A Reproducibility

We finetuned 80 models across two different model sizes and 4 different data sets; 40 models were finetuned using Bloom560M and another 40 models were finetuned using Bloom 1.1B. On average, finetuning took approximately 1 hour per model, which makes for 80 GPU hours. Regarding the main experiments, we conducted unlearning using both GA and ICUL. First, for ICUL we ran inference across 3 context length configurations across 80 models and each run took 2 hours on average. This amount to 480 GPU hours. Second, for GA, the situation was very similar. Updating the models using GA across 3 learning rate configurations for all 80 models where each run took approximately 2 hours amounts to another 480 GPU hours. Finally, we ran the additional sensitivity experimnts on SST-2 using Bloom 1.1B. These experiments were conducted for 10 models, using 3 context length configurations and 3 sensitivity setups, where each model run took approximately 1.5 hours, which makes for a total of 135 GPU hours. In total, we used 1175 GPU hours which approximately amounts to 49 GPU days. Note that these numbers include run times to find competitive learning rates and context lengths.

B Additional related works

Here we briefly discuss additional related works that study (supervised) in-context learning theoretically (Xie et al., 2022; Akyürek et al., 2023; Von Oswald et al., 2023; Nagler, 2023; Zhang et al., 2023; Mahankali et al., 2023; Panigrahi et al., 2023; Ahn et al., 2023). Initially, Garg et al. (2022) empirically showed that linear transformers can learn simple function classes like linear regressors in-context. Inspired by these observations, Von Oswald et al. (2023) puts forth a weight construction for trained transformers that implements a single step of gradient descent in a forward pass, which has subsequently been studied in more detail showing that the corresponding weight construction is globally optimal (Zhang et al., 2023; Mahankali et al., 2023; Ahn et al., 2023) and that gradient flow reaches this optimum (Zhang et al., 2023).

C Additional Experimental Results

Both `ICUL` and `GA` have each one crucial parameter, namely the context length and the learning rate. Here we investigate the impact that these parameters have on unlearning success and model performance.

Vary the context length on `ICUL`. For `ICUL`, changing the context length can significantly improve results in terms of unlearning success as seen in Figure 5. In terms of model performance, the situation is more nuanced as can be seen from Figure 6. While the context length has clear impact on forget points, test points seem to be impacted very little by the number of context examples.

Vary the learning rate on `GA`. For `GA`, changing the learning rate can dramatically improve results, where smaller learning rates usually significantly improve results in terms of unlearning success and model performance as shown in Figures 7 and 8.

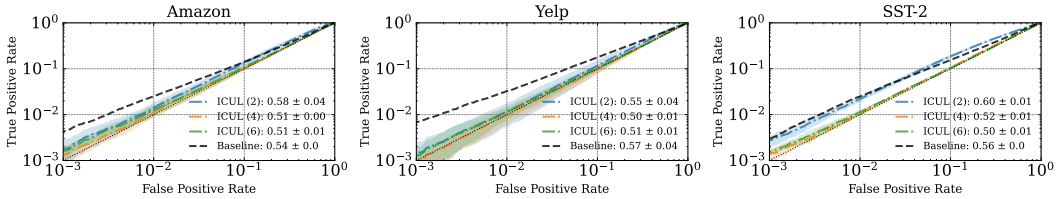
D Details on the Machine Unlearning Evaluation

We term an unlearning algorithm $\mathcal{U}$ successful if an auditor $\mathcal{A}$ cannot tell if information from the forget set S_f still exists within the model after applying the unlearning procedure $\mathcal{U}$, or whether the model was exclusively trained on the retain set $S \backslash S_f$. We refer to the model after the unlearning procedure was applied as the "unlearned model", while we call the model trained on the retain set the "retrained model". Using this motivation, we define unlearning success as follows:

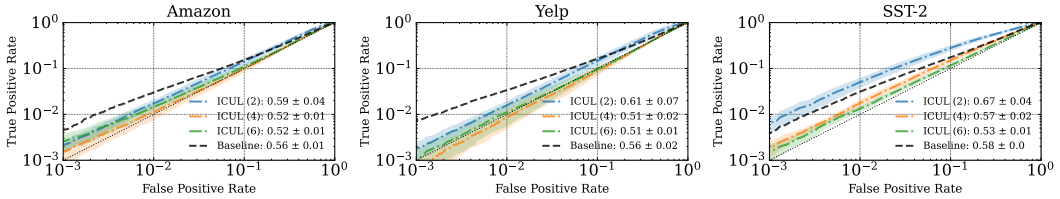
Framing Unlearning Success as a Hypothesis Testing Problem. Given the setup described in the previous section, it is reasonable to employ an hypothesis test to make an educated guess about whether the model after unlearning $\bar{f}$ still contains information from the forget set or not, i.e.,

$$H_0 : S_f \subset S \text{ vs. } H_1 : S_f \not\subset S, \tag{3}$$

where rejecting the null hypothesis corresponds to inferring that the forget set S_f is not part of the model (anymore) whereas failing to reject the null hypothesis means to detect the presence of

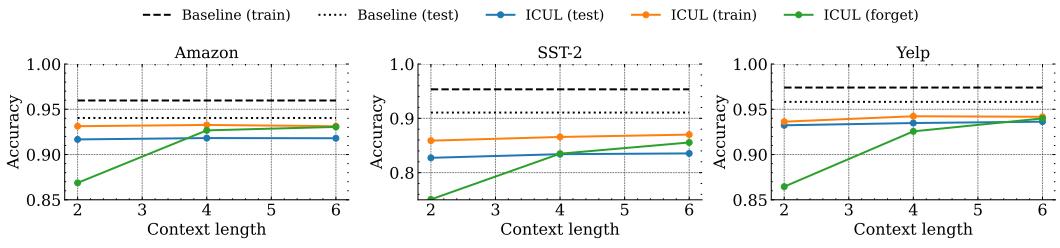


(a) Bloom (560m parameters)

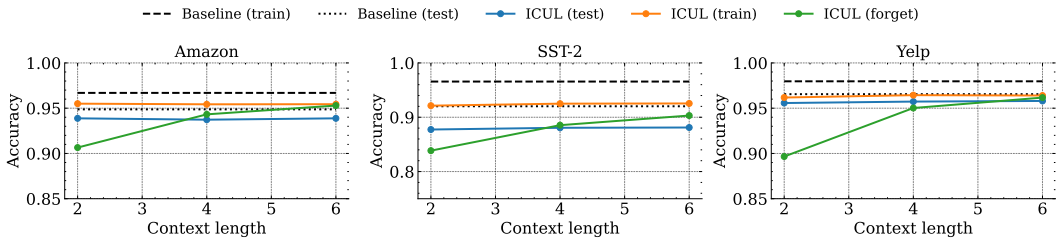


(b) Bloom (1.1b parameters)

Figure 5: **Varying context length for `ICUL`**. Same setup as in Figure 3. We plot the MI attack performance using log scaled ROC curves across different datasets and model sizes. The MI attacks were run against the updated models $\bar{f}$, which was updated using `ICUL`. The closer to the diagonal, which amounts to the adversary randomly guessing whether a forget point is still part of the model or not, the better. The numbers in brackets denote the best parameters and the numbers after that show the AUC ± 1 standard deviation across 10 evaluation runs. The black dashed line represents the baseline performance of not removing the point where the same attack is run on the model $f_{\theta(S)}$, as described in Section 5.1. Shades indicate ± 1 standard deviation across 10 evaluation runs.



(a) Bloom (560M parameters)

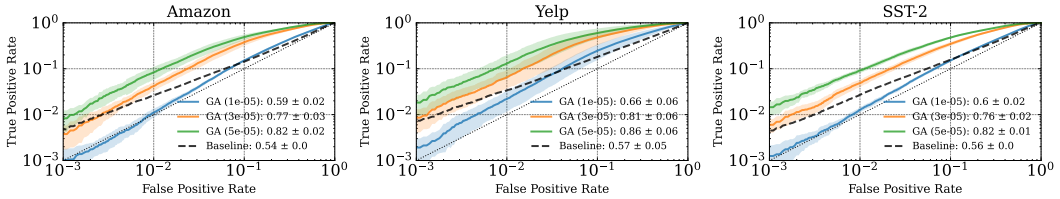


(b) Bloom (1.1B parameters)

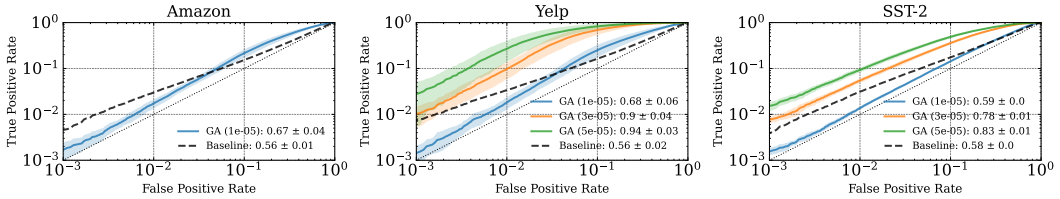
Figure 6: **Classification performance as we vary context length for `ICUL`**. We report classification accuracy on train, forget and test points across all data sets and model sizes. For better readability, ± 1 standard deviation was excluded from the figure.

S_f in the model and thus the unlearning procedure $\mathcal{U}$ did not successfully remove all information of S_f from the model. Taking inspiration from the Neyman-Pearson lemma (Neyman & Pearson, 1933), which asserts that the optimal hypothesis test at a predetermined false-positive rate involves thresholding the likelihood-ratio test Λ, we define the measure of unlearning success by comparing two distributions over models:

$$\Lambda = \frac{p\left(f_{\tilde{\boldsymbol{\theta}}}\right)}{p\left(f_{\boldsymbol{\theta}(S \setminus S_f)}\right)}. \tag{4}$$

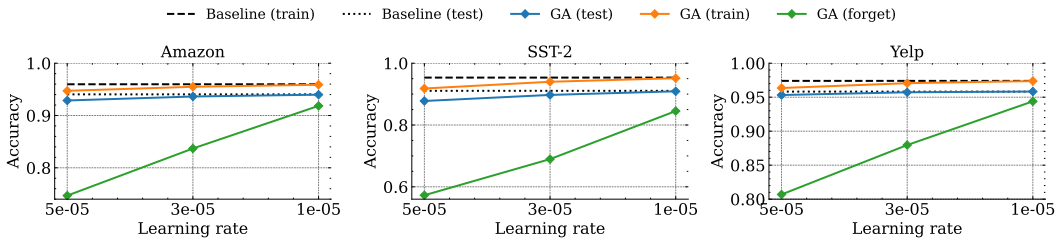


(a) Bloom (560M parameters)

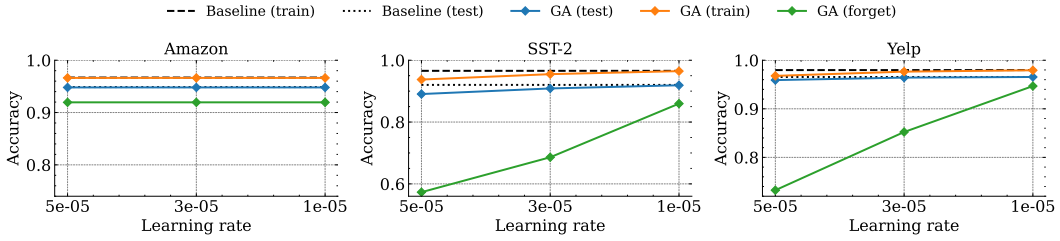


(b) Bloom (1.1B parameters)

Figure 7: **Varying the learning rate for GA.** We plot the MI attack performance using log scaled ROC curves across different datasets and model sizes. The MI attacks were run against the updated models $\bar{f}$, which was updated using GA. The closer to the diagonal, which amounts to the adversary randomly guessing whether a forget point is still part of the model or not, the better. The numbers in brackets denote the best parameters and the numbers after that show the AUC ± 1 standard deviation across 10 evaluation runs. The black dashed line represents the baseline performance of not removing the point where the same attack is run on the model $f_{\theta(S)}$, as described in Section 5.1. Shades indicate ± 1 standard deviation across 10 evaluation runs.



(a) Bloom (560M parameters)



(b) Bloom (1.1B parameters)

Figure 8: **Classification performance as we vary the learning rate for GA**. We report classification accuracy on train, forget and test points across all data sets and model sizes. For better readability, ± 1 standard deviation was excluded from the figure.

In the first scenario, the model undergoes training using a randomly sampled training dataset S containing the forget set S_f. Subsequently, the unlearning procedure $\mathcal{U}$ (approximately) removes the forget set S_f from the trained model. The second scenario involves the model being trained without the inclusion of the forget set S_f. Note that the likelihood ratio test in (4) is not feasible as the distributions over models in the nominator and denominator are usually not analytically known. To make the situation more tractable, we instead define the distribution over losses on points $(\mathbf{x}, y)$

for both models. Then, we replace both distributions from (4) with the tractable test

$$\hat{\Lambda} = \frac{p\big(\ell\big(f_{\hat{\boldsymbol{\theta}}}(\mathbf{x}), y\big)\big)}{p\big(\ell\big(f_{\boldsymbol{\theta}(S\setminus S_f)}(\mathbf{x}), y\big)\big)}, \tag{5}$$

where ℓ denotes an appropriate loss function.

Operationalizing the Likelihood-ratio Audit Operationalizing the likelihood ratio test from (5) requires access to the distribution of losses under the null and alternative hypotheses. While analytical solutions are usually not available, we can readily get large samples from these two distributions. In an ideal scenario, this entails that we would need to fit as many retain models and unlearned models as possible for every forget set of interest. Since this approach becomes computationally too burdensome, we use the following two-step approximation:

Approximating the distributions under H_0 and H_1. Here we adapt the sample splitting procedure first introduced by Carlini et al. (2022) to forget sets with sizes J greater than 1. We train K shadow models on random samples from the data distribution $\mathcal{D}$ so that a fraction p of these models are trained on the forget set $S_f = \{(\mathbf{x}_j, \mathbf{y}_j)\}_{j=1}^{J}$, and a fraction $(1-p)$ are not. In particular, we train shadow models on $K = 10$ subsets of $\mathcal{D}$ so that each forget set $S_f \in D$ appears in $K \cdot p$ subsets. This approach has the advantage that the same K shadow models can be used to estimate the likelihood-ratio test for all the forget sets. Finally, we fit the parameters of two Gaussian distributions to the confidence scores of the retain models and the unlearned models on S_f. Across all experiments, we use $p = 0.5$ and $J = 1$.

Model losses. Instead of using the actual losses, we compute model confidences as $\phi(f(\mathbf{x}), \mathbf{y}) = \log(f(\mathbf{x})_y) - \log(\sum_{y'} f(\mathbf{x})_{y'})$. This score compares the confidence the model assigns to the true class (e.g., 'positive') with the confidences the model assigns to all other classes (i.e., all other words from the approximately 250680 dimensional vocabulary). The higher the score is the more confident the model is in the correct prediction.