\,\text{Cov}(\overline{Y}_c(g), \overline{Y}_c(\infty)) \tag{24}$$

Assuming batches $\mathcal{B}_g$ and $\mathcal{B}_\infty$ were drawn independently, then the estimators $\overline{Y}_c(g)$ and $\overline{Y}_c(\infty)$ should also be independent. Thus, $\text{Cov}(\overline{Y}_c(g), \overline{Y}_c(\infty)) = 0$.

We now look at the variance of the difference-in-differences estimator. Let the correlation between $Y_c(\boldsymbol{x}; g)$ and $Y_{g-1}(\boldsymbol{x}; g)$ be ρ. These are, respectively, the potential outcomes of our model on a specific instance $\boldsymbol{x}$ before and after training on it. For shorthand, let $\Delta\overline{Y}_g = \overline{Y}_c(g) - \overline{Y}_{g-1}(g)$ and $\Delta\overline{Y}_\infty = \overline{Y}_c(\infty) - \overline{Y}_{g-1}(\infty)$. We can show that:

$$\text{Var}(\Delta\overline{Y}_g) = \text{Var}\left(\overline{Y}_c(g) - \overline{Y}_{g-1}(g)\right) \tag{25a}$$

$$= \text{Var}\left(\frac{1}{|\mathcal{B}_g|} \sum_{\boldsymbol{x} \in \mathcal{B}_g} \left(Y_c(\boldsymbol{x}; g) - Y_{g-1}(\boldsymbol{x}; g)\right)\right) \tag{25b}$$

$$= \frac{1}{|\mathcal{B}_g|^2} \sum_{\boldsymbol{x} \in \mathcal{B}_g} \left(\sigma^2 + \sigma^2 - 2\,\text{Cov}\left(Y_c(\boldsymbol{x}; g), Y_{g-1}(\boldsymbol{x}; g)\right)\right) \tag{25c}$$

$$= \frac{1}{|\mathcal{B}_g|^2} \sum_{\boldsymbol{x} \in \mathcal{B}_g} (2\sigma^2 - 2\rho\sigma^2) = \frac{2\sigma^2}{|\mathcal{B}_g|}(1 - \rho) \tag{25d}$$

We can derive the variance for $\Delta\overline{Y}_\infty$ in the exact same manner. We thus have that:

$$\text{Var}(\widehat{\tau}_{g,c}^{\texttt{did}}) = \text{Var}(\Delta\overline{Y}_g) + \text{Var}(\Delta\overline{Y}_\infty) - 2\text{Cov}(\Delta\overline{Y}_g, \Delta\overline{Y}_\infty) \tag{26}$$

Note that $\Delta\overline{Y}_g$ and $\Delta\overline{Y}_\infty$ are estimated with independent samples, and thus, $\text{Cov}(\Delta\overline{Y}_g, \Delta\overline{Y}_\infty) = 0$. We can thus rewrite this estimator's variance as:

$$\text{Var}(\widehat{\tau}_{g,c}^{\texttt{did}}) = \frac{2\sigma^2}{|\mathcal{B}_g|}(1 - \rho_g) + \frac{2\sigma^2}{|\mathcal{B}_g|}(1 - \rho_\infty) \tag{27}$$

If we have $\rho_g > 0.5$ and $\rho_\infty > 0.5$, then the variance of $\widehat{\tau}_{g,c}^{\texttt{did}}$ should be lower than that of the $\widehat{\tau}_{g,c}^{\texttt{diff}}$. This is a reasonable assumption since—for fixed timesteps $g - 1$ and c—there should be a strong relationship between a model's performance on an instance before ($g - 1$) and after (c) it has been trained on due to factors such as vocabulary richness and grammatical structure.

B Statistical Estimands and Estimators in Prior Work

In this section, we formalise prior works' estimators of memorisation using our formalisation of counterfactual memorisation.

B.1 Architectural Counterfactual Memorisation

In this section, we describe one potential estimator for architectural counterfactual memorisation $\tau_{\boldsymbol{x},p(\psi)}$ (in Defn. 4). First, we need the following assumption in order to identify the causal estimand for this quantity.

Assumption 4 (Negligible training effect)**.** *In expectation, the effect of having a specific instance in the training set is negligible on any validation instance. That is, for any two instances $\boldsymbol{x}$ and $\boldsymbol{x}'$:*

$$\mathbb{E}_{\boldsymbol{\psi}}\left[Y_T(\boldsymbol{x}';\infty)\,|\,G(\boldsymbol{x})=\infty\right] = \mathbb{E}_{\boldsymbol{\psi}}\left[Y_T(\boldsymbol{x}';\infty)\,|\,G(\boldsymbol{x})\neq\infty\right] \tag{28}$$

Given this assumption, we can identify the following statistical estimand for $\tau_{\boldsymbol{x},p(\psi)}$:

$$\tau_{\boldsymbol{x},p(\psi)}^{\texttt{arch}} = \mathbb{E}_{\boldsymbol{\psi}}\left[Y_T(\boldsymbol{x};G(\boldsymbol{x}))\,|\,G(\boldsymbol{x})\neq\infty\right] - \mathbb{E}_{\boldsymbol{\psi}}\left[Y_T(\boldsymbol{x};\infty)\,|\,G(\boldsymbol{x})=\infty\right] \tag{29}$$

We now define the architectural estimator, associated with this statistical estimand.

Estimator 3. *The **architectural estimator**, defined as:*[20]

$$\widehat{\tau}_{\boldsymbol{x},p(\psi)}^{\texttt{arch}} = \frac{1}{|\Theta_g|}\sum_{\boldsymbol{\theta}\in\Theta_g}Y_T(\boldsymbol{x}) - \frac{1}{|\Theta_\infty|}\sum_{\boldsymbol{\theta}\in\Theta_\infty}Y_T(\boldsymbol{x}) \tag{30}$$

is an unbiased estimator of $\tau_{\boldsymbol{x},p(\psi)}$ under Assump. 4. In this equation, Θ_g and Θ_∞ are sets of model parameters trained independently with or without $\boldsymbol{x}$ in the training set.

Proof. First, we prove that the statistical estimand $\tau_{\boldsymbol{x},p(\psi)}^{\texttt{arch}}$ identifies $\tau_{\boldsymbol{x},p(\psi)}$:

$$\tau_{\boldsymbol{x},p(\psi)}^{\texttt{arch}} = \mathbb{E}_{\boldsymbol{\psi}}\left[Y_T(\boldsymbol{x};G(\boldsymbol{x}))\,|\,G(\boldsymbol{x})\neq\infty\right] - \mathbb{E}_{\boldsymbol{\psi}}\left[Y_T(\boldsymbol{x};\infty)\,|\,G(\boldsymbol{x})=\infty\right] \tag{31a}$$

$$= \mathbb{E}_{\boldsymbol{\psi}}\left[Y_T(\boldsymbol{x};G(\boldsymbol{x}))\,|\,G(\boldsymbol{x})\neq\infty\right] - \mathbb{E}_{\boldsymbol{\psi}}\left[Y_T(\boldsymbol{x};\infty)\,|\,G(\boldsymbol{x})\neq\infty\right] \qquad \text{By Assump. 4} \tag{31b}$$

$$= \mathbb{E}_{\boldsymbol{\psi}}\left[Y_T(\boldsymbol{x};G(\boldsymbol{x})) - Y_T(\boldsymbol{x};\infty)\,|\,G(\boldsymbol{x})\neq\infty\right] \qquad \text{Linearity of expectations} \tag{31c}$$

$$= \tau_{\boldsymbol{x},p(\psi)} \tag{31d}$$

We now prove the estimator above is unbiased:

$$\widehat{\tau}_{\boldsymbol{x},p(\psi)}^{\texttt{arch}} = \mathbb{E}_{\Theta_g,\Theta_\infty}\left[\frac{1}{|\Theta_g|}\sum_{\boldsymbol{\theta}\in\Theta_g}Y_T(\boldsymbol{x}) - \frac{1}{|\Theta_\infty|}\sum_{\boldsymbol{\theta}\in\Theta_\infty}Y_T(\boldsymbol{x})\right] \tag{32a}$$

$$= \mathbb{E}_{\Theta_g}\left[\frac{1}{|\Theta_g|}\sum_{\boldsymbol{\theta}\in\Theta_g}Y_T(\boldsymbol{x})\right] - \mathbb{E}_{\Theta_\infty}\left[\frac{1}{|\Theta_\infty|}\sum_{\boldsymbol{\theta}\in\Theta_\infty}Y_T(\boldsymbol{x})\right] \tag{32b}$$

$$= \mathbb{E}_{\boldsymbol{\psi}}\left[Y_T(\boldsymbol{x};G(\boldsymbol{x}))\,|\,G(\boldsymbol{x})\neq\infty\right] - \mathbb{E}_{\boldsymbol{\psi}}\left[Y_T(\boldsymbol{x};\infty)\,|\,G(\boldsymbol{x})=\infty\right] \tag{32c}$$

$$= \tau_{\boldsymbol{x},p(\psi)}^{\texttt{arch}} \tag{32d}$$

This completes the proof. $\qquad\square$

B.2 Influence Functions

As mentioned in §5.2, influence functions approximate $\boldsymbol{\theta}_{-\boldsymbol{x},T}$ using a first-order Taylor expansion of the training objective around $\boldsymbol{\theta}_T$. This should lead to small errors under the following assumptions: (i) the loss function is strictly convex in $\boldsymbol{\theta}$, (ii) $H_{\boldsymbol{\theta}}$ is a positive-definite matrix, and (iii) the model has converged (Koh and Liang, 2017). We make these assumptions explicit now.

Assumption 5 (Strict Convexity)**.** *The loss function $\mathcal{L}$ is strictly convex with respect to the parameters $\boldsymbol{\theta}$.*

---

[20]We note that prior work has proposed more efficient estimators of the above (Bachmann et al., 2022; Lin et al., 2022; Ilyas et al., 2022; Park et al., 2023). However, these estimators remain computationally expensive for large LMs.

Assumption 6 (Local Optimality). *The parameters $\boldsymbol{\theta}_T$ locally minimise the loss function $\mathcal{L}$, meaning that the $H_{\boldsymbol{\theta}}$ is a positive-definite matrix and that gradient of the loss with respect to the parameters $\boldsymbol{\theta}_T$ is zero.*

Given these assumptions, we can estimate the counterfactual term in $\tau_{\boldsymbol{x},c}$ (in eq. (2)) by computing the performance using the updated parameters $\boldsymbol{\theta}_{-\boldsymbol{x},T}$. As mentioned in the main text, we thus define $Y_{-\boldsymbol{x},T}(\boldsymbol{x}) = \gamma(\boldsymbol{\theta}_{-\boldsymbol{x},T}; \boldsymbol{x})$ and equate $Y_T(\boldsymbol{x}; \infty) = Y_{-\boldsymbol{x},T}(\boldsymbol{x})$.

Estimator 4. *The **influence function estimator**, defined as:*

$$\hat{\tau}_{\boldsymbol{x},T}^{\texttt{infl}} = Y_T(\boldsymbol{x}) - Y_{-\boldsymbol{x},T}(\boldsymbol{x}) \tag{33}$$

is an unbiased estimator of $\tau_{\boldsymbol{x},T}$ under Assumps. 5 and 6.

Proof. See Cook and Weisberg (1980) or Koh and Liang (2017) for derivations of how $Y_{-\boldsymbol{x},T}(\boldsymbol{x})$ approximates the counterfactual $Y_T(\boldsymbol{x}; \infty)$ under the assumptions above. The estimator then follows trivially from replacing $Y_T(\boldsymbol{x}; \infty)$ in eq. (2). □

B.3 Extractable Memorisation

As mentioned in §5.3, extractable memorisation assumes zero-valued counterfactual performances $Y_c(\boldsymbol{x}; \infty) = 0$. We formalise this assumption, and the associated statistical estimand and estimator in this section.

Assumption 7 (Negligible counterfactual). *In the absence of training, performance on a string should be zero: $Y_c(\boldsymbol{x}; \infty) = 0$.*

Given this assumption, we can trivially identify counterfactual memorisation as being equivalent to the statistical estimand: $\tau_{\boldsymbol{x},c}^{\texttt{extr}} = Y_c(\boldsymbol{x}; g)$. We can now define the (k,ℓ)-extractable memorisation estimator under our framework.

Estimator 5. *The (k, ℓ)-**extractable memorisation** estimator, defined as:*

$$\hat{\tau}_{\boldsymbol{x},c}^{\texttt{extr}} = Y_c(\boldsymbol{x}) \tag{34}$$

is an unbiased estimator of $\tau_{\boldsymbol{x},c}$ under Assump. 7.

Proof. This follows trivially from replacing $Y_c(\boldsymbol{x}; \infty)$ with 0 in eq. (2). □

C Implementation Details

We implement all experiments using the PyTorch framework (Paszke et al., 2019). We use the Pythia models as available through the transformers library (Wolf et al., 2020). For a consistent evaluation between scales, we load every model using bfloat16 precision, which is needed for the larger versions. We control randomness using CUDA deterministic operations and seeding the pseudo-random number generators at every level of the stack and for each multi-processing worker. We use the implementation of the Callaway and Sant'Anna (2021) estimator as available in the differences library.[21]

C.1 The Pythia Suite

We use the publicly available Pythia model suite (Biderman et al., 2023b), which was trained on the Pile (Gao et al., 2020; Biderman et al., 2022). Both the preprocessed training data and intermediate checkpoints are publicly available.[22]

Data. The Pile is a 300B-token curated collection of English documents. The deduplicated version of the dataset is obtained by applying a near-deduplication method based on MinHashLSH and has 207B tokens. Before being used for training, the dataset is shuffled, tokenised, and "packed" into sequences of 2,049 tokens with no end-of-document token.[23] By design, each sequence can pack multiple documents and tokens can attend across document boundaries. Noticeably, the packing process implies that the second half-epoch of deduplicated data contains the same documents but not necessarily the same sequences. There does not exist an official validation set for Pythia models. However, we confirmed with the authors that the original Pile validation set has not been used for training.

---

[21]github.com/bernardodionisi/differences.
[22]github.com/EleutherAI/pythia.
[23]github.com/EleutherAI/pythia/issues/123.

Models. The Pythia model suite is composed of 16 models: transformers of 8 different sizes trained on the Pile as-is or deduplicated. All model sizes were trained using a cosine learning rate schedule with warm-up, the same data order, and a batch size of $1{,}024$ sequences, resulting in exactly 143k optimisation steps. The final 48k optimisation steps correspond to the second half-epoch. Thus, we focus on model checkpoints at initialisation (step 0), and after every 1k iterations (steps 1k-95k) resulting in 96 checkpoints evenly spaced throughout training. For completeness, we report the second half-epoch (steps 96k-143k) analysis in App. D. Additionally, log-spaced checkpoints are available for timesteps early in training (timesteps $c \in \{2^i\}_{i=0}^9$). We do not consider them to obtain evaluations at evenly spaced intervals. We use all available model sizes, that is, 70M, 160M, 410M, 1.4B, 6.9B, and 12B, except 2.8B. We exclude 2.8B from the results since we found a potential mismatch between the available checkpoints and the data order used during training.
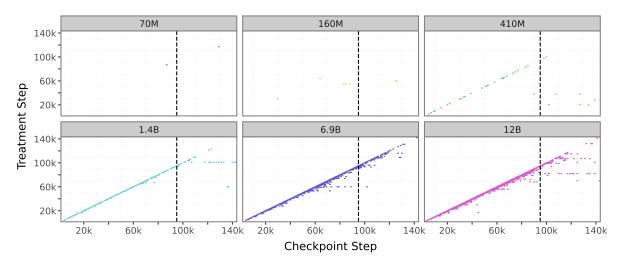
C.2   Hardware Details

We use a server with one NVIDIA A100 80GB PCIe, 32 CPUs, and 32 GB of RAM for all experiments. Below, we report a subset of the output of the lscpu command:

```
Architecture:        x86_64
CPU op-mode(s):      32-bit, 64-bit
Address sizes:       46 bits physical,
                     48 bits virtual
Byte Order:          Little Endian
CPU(s):              32
On-line CPU(s) list: 0-31
Vendor ID:           GenuineIntel
Model name:          Intel(R) Xeon(R)
                     Silver 4210R CPU
                     @ 2.40GHz
CPU family:          6
Model:               85
Thread(s) per core:  1
Core(s) per socket:  1
Socket(s):           8
Stepping:            7
BogoMIPS:            4800.11
```
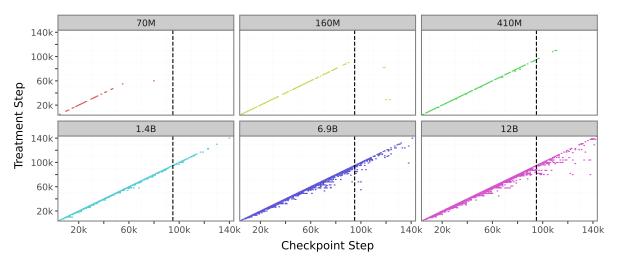
D   Additional Results

On the next page in Fig. 7, we report the memorisation profiles obtained using other metrics besides sequence-level log-likelihood. Specifically, the average token-level accuracy given the true context and the average rank assigned by the model to the correct next token given the true context. We report the results for the entire training process—i.e., using all the available checkpoints: $c \in \{0, 1\text{k}, ..., 143\text{k}\}$ and $g \in \{1\text{k}, ..., 143\text{k}\}$. We present the metrics from most coarse—i.e., average token accuracy (Fig. 7a)—to most fine-grained—i.e., sequence log-likelihood (Fig. 7c).
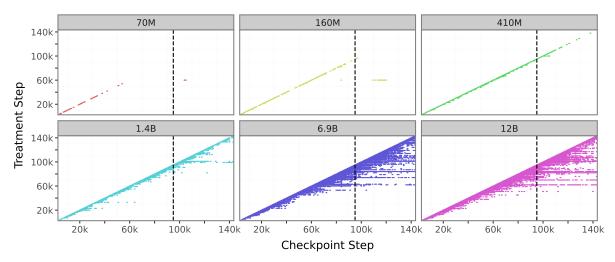
As shown in Fig. 7, different performance metrics result in distinct memorisation estimates. Specifically, finer-grained metrics—like sequence log-likelihood—allow us to detect smaller memorisation effects, and vice versa. For example, average token accuracy, which is the most coarse-grained metric, mostly does not capture instantaneous memorisation for Pythia 410M. Instead, a finer-grained metric—like average token rank or sequence log-likelihood—detects additional effects. Depending on the use-case different metrics might be appropriate. For example, analogously to extractable memorisation (Carlini et al., 2021), average token accuracy could be used to measure memorisation as it matches the specific use-case: detecting whether a model would generate a specific sequence when prompted with some of its tokens. We chose sequence log-likelihood because it allows us to capture more fine-grained memorisation effects beyond the capability of the model to generate a specific sequence. In particular, accuracy captures "hard" transitions in the model's output by determining whether a token is the most likely in the model's output distribution. Log-likelihood, on the other hand, captures more nuanced impacts of training on an instance by assessing whether a token is more likely to be generated than it would be otherwise.

(a) **Average Token Accuracy:**, $\gamma(\boldsymbol{\theta}_c, \boldsymbol{x}) = \frac{1}{|\boldsymbol{x}|} \sum_{i=1}^{|\boldsymbol{x}|} \mathbb{1}(\hat{x}_i = x_i)$, where $\hat{x}_i = \operatorname{argmax}_{x \in \mathcal{V}} p_{\boldsymbol{\theta}_c}(x \mid \boldsymbol{x}_{<i})$ is the predicted token at position i computed using the correct previous tokens as context and $|\boldsymbol{x}|$ is the number of tokens in the sequence.

(b) **Average Rank of the True Token:** $\gamma(\boldsymbol{\theta}_c, \boldsymbol{x}) = \frac{1}{|\boldsymbol{x}|} \sum_{i=1}^{|\boldsymbol{x}|} \operatorname{rank}(x_i)$, where the function $\operatorname{rank}(\cdot)$ returns the rank of the true token at position i computed from the probabilities assigned by the model using the correct previous tokens as context, i.e. $p_{\boldsymbol{\theta}_c}(x \mid \boldsymbol{x}_{<i})$, and $|\boldsymbol{x}|$ is the number of tokens in the sequence.

(c) **Sequence Log-Likelihood:** $\gamma(\boldsymbol{\theta}_c, \boldsymbol{x}) = \log p_{\boldsymbol{\theta}_c}(\boldsymbol{x})$, where $\log p_{\boldsymbol{\theta}_c}(\boldsymbol{x}) = \sum_{i=1}^{|\boldsymbol{x}|} \log p_{\boldsymbol{\theta}_c}(x_i \mid \boldsymbol{x}_{<i})$ and $|\boldsymbol{x}|$ is the number of tokens in the sequence.

Figure 7: Memorisation profiles ($\tau_{g,c}$) computed using different performance metrics γ using all the available checkpoints—i.e., $c \in \{0, 1k, ..., 143k\}$ and $g \in \{1k, ..., 143k\}$. The dashed vertical line indicates the end of the first epoch ($c = 95k$). We only show statistically significant entries.