DISENTANGLING SEQUENCE MEMORIZATION AND GENERAL CAPABILITY IN LLMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Verbatim memorization in large language models remains a persistent and unsolved challenge, raising critical concerns for privacy, copyright, and responsible deployment. Existing research suggests that effective unlearning requires targeting the specific neurons responsible for memorization, as broad model updates fail to erase content reliably. However, we show that even these approaches rest on a flawed premise. Through controlled experiments, we demonstrate that memorized sequences are not naturally isolated to specific neurons during training, except in cases where the sequences are highly atypical. In this work, we put forward a new training paradigm that attempts to **isolate memorization to specific neurons by design**. The core challenge is that gradients from the repeated sequences entangle both "generalizing" features that improve general capability, in addition to sequence-specific memorization. We show that a simple change to standard training can implicitly disentangle these by leveraging metadata that identifies repeated sequences. We verify the efficacy of our method (`SeqTD`) in a proof-of-concept natural language setting and unveil the mechanism by which this disentanglement is possible through the training dynamics of memorization. We conclude by discussing the practical considerations of the deployment of `SeqTD` and highlight potential avenues for incorporating it into large-scale settings.

1 INTRODUCTION

Large language models are known to memorize sequences that they observe frequently during pretraining (Carlini et al., 2023; Nasr et al., 2023). As a result, it remains possible to extract private information, copyrighted content, and infer the membership of sequences in the training dataset. Due to the legal and ethical risks of these possibilities, significant research has investigated techniques for identifying and removing such memorized sequences (Maini et al., 2024; Patil et al., 2023; Barbulescu & Triantafillou, 2024). Extensive prior research has aimed to identify the parts of a model responsible for memorization and selectively remove them (Chang et al., 2024b; Chen et al., 2024; Bayazit et al., 2024; Guo et al., 2024). These methods rest on a critical assumption: that memorization is confined to specific neurons that play little role in broader language modeling. But does standard training actually produce such neatly isolated memorization neurons? Surprisingly, this fundamental question remains largely unexplored.

In Section 3, we perform a controlled study and find that existing localization methods struggle when memorized sequences are *typical* (linguistically similar to the broader training distribution). Many undesirable cases of memorization fall in this class: copyrighted books and articles generally include broadly applicable linguistic patterns. Our findings challenge the underlying premise of post-hoc localization—in many cases, cleanly isolated memorization neurons may not exist.

If standard pretraining techniques do not induce isolation, are there alternative strategies that promote it?

A potentially "obvious" approach is to route repeated sequences to their own set of neurons, essentially creating memorization neurons by design. In Section 4, we show a critical flaw of this approach: it inhibits learning general linguistic patterns across sequences, undermining the fundamental goal of pretraining. Thus, it appears some neurons must be "shared" (allowed to learn from all sequences) to maximally pick up generalizing patterns. This presents a dilemma: if we

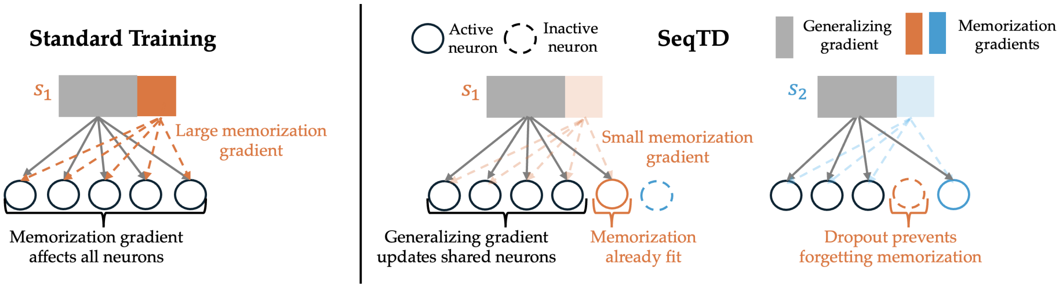


Figure 1: **Conceptual Intution of `SeqTD`**. We conceptually partition the learning signal from each example to into "generalization" and "memorization" components. On the left, we show that standard training can store memorization signal in any neurons. In `SeqTD`, we provide a set of memorization neurons which are shielded from forgetting induced by other examples. As a result, (a) memorization accumulates in these neurons and (b) once these neurons fit the memorized sequence well, memorization is no longer reinforced in shared neurons.

allow shared neurons, they could implement memorization—leading us back to the failure mode of standard training. Could we somehow decompose what a model learns into "generalizing" and "memorizing" components and direct them to different neurons? This is a daunting task; it is difficult to even precisely delineate these components. However, we show that it is indeed possible, by carefully leveraging the training dynamics of memorization.

We introduce *Sequence-Tied Dropout* (`SeqTD`) which splits hidden-layer neurons in MLP layers of transformers into two groups: a pool of "shared" neurons that all examples can update, and a set of "memorization" neurons that each repeated sequence consistently activates (Section 5). By ensuring each repeated sequence drops out *all but a fixed subset* of the memorization neurons, we let memorization accumulate in that subset while shielding shared neurons from repeatedly having memorization reinforced. This design is inspired from Maini et al. (2023), and leverages the learning-and-forgetting cycles of memorization (Toneva et al., 2018): repeated text is systematically "forgotten" in the shared parameters due to interference from other examples, while memorization neurons that only see a small subset of data become stable long-term storage. Crucially, this allows partial parameter sharing so that repeated text can contribute general linguistic signals to the model. In contrast, other strategies for enforcing localization (Chang et al., 2024b) fail to preserve this property (Section 4.1).

On a modified TinyStories pretraining setup, we show that `SeqTD` isolates memorization significantly better than post-hoc localization—at equal or better validation accuracy/perplexity. After training, simply zeroing out the memorization neurons suffices to "unlearn" repeated sequences without noticeably harming the model's performance on other data (Section 5.1). We then investigate the two main practical requirements for applying `SeqTD`: the accuracy of sequence metadata and model size (Section 5.2). We find `SeqTD` is capable of withstanding some amount of noise in sequence metadata (up to 10%) and can isolate memorization across a wide range of model sizes. Finally, we investigate the mechanism by which `SeqTD` isolates memorization and provide experimental evidence of the role of learning-forgetting dynamics in its success (Section 5.3). Ultimately, we present a principled approach for the intricate, yet crucial puzzle of disentangling memorization from the general capabilities of LLMs.

2 RELATED WORKS

Forgetting Memorized Sequences. A large body of work has focused on unlearning or forgetting memorized information from neural models, especially in the classification domain. This includes methods such as SISA (Bourtoule et al., 2021) that exactly unlearn information by maintaining multiple model copies, and a recent influx of approximate unlearning approaches (Triantafillou et al., 2023) that aim to perform post-hoc procedures on a model in order to remove information in question. More recently, with the discovery of memorization of sequences in large language models (Carlini et al., 2023; Nasr et al., 2023), there has been interest in techniques to remove these sequences post-hoc. One class of methods involves further updates of all language model parameters to reduce the likelihood of a memorized sequence Thudi et al. (2022); Liu et al. (2022); Zhang et al. (2024); Yao et al. (2024). Thudi et al. (2022) presents the technique of simply training to increase the loss of memorized examples. Liu et al. (2022) further regularizes this by concurrently minimizing the loss on a *retain set* of validation set examples. Another class of methods focuses on identifying model components responsible for storing memorized sequences and removing them Chang et al. (2024b); Chen et al. (2024); Stoehr et al. (2024); Bayazit et al. (2024); Guo et al. (2024). However, currently all such unlearning methods are prone to degrading general model capabilities, beyond the desired unlearning target (Maini et al., 2023; Zhang et al., 2024). Here, we propose a pretraining technique to better enable removal of memorized content without harming model capability.

Understanding Memorization. There has also been significant interest in understanding the dynamics and mechanisms of sequence memorization. Tirumala et al. (2022); Carlini et al. (2019) showed that sequence memorization in LLMs often occurs before overfitting. Leybzon & Kervadec (2024); Chang et al. (2024a); Toneva et al. (2018) demonstrate that memorization often occurs in cycles of learning and forgetting throughout training. Geva et al. (2021); Dai et al. (2022) study the mechanistic implementation of memorization, finding MLP layers function as key-value memories. Huang et al. (2024) demonstrate that the decoding of memorized sequences may not be causally driven by a single memorization trigger, rather depending partially on certain likely next-token predictions. As a result, they argue that memorization can be highly “entangled” with general capabilities. In Section 3, we extend this finding, showing even when memorization significantly changes the models output (i.e. memorized sequences incur much lower loss than the validation set), identifying the neurons responsible for memorization can be infeasible.

3 PITFALLS OF POST-HOC LOCALIZATION

Prior works have put forward localization as a means to remove memorization. These proposals employ heuristics to measure the contribution of each neuron to memorization and subsequently remove the top scoring ones. This paradigm assumes sequence memorization is sufficiently **isolated** in the neurons. Specifically, memorization must be implemented by some subset of neurons and these neurons must also *not* contribute to the model’s general capabilities. In this section, we study whether standard training naturally satisfies these requirements by examining the performance of two classes of localization methods: pruning and gradient attribution. We defer additional details of both methods to Appendix B.

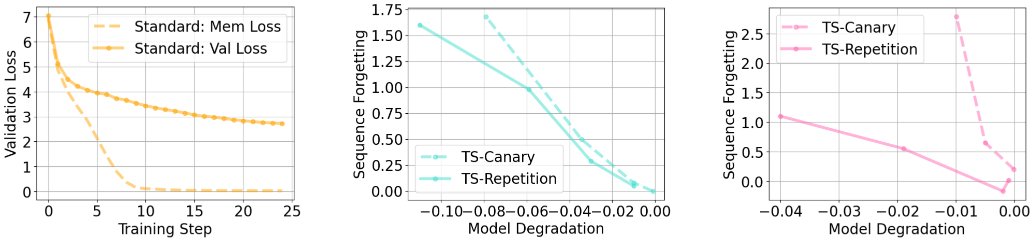
3.1 EXPERIMENTAL SETTING

We train models on two controlled settings designed to induce different types of memorization: highly atypical canaries and sequences that resemble normal test. We then test the ability of the two post-hoc methods to remove these sequences without harming the model.

Datasets. We conduct our experiments in a controlled setting using a subset of the TinyStories dataset (Eldan & Li, 2023). In many real-world settings, memorization arises because sequences are repeatedly seen during training. To simulate this, we randomly sample 100 stories from the TinyStories training set and repeat them 128 times (`TS-Repetition`). Here, memorized sequences are “typical” and come from the same distribution as other training sequences and the validation set. As a comparison, we study a second setting where memorized sequences may be more atypical (`TS-Canary`). We concatenate random sequences of tokens (Canaries) to 100 stories and repeat them 128 times in training. While `TS-Canary` more closely resembles memorization of label noise or atypical examples in supervised settings, it may be less representative of memorization in language models. In both cases, we additionally include 20,000 un-repeated TinyStories sequences.

Evaluation Metrics. We measure **sequence forgetting** as the difference in loss on repeated sequences before and after localization and dropout (higher is better). We measure the **model degradation** as the difference between the validation loss before and after removal (higher is better). This reflects that we hope to avoid increases in validation loss when removing memorization.

We present details of model training and localization in Appendix B.



(a) Learning Curve (b) Pruning (c) Integrated Gradients

Figure 2: **Study of Localization** (a) Loss curve when training on `TS-Repetition`. We observe that memorization decreases alongside the validation loss, indicating that the model gains capability even as it memorizes sequences (b) We plot the unlearning-model degradation tradeoff of pruning by varying the number of dropped out neurons and demonstrate the method struggles to unlearn sequences of both kinds (c) We plot the performance of integrated gradients and demonstrate that although it mitigates model degradation in both cases, it struggles with removing typical sequences.

3.2 EMPIRICAL OBSERVATIONS

We show the results of our analysis in Figure 2. We observe that both post-hoc methods achieve limited success and struggle particularly to remove typical memorized sequences from `TS-Repetition`.

Memorization and Generalization Occur Simultaneously. In Figure 2(a), we plot the validation and memorization of a model trained on `TS-Repetition`. We see that the loss on repeated sequences and the validation set descend simultaneously. Our observations are supported by prior works, such as Tirumala et al. (2022), that observe memorization of sequences occurs prior to overfitting. The simultaneous learning of memorization and generalization illustrates the challenge of avoiding memorization: simply removing repeated sequences can harm model capability.

Localization Methods Achieve Partial Success. In Figure 2(b) we show the trade-off in sequence forgetting and model degradation of pruning. We observe in both settings that dropping out the identified neurons leads to an increase in the memorized sequence's loss, suggesting some success in localization. There are similar trends in Figure 2(c) for integrated gradients, although we observe it generally produces less model degradation than pruning. Additionally, we see that integrated gradients is less effective in removing memorization in `TS-Repetition`, while being highly effective in `TS-Canary`.

Typical Sequence are Difficult to Remove Post-hoc. Across both methods, we find that applying post hoc methods to `TS-Repetition` results in greater model degradation than `TS-Canary`. This difference is particularly pronounced for integrated gradients. Recall that the memorized sequences in `TS-Repetition` are "typical"– similar to the non-repeated training data and the validation set. Our results suggest the memorization of typical sequences may not be isolated in neurons.

In summary, our controlled study suggests that while highly *atypical* memorized sequences appear to be isolated by standard training, the same is not true for more typical sequences. In particular, the model degradation experienced on typical sequences suggests that some neurons contribute both to memorization and general capabilities. Our findings challenge the feasibility of simply removing memorization post-hoc and suggest the need for a new paradigm that explicitly promotes isolation during pre-training.

4 INSUFFICIENCY OF ENFORCING LOCALIZATION

In Section 3, we observed that identifying and removing memorization neurons from a standard pretrained model is not always feasible. This motivates a more direct approach: *enforce* the creation of memorization neurons during pretraining. A simple way to achieve this is to restrict repeated sequences to update a known and separate set of neurons from all other sequences. This would appear to guarantee the existence of memorization neurons by definition: we control exactly which neurons changed as a result of observing a given sequence. For this reason, variants of this technique have been proposed in prior works to guarantee localization (Chang et al., 2024b; Cloud et al., 2024).

However, we find that this rigid approach **both** (a) hinders learning general features across sequences and (b) fails to truly isolate memorization. This reveals that simply "forcing" localization can do more harm than good: memorization can continue to become entangled with general capability, while desirable cross-sequence learning is inhibited.

4.1 GRADIENT MASKING

Implementation. We adopt a similar methodology to Cloud et al. (2024). In each layer we partition the intermediate neurons in the MLP into generalization and memorization neurons. During training, the gradients in the MLP layer from repeated sequences are masked to only modify weights corresponding to memorization neurons and conversely, non-repeated sqeuences have their gradients routed to the generalization neurons. We provide complete experimental details in Appendix C.

4.1.1 EXPERIMENTAL RESULTS

Gradient Masking Hinders Cross-Sequence Learning. We observe that the performance of gradient masking is inferior to a standard model, even before memorization neurons are removed (Figure 3). This observation renders gradient masking impractical, as it significantly worsens the model's general capabilities. This finding suggests that it is essential for some "shared" neurons to be updated by all sequences to aggregate general features.

Gradient Masking Does Not Fully Isolate Memorization. In Figure 3, we see removing memorization neurons further degrades validation performance. This indicates that gradient masking also fails to fully isolate memorization from general capabilities. Even though the dropped out neurons only received "memorization" gradients, the forward pass leaks activations between memorization neurons and the rest of the model. As a result, even general capabilities become sensitive to the removal of memorization neurons during training.

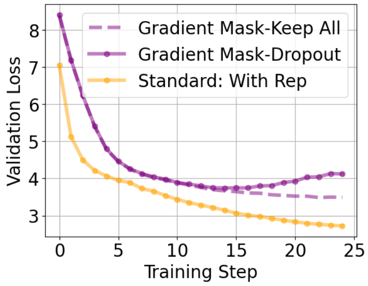


Figure 3: **Impact of Gradient-Masked Training on Validation Performance**. We compare the validation loss of gradient-masked training with (Gradient Mask-Dropout) and without (Gradient Mask-Keepall) memorization neurons removed to a standard training run (Standard: With Rep). We observe that (a) gradient-masked training achieves a significantly worse validation loss and (b) dropping out memorization neurons further degrades validation performance as training progresses.

In summary, shared neurons are necessary to facilitate the learning of general linguistic capabilities across all sequences. Moreover, isolation must go beyond simply forcing memorization to separate neurons: sensitivity to the removal of memorization neurons can develop simply from leakage in the forward pass. Is it possible to simultaneously resolve both challenges? In the next section, we show how carefully leveraging the dynamics of memorization can cause isolation to *naturally arise*, even in the presence of shared neurons.

5 SEQUENCE-TIED DROPOUT (`SeqTD`)

To address the challenge of isolating memorization and generalization signal in LLMs, we propose a novel pretraining strategy for transformers called Sequence-Tied Dropout (`SeqTD`), to simultaneously achieve two goals:

1. **Preserve cross-sequence learning.** Memorized and non-memorized examples still share some parameters, so that memorized examples can contribute linguistic or domain-specific information to the entire model.
2. **Enforce effective isolation.** Memorized examples are consistently routed to a small *pool* of memorization neurons that experience *less forgetting* and thus naturally accumulate memoriza-

tion. Moreover, memorization neurons are dropped out during training, disincentivizing general capabilities from relying on them.

Under standard training, sequence memorization undergoes learning and forgetting cycles (Toneva et al., 2018; Maini et al., 2022). Thus, on each repetition of a memorized sequence, it is reinforced throughout the model. If we could ensure that a set of memorization neurons experienced less forgetting, (a) memorization would accumulate quickly in those neurons and (b) this accumulation would reduce future reinforcement of memorization throughout the model. Since forgetting is driven by interfering gradient updates, we can selectively suppress it in memorization neurons by ensuring that they are updated by a smaller subset of sequences. Crucially, this relies on the same memorization neurons being activated across repetitions of a sequence. To satisfy both requirements, we propose to maintain a pool of memorization neurons and assign each sequence a subset of them to be activated. As a result, any given memorization neuron is deactivated on most other sequences (insulating them from forgetting), while remaining consistent across repetitions of the same sequence. We instantiate this training procedure in the next section, terming it Sequence-Tied Dropout (`SeqTD`).

`SeqTD` extends prior work on localizing memorization (Maini et al., 2023) in three key ways:

1. We position the localization of memorization in the realistic scenario of *typical* sequence memorization (like copyrighted books), as opposed to *atypical* examples (such as mislabeled examples in image classification), which have been a focal point of prior work. Our results in Section 3 have demonstrated that it is significantly harder to localize memorization in the *typical* example scenario, making the goal much more challenging.
2. We make crucial design decisions that allow us to implement localization in the transformer architecture for language modeling task (as opposed to past work in the image classification paradigm). This includes implementing `SeqTD` in synergy with key-value memory stores in the MLP layers of transformers as found by Nanda et al. (2023); Geva et al. (2021).
3. We explain the phenomenon of isolation of memorization by dropout-based regularizers in Section 5.3, which was an open question in the work of Maini et al. (2023).

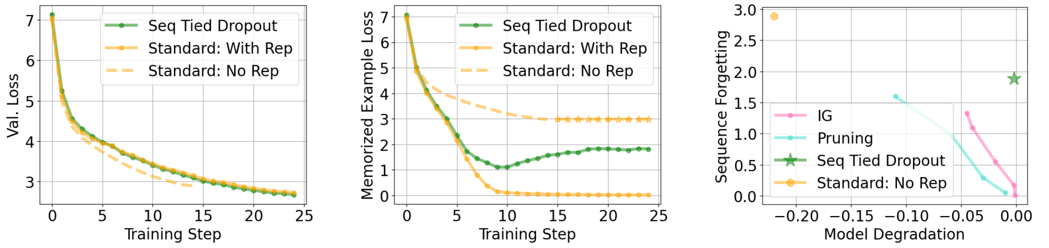
Implementation. We partition the MLP neurons in each layer into shared neurons which are activated across all sequences, and memorization neurons of which only a fraction are activated on any given example (where the fraction is controlled by the memorization neuron dropout ratio p). We assign each sequence in pre-training data a sequence ID and use this as a seed to generate memorization neuron dropout masks. This enables us to ensure the consistency of dropout masks across repetitions of a sequence without precomputing and storing them in advance. We further emphasize that sequence IDs can be arbitrarily assigned (as long as repetitions of a sequence have the same ID). Thus, sequence ID can be generated "on the fly" for example by hashing the sequence.

Experimental Details. We train a GPT Medium model (same as all previous experiments), where 70% of MLP neurons are shared and the remaining 30% are allocated to the pool of memorization neurons. We emphasize that there are *far less* memorization neurons than total sequences. Thus, we do not assume each sequence can be allocated its own memorization neurons. We set the memorization neuron dropout ratio $p = 0.3$, but explore other choices in Section 5.3. We train on the `TS-Repetition` dataset from Section 3.

5.1 EMPIRICAL RESULTS

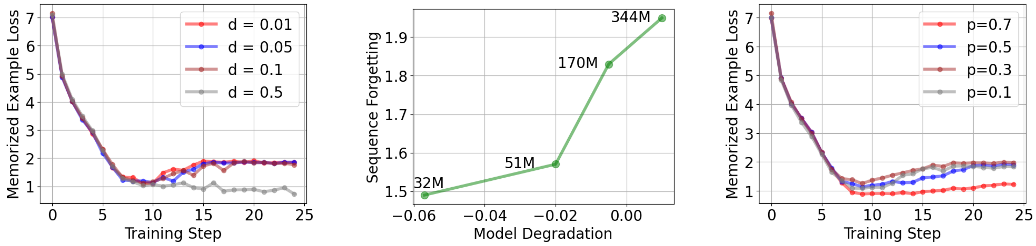
Sequence-Tied Dropout Enables Learning Across Sequences. In Figure 4(a), we compare the validation loss of sequence-tied dropout with standard training with and without repeated documents. Firstly, note that standard training with repeated sequences outperforms filtering them out. This indicates that the model does learn general capabilities from observing documents repeated multiple times in our setting. Next, we compare the standard trained models with *Sequence-Tied Dropout*. We observe that when evaluating without the memorization neurons, sequence-tied dropout achieves comparable validation loss to standard training with repetition.

Dropping Out Memorization Neurons Forgets Memorized Examples. In Figure 4(b), we show the loss on the repeated TinyStories documents. A standard trained model memorizes these sequences during training, achieving close to 0 loss on them. We observe that dropping out the memorization neurons significantly increases the loss on these sequences, increasing the loss to roughly



(a) Validation Loss (b) Loss on Memorized Examples (c) Unlearning-Degradation Trade-off

Figure 4: **Performance of `SeqTD`** (a) We find that `SeqTD` achieves a comparable validation loss to a normally trained model on `TS-Repetition`, outperforming a model trained without repeated sequences. (b) We show the loss of `SeqTD` on the repeated sequences, showing that it memorizes significantly less than a normally trained model. (c) We compare the sequence forgetting-model degradation tradeoff of `SeqTD`, relative to the post-hoc methods tested in Section 3, finding `SeqTD` outperforms both. We compute the model degradation for `SeqTD` and Standard: No Rep as the difference in the validation loss relative to a standard trained model on `TS-Repetition`.



(a) Sequence ID Noise (b) Model Size (c) Mem. Neuron Overlap

Figure 5: **Practicality of `SeqTD`** (a) We study the impact sequence ID noise d, where a fraction of repeated documents have an inconsistent ID. `SeqTD` withstands small amounts of noise (up to 10%) (b) We examine the performance of `SeqTD` across model sizes, where we measure the model degradation as the change in validation loss relative to a standard model and the sequence forgetting as the loss on repeated sequences, we find that `SeqTD` works across scales but smaller models experience greater degradation from dropout (c) We study the impact of the fraction of memorization neurons activated (p) on any given sequence. We find that the model is generally robust to this choice, but activating too many can interfere in the isolation of memorization.

66% of a standard trained model that does not memorize. Interestingly, we observe that in the latter part of training, the loss of sequence-tied dropout on memorized sequences begins to increase. This suggests that while shared neurons may initially implement some memorization, further training *forgets* it. We further examine this finding in Section 5.3.

Sequence-Tied Dropout Enables Superior Sequence Forgetting-Model Degradation Tradeoff. In Figure 4(c), we compare `SeqTD`'s tradeoff between sequence forgetting and model degradation, compared to the post-hoc methods from Section 3. We show that `SeqTD` achieves the best tradeoff relative to post-hoc methods, achieving a higher loss on memorized sequences with significantly lower impact on validation performance. In particular, `SeqTD` achieves the closest sequence forgetting to a model trained without repeated sequences (Standard: No Rep), while significantly outperforming that model in validation loss. This underscores `SeqTD` 's capability to learn from repeated sequences, while simultaneously isolating memorization.

5.2 PRACTICALITY OF `SEQTD`

There are two crucial requirements in deploying `SeqTD`: (a) accurate metadata that identifies repeated sequences and (b) the presence of memorization neurons which activate only on a subset of sequences. In this section, we study the sensitivity of `SeqTD` to these requirements.

Noisy Metadata. `SeqTD` requires every repetition of a sequence to be presented with a consistent ID, so that the same set of memorization neurons are activated across repetitions. This implies accurate metadata which consistently identifies repetitions of sequences. However, in uncurated large-scale pretraining corpora, metadata can often be noisy or incomplete. We test the robustness of `SeqTD` when some repetitions are assigned inconsistent IDs. Concretely, whenever a repeated sequence is encountered, we randomly perturb its ID with probability d. Our results show that `SeqTD` is robust to relatively small values of d, up to 10%. This suggests that having some amount of noise in the sequence IDs is permissible. On the other hand, when sequence IDs are highly inconsistent across repetitions (50% noise), `SeqTD` fails to isolate memorization as effectively. This verifies that consistency of sequence IDs across repetitions is an important factor behind its success.

Impact of Model Size. Another concern is that `SeqTD` could necessitate using significantly larger models. In Figure 5(b), we test the performance of `SeqTD` on a range of model sizes and find that it is capable of isolating memorization across model scales—as indicated by the comparably high losses on repeated sequences (relative to a normally trained model which attains nearly 0 loss). On the other hand, we find that model degradation (the increase in validation loss compared to a standard trained model of the same size) does grow as the model architecture becomes smaller. However, even on smaller models `SeqTD` outperforms post-hoc methods as shown in Figure 4(c). Thus, while model size plays a role in the success of `SeqTD`, the method has benefits in small models as well.

Overlap in Memorization Neurons. As noted previously, it is impractical for each sequence to have neurons allocated exclusively for it. Thus, there will be some level of "sharing" between the neurons activated across sequences and this will result in some forgetting even in memorization neurons. In Figure 5(c), we study the role of this impact by testing the choice of different memorization neuron dropout ratios p. At higher values of p, more neurons activate per sequence, resulting in more overlap. We find that `SeqTD` is robust to some degree of overlap, but levels that are too high can cause isolation to break down. The findings on overlap can also explain the impact of metadata noise and model size. A decreasing model size corresponds to less available memorization neurons (and more overlap). Similarly, metadata noise forces more overlap between the neurons used on the repeated sequences. We examine the role of memorization neuron overlap further in Section 5.3.

5.3 HOW DOES `SEQTD` ISOLATE MEMORIZATION?

In this section, we investigate the mechanisms behind `SeqTD`'s ability to isolate memorization. Recall our hypothesis: having a set of neurons that (a) activate consistently across repetitions of a sequence and (b) activate on only a subset of other sequences would allow sequence-specific memorization to accumulate in these neurons and prevent it from being reinforced in shared neurons. Is this actually how `SeqTD` works?

Testing the Role of Memorization Neurons. We empirically test the hypothesis that memorization neurons in `SeqTD` are shielded from forgetting. For simplicity, we reran TinyStories pretraining setup with a single repeated sequence that is observed every 40 gradient steps. We track the training loss on this sequence for standard training and `SeqTD` (Figure 6). Recall that in `SeqTD`, the training loss on a sequence uses a forward pass with shared neurons and the sequence's assigned memorization neurons activated. Later in training, standard training continues to experience high-amplitude learning/forgetting cycles. `SeqTD`, on the other hand experiences less such fluctuations, maintaining a lower train loss on the repeated sequence. This provides evidence that memorization neurons have a shielding effect from the forgetting dynamics.

Why Can Memorization Neurons Tolerate Overlap? We hypothesized that `SeqTD` insulates memorization neurons from interference and forgetting. However, this insulation is not perfect: as the number of neurons is *much smaller* than the number of sequences, there must be overlap between the memorization neurons assigned to different sequences. In Figure 5(c), we observe that when p is set high (increasing the amount of overlap across sequences), the isolation effects of `SeqTD` do break down. For more moderate values of p, `SeqTD` is fairly robust. We note that it is not necessary to perfectly isolate memorization neurons from interference. Rather we must simply ensure that these neurons experience *relatively* less interference than shared neurons. In Theorem 3, we formalize this argument in a simplified setting, showing that different values of p control the accumulation of memorization in shared versus memorization neurons.

Unlearning in Shared Neurons. In Sections 5.1 and 5.2, the loss on repeated sequences increases later in training. This suggests that some amount of memorization initially takes place in the shared neurons and is progressively "forgotten" later in training. We hypothesize that once the memorization neurons sufficiently"fit" the repeated sequences, additional observations no longer reinforce memorization in the shared neurons. Meanwhile, updates from other sequences remove memorization in the shared neurons, due to standard forgetting dynamics. This contrasts with standard training where any forgetting that occurs between observations of a sequence is reinforced throughout the *entire* model on subsequent encounters.

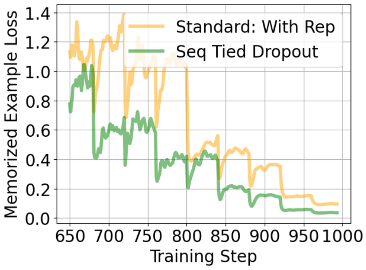


Figure 6: **Learning and Forgetting Dynamics of `SeqTD`.** We study a controlled setting where a specific TinyStories example is inserted every 10 gradient steps and compare the training loss on this sequence for standard training and `SeqTD`. We observe that `SeqTD` experiences lower loss and less forgetting spikes than standard training. This suggests that memorization neurons may provide insulated, long-term storage for repeated sequences.

6 DISCUSSION

Contribution. Our work addresses a significant open problem: Can memorization be disentangled from general model capabilities? In a controlled setting, we demonstrate that standard training can fail to do this—particularly in the practically impactful setting of typical sequences. Our findings extend and support hypotheses in prior work on the "entanglement" of memorization with general capabilities. However, we present a way to naturally promote disentanglement in pre-training by carefully leveraging the learning dynamics of memorization (`SeqTD`). In a small-scale setting, we verify that our method induces the isolation of memorization without compromising the learning of general capabilities. Moreover, we unveil the underlying mechanisms of `SeqTD`, which can inspire future techniques to promote isolation and modularity in LLM pretraining.

Practical Considerations. There are practical considerations on the way to deploying `SeqTD` in real-world settings. Firstly, `SeqTD` can increase the size of model required for learning: it is necessary to have a model large enough to maintain appropriately sized shared and memorization neuron pools. Future work can examine memory efficient ways to implement the memorization neuron pool, for example finding ways to offload the bulk of their parameters to inactive memory by taking advantage of their sparsity. Secondly, `SeqTD` relies crucially on correct metadata to ensure that repeated sequences get routed to the right memorization neurons. Having access to this metadata can be challenging when repetitions of a sequence may be 'embedded" within different surrounding contexts. Thus, future work can examine efficient techniques for generating and correcting metadata annotations based off of the semantic contents of sequences.

Extensions Beyond Memorization. A promising area for future investigation is applying `SeqTD` for cases beyond sequence memorization. For example, `SeqTD` could be applied to promote isolation in the storage of factual knowledge. This could enable more specific editing, as well as more effective test-time control over parametric knowledge. Similarly, extending `SeqTD` to operate on domain metadata could enable robust removal of harmful capabilities downstream–potentially avoiding the pitfalls of fine-tuning based safety-training. Ultimately, we hypothesize that `SeqTD` can be flexibly applied given multiple forms of metadata to promote disentanglement and improve the downstream adaptability of foundation models.

REFERENCES

George-Octavian Barbulescu and Peter Triantafillou. To each (textual sequence) its own: Improving memorized-data unlearning in large language models, 2024. URL `https://arxiv.org/abs/2405.03097`.

Deniz Bayazit, Negar Foroutan, Zeming Chen, Gail Weiss, and Antoine Bosselut. Discovering knowledge-critical subnetworks in pretrained language models, 2024. URL `https://arxiv.org/abs/2310.03084`.

Lucas Bourtoule, Varun Chandrasekaran, Christopher A Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning. In *2021 IEEE Symposium on Security and Privacy (SP)*, pp. 141–159. IEEE, 2021.

Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. The secret sharer: Evaluating and testing unintended memorization in neural networks, 2019. URL `https://arxiv.org/abs/1802.08232`.

Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramer, and Chiyuan Zhang. Quantifying memorization across neural language models, 2023. URL `https://arxiv.org/abs/2202.07646`.

Hoyeon Chang, Jinho Park, Seonghyeon Ye, Sohee Yang, Youngkyung Seo, Du-Seong Chang, and Minjoon Seo. How do large language models acquire factual knowledge during pretraining?, 2024a. URL `https://arxiv.org/abs/2406.11813`.

Ting-Yun Chang, Jesse Thomason, and Robin Jia. Do localization methods actually localize memorized data in llms? a tale of two benchmarks, 2024b. URL `https://arxiv.org/abs/2311.09060`.

Ruizhe Chen, Tianxiang Hu, Yang Feng, and Zuozhu Liu. Learnable privacy neurons localization in language models, 2024. URL `https://arxiv.org/abs/2405.10989`.

Alex Cloud, Jacob Goldman-Wetzler, Evžen Wybitul, Joseph Miller, and Alexander Matt Turner. Gradient routing: Masking gradients to localize computation in neural networks, 2024. URL `https://arxiv.org/abs/2410.04332`.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. Knowledge neurons in pretrained transformers, 2022. URL `https://arxiv.org/abs/2104.08696`.

Ronen Eldan and Yuanzhi Li. Tinystories: How small can language models be and still speak coherent english?, 2023. URL `https://arxiv.org/abs/2305.07759`.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (eds.), *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 5484–5495, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.446. URL `https://aclanthology.org/2021.emnlp-main.446/`.

Phillip Guo, Aaquib Syed, Abhay Sheshadri, Aidan Ewart, and Gintare Karolina Dziugaite. Mechanistic unlearning: Robust knowledge unlearning and editing via mechanistic localization, 2024. URL `https://arxiv.org/abs/2410.12949`.

Jing Huang, Diyi Yang, and Christopher Potts. Demystifying verbatim memorization in large language models, 2024. URL `https://arxiv.org/abs/2407.17817`.

Danny D. Leybzon and Corentin Kervadec. Learning, forgetting, remembering: Insights from tracking LLM memorization during training. In Yonatan Belinkov, Najoung Kim, Jaap Jumelet, Hosein Mohebbi, Aaron Mueller, and Hanjie Chen (eds.), *Proceedings of the 7th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pp. 43–57, Miami, Florida, US, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.blackboxnlp-1.4. URL `https://aclanthology.org/2024.blackboxnlp-1.4/`.

Bo Liu, Qiang Liu, and Peter Stone. Continual learning and private unlearning, 2022. URL `https://arxiv.org/abs/2203.12817`.

Pratyush Maini, Saurabh Garg, Zachary Lipton, and J. Zico Kolter. Characterizing datapoints via second-split forgetting. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 30044–30057. Curran Associates, Inc., 2022. URL `https://proceedings.neurips.cc/paper_files/paper/2022/file/c20447998d6c624b4b97d4466a3bfff5-Paper-Conference.pdf`.

Pratyush Maini, Michael C. Mozer, Hanie Sedghi, Zachary C. Lipton, J. Zico Kolter, and Chiyuan Zhang. Can neural network memorization be localized?, 2023. URL `https://arxiv.org/abs/2307.09542`.

Pratyush Maini, Zhili Feng, Avi Schwarzschild, Zachary C. Lipton, and J. Zico Kolter. Tofu: A task of fictitious unlearning for llms, 2024. URL `https://arxiv.org/abs/2401.06121`.

Neel Nanda, Senthooran Rajamanoharan, Janos Kramar, and Rohin Shah. Fact finding: Attempting to reverse-engineer factual recall on the neuron level, Dec 2023. URL `https://www.alignmentforum.org/posts/iGuwZTHWb6DFY3sKB/fact-finding-attempting-to-reverse-engineer-factual-recall`.

Milad Nasr, Nicholas Carlini, Jonathan Hayase, Matthew Jagielski, A. Feder Cooper, Daphne Ippolito, Christopher A. Choquette-Choo, Eric Wallace, Florian Tramèr, and Katherine Lee. Scalable extraction of training data from (production) language models, 2023. URL `https://arxiv.org/abs/2311.17035`.

Vaidehi Patil, Peter Hase, and Mohit Bansal. Can sensitive information be deleted from llms? objectives for defending against extraction attacks, 2023. URL `https://arxiv.org/abs/2309.17410`.

Niklas Stoehr, Mitchell Gordon, Chiyuan Zhang, and Owen Lewis. Localizing paragraph memorization in language models, 2024. URL `https://arxiv.org/abs/2403.19851`.

Anvith Thudi, Gabriel Deza, Varun Chandrasekaran, and Nicolas Papernot. Unrolling sgd: Understanding factors influencing machine unlearning, 2022. URL `https://arxiv.org/abs/2109.13398`.

Kushal Tirumala, Aram H. Markosyan, Luke Zettlemoyer, and Armen Aghajanyan. Memorization without overfitting: Analyzing the training dynamics of large language models, 2022. URL `https://arxiv.org/abs/2205.10770`.

Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J Gordon. An empirical study of example forgetting during deep neural network learning. *arXiv preprint arXiv:1812.05159*, 2018.

Eleni Triantafillou, Fabian Pedregosa, Jamie Hayes, Peter Kairouz, Isabelle Guyon, Meghdad Kurmanji, Gintare Karolina Dziugaite, Peter Triantafillou, Kairan Zhao, Lisheng Sun Hosoya, Julio C. S. Jacques Junior, Vincent Dumoulin, Ioannis Mitliagkas, Sergio Escalera, Jun Wan, Sohier Dane, Maggie Demkin, and Walter Reade. Neurips 2023 - machine unlearning. `https://kaggle.com/competitions/neurips-2023-machine-unlearning`, 2023. Kaggle.

Yuanshun Yao, Xiaojun Xu, and Yang Liu. Large language model unlearning. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL `https://openreview.net/forum?id=8Dy42ThoNe`.

Ruiqi Zhang, Licong Lin, Yu Bai, and Song Mei. Negative preference optimization: From catastrophic collapse to effective unlearning, 2024. URL `https://arxiv.org/abs/2404.05868`.

A IMPLEMENTATION DETAILS OF TINYSTORIES TRAINING

Implementation and Architecture. We use the nanoGPT library to perform standard pretraining of the models. We train a GPT-2-Medium like architecture with embedding dimension 1024 and a 4 times expansion in the MLP layer. We used 24 layers, the resulting model had approximately 344 M parameters.

Table 1: Hyperparameter Tuning for Standard Training

Parameter	Values
Max Learning Rate	{6e-5,6e-4,6e-3}
Weight Decay	{1e-5,1e-3,1e-1}
Min Learning Rate	$\frac{\text{Max Learning Rate}}{10}$
LR Decay Steps	Total Training Steps

Hyperparameter Tuning. We set the hyperparameters for our training as shown in Table 1. For parameters denoted in sets, we tuned over choices of these parameters relative to the validation loss. We also performed early stopping on the validation loss, but generally found that overfitting did not occur.

B IMPLEMENTATION DETAILS OF POST-HOC LOCALIZATION TECHNIQUES

We examine the following variants We generally follow the methodology used in Chang et al. (2024b) and directly used their code as released online. We restrict our attention to their Hard-Concrete and Integrated Gradients methods presented in the papers.

Hyperparameters: Hard Concrete. We tuned λ, the ℓ_1 loss coefficient used in training the pruning mask M over the values $\{100, 500, 1000\}$ on a tuning set of 5 sequences. Additionally, we tuned the number of pruning iterations in the range $\{1000, 2000, 4000\}$. The remainder of hyperparameters were set to the optimal values reported by Chang et al. (2024a). We tuned relative to the lowest validation loss achieved after dropping out the identified neurons.

Hyperparameters: Integrated Gradients. For Integrated Gradients, the only hyperparameter was the number of IG steps. As a result, we set this to the value reported in the paper, 16.

Dropout Procedure. Following the computation of mask scores by either Hard Concrete or attribution scores by Integrated Gradients, we sorted the neurons in each layer by these scores. Given a dropout parameter r, we dropped out an r proportion of the neurons in each layer, as was performed in Chang et al. (2024a).

C IMPLEMENTATION OF GRADIENT MASKING

We generally follow the implementation outlined in Cloud et al. (2024). We partition each MLP layer into memorization and generalization neurons. We tune this delineation of memorization and generalization neurons by the proportion of generalization neurons g. We additionally partition our dataset into examples seen once and the repeated examples. During training, we mask the gradients in each MLP layer such that the gradients from the repeated examples update only a the memorization block, whereas gradients of all other examples are routed to the generalization block.

Hyperparameter Tuning. We show the hyperparameters tuned for this method in Table 2. Hyperparameter denoted in sets are tuned relative to the validation loss *before* dropping out memorization neurons.

D IMPLEMENTATION OF `SEQTD`

Model Architecture and Implementation. We used the same model architecture as reported in Appendix A. We set the first g fraction of neurons in each MLP as the “shared neurons” and left the

Table 2: Hyperparameter Tuning for Sequence-Tied Dropout

Parameter	Values
Max Learning Rate	{6e-5,6e-4,6e-3}
Weight Decay	{1e-5,1e-3,1e-1}
Min Learning Rate	$\frac{\text{Max Learning Rate}}{10}$
LR Decay Steps	Total Training Steps
g	{0.7,0.9,0.95}

Table 3: Hyperparameter Tuning for Sequence-Tied Dropout

Parameter	Values
Max Learning Rate	{6e-5,6e-4,6e-3}
Weight Decay	{1e-5,1e-3,1e-1}
Min Learning Rate	$\frac{\text{Max Learning Rate}}{10}$
LR Decay Steps	Total Training Steps
g	{0.1,0.3,0.5,0.7}
p	{0.1,0.3,0.5,0.7}

remaining $1 - g$ fraction as the memorization neuron pool. We applied the dropout layer after the GeLU activation function, prior to the downprojection layer.

Assignment of Sequence IDs. We sequentially numbered the sequences in the TinyStories training set and use these indices as the sequence IDs.

Hyperparameter Tuning. In Table 3, we show the hyperparameter ranges tuned over for `SeqTD`. Hyperparameters denoted in sets were tuned over using the validation loss *when the memorization is dropped out*.

E ANALYSIS OF SEQ-TIED DROPOUT

E.1 FORMALIZATION OF TRAINING PROCESS

Architecture. For simplicity, we study the training dynamics of an MLP layer $f(x) = \mathbf{W}_{\text{proj}}\mathbf{W}_{\text{fc}}\mathbf{x}$, where $\mathbf{W}_{\text{proj}} \in \mathbb{R}^{d_{\text{h}} \times d_{\text{emb}}}$, $\mathbf{W}_{\text{fc}} \in \mathbb{R}^{d_{\text{emb}} \times d_{\text{h}}}$. Here, d_{emb} refers to the embedding size of the model and d_{h} refers to the number of hidden neurons in the MLP. Given a sequence $\mathbf{s}$, we consider that f takes in the final position embedding of $\mathbf{s}$, which we denote $\phi(\mathbf{s})$ and directly outputs the logits of the next token (i.e. $\text{softmax}(f(\phi(\mathbf{s})))$ is a probability distribution over the next token in sequence $\mathbf{s}$.

For convenience, we will denote the hidden activations of sequence $\mathbf{s}$ as $\mathbf{z}(\mathbf{s})$. In our analysis, we will assume that the activation space of $\mathbf{z}(\mathbf{s})$ can be split into two subspaces $\mathbf{z}(\mathbf{s}) = [\mathbf{z}(\mathbf{s})_{\text{shared}} \quad \mathbf{z}(\mathbf{s})_{\text{mem}}]$. These components will correspond to our choice of shared and memorization neurons. We will additionally consider $\mathbf{W}_{\text{fc}}$ frozen throughout training and mainly study the training dynamics of $\mathbf{W}_{\text{proj}}$. Thus for convenience, we will also decompose $\mathbf{W}_{\text{fc}}$ into two column-blocks (corresponding to the shared and memorization neurons, respectively): $\mathbf{W}_{\text{proj}} = \left[\mathbf{W}^{\text{shared}}_{\text{proj}} \quad \mathbf{W}^{\text{mem}}_{\text{proj}}\right]$

Data Setup. We will treat our data as (embedding, next token) pairs. We consider we have a repeated sequence $\mathbf{s}^{\text{mem}}$ with corresponding next token $\mathbf{e}^{\text{mem}}$. Next, we will assume we have a large dataset of sequences seen only once during training $\mathcal{D}_{\text{once}} = \{(\mathbf{s}^{(1)}, \mathbf{e}^{(1)}), ..., (\mathbf{s}^{(\text{N})}, \mathbf{e}^{(N)}\})\}$. For simplicity, we will consider the case where $\forall i \ \mathbf{e}^{(i)} \neq \mathbf{e}^{\text{mem}}$. Since we treat $\mathbf{W}_{\text{proj}}$ as frozen, we will also define $\epsilon_{\text{shared}} = \min \mathbf{z}(\mathbf{s}^{\text{mem}})^{\top}_{\text{shared}}\mathbf{z}(\mathbf{s}^{(i)})_{\text{shared}}$ and likewise that $\epsilon_{\text{mem}} = \min_i \mathbf{z}(\mathbf{s}^{\text{mem}})^{\top}_{\text{mem}}\mathbf{z}(\mathbf{s}^{(i)})_{\text{mem}}$. Intuitively, these quantities lower bound how similar the activations in the shared and memorization neurons are between the repeated example and any other example. For simplicity we will assume that the $||\mathbf{z}^{(\mathbf{i})}||_2 = 1$ for all $\mathbf{z}^{(\mathbf{i})}$ and that the parameter $||\mathbf{W}_{\text{proj}}||_2 < \frac{C_{\text{proj}}}{2}$ remains bounded throughout training. Finally we assume that the ouput embeddings e are mutually orthogonal.

Training Process. In standard training, we study the training trajectory (with learning rate γ) of minimizing the cross entropy loss with respect to the parameter $\mathbf{W}_{\text{proj}}$. We consider training with batch size 1.

E.2 FORGETTING UNDER NORMAL TRAINING DYNAMICS

To begin, we introduce a result on the softmax with bounded inputs

Theorem 1 (Softmax on ℓ_∞ bounded vectors)**.** *Consider $x \in \mathbb{R}^d$ and suppose $x_\infty \le C$. Then* $\max_i(\sigma(x))_i \le \frac{e^{2k}}{d-1}$ *and* $\min_i(\sigma(x))_i \ge \frac{e^{-2k}}{d}$

Proof. $\sigma(x)_i = \frac{\exp(x_i)}{\sum\limits_{j\in d}\exp(x_j)} \le \frac{\exp(C)}{\exp(C)+(d-1)\exp(-C)} = \frac{\exp(2C)}{\exp(2C)+(d-1)} \le \frac{\exp(2C)}{d-1}$. Likewise $\sigma(x)_i \ge \frac{\exp(-C)}{\exp(-C)+(d-1)\exp(C)} = \frac{\exp(-2C)}{\exp(-2C)+(d-1)} \ge \frac{\exp(-2C)}{d}$. ☐

Given our assumption that $||\mathbf{z}^{(i)}||_2 = 1$ and the bounded parameter norm assumption $||\mathbf{W}_{\text{proj}}||_2 < \frac{C_{\text{proj}}}{2}$, it follows that $||\mathbf{W}_{\text{proj}}\mathbf{z}^{(i)}||_\infty \le \frac{C_V}{2}$. By Theorem 1, we have that the entries of $\frac{\exp(-C_{\text{proj}})}{d_{\text{emb}}} \le \sigma(f(\mathbf{z}^{(i)}) \le \frac{\exp(C_{\text{proj}})}{d_{\text{emb}}-1}$, element wise. In the remainder of the theory, we denote $c_{min} = \frac{\exp(-C_{\text{proj}})}{d_{\text{emb}}}$ and $c_{max} = \frac{\exp(C_{\text{proj}})}{d_{\text{emb}-1}}$.

We will first show that the memorization of the repeated sequence $\mathbf{s}_{\text{mem}}$ is forgotten when we take intervening steps on non-repeated sequences $\text{x}\mathbf{s}^{(i)}, ..., \mathbf{s}^{(i+n)}$. Formally, we have the following proposition. Formally, suppose that at after step i, we have just seen $\mathbf{s}^{(mem)}$. Then we will show that the logit $\mathbf{e}^{(\text{mem})}$ decreases during subsequent training steps i through $i+n$. For this analysis, we will focus on the dynamics the shared neurons.

Theorem 2 (Forgetting in Standard Training)**.** *Suppose we take a gradient step on $\mathbf{s}^{(mem)}$ at gradient step i and subsequently make gradient updates on non-repeated sequences $\mathbf{s}^{(i)}, ..., \mathbf{s}^{(i+m)}$. After the m gradient steps, we have that $(\mathbf{e}^{mem})^\top f^{(i+m)}(\mathbf{z}^{mem}) \le (\mathbf{e}^{mem})^\top f^{(i)}(\mathbf{z}^{mem}) - \gamma m \epsilon c_{min}$.*

Proof. Only the parameter $\mathbf{W}_{\text{proj}}$ changes throughout training, so we can restrict our attention to its dynamics. We have that the gradient of $\mathbf{W}_{\text{proj}}$ on the sequence-next token pair $(\mathbf{z}, \mathbf{e})$

$$\frac{\partial L}{\partial \mathbf{W}_{\text{proj}}} = (\mathbf{e} - \sigma(f(\mathbf{z}))\mathbf{z}^\top$$

Now let ${\mathbf{W}_{\text{proj}}}^{(i)}$ denote the parameter value of $\mathbf{W}_{\text{proj}}$ after the i-th observation. We have that

$${\mathbf{W}_{\text{proj}}}^{(i+m)} = {\mathbf{W}_{\text{proj}}}^{(i)} + \gamma \sum_{j=1}^{m} (\mathbf{e}^{(j)} - \sigma(f^{(j+i)}(\mathbf{z}^{(\mathbf{i})}))\mathbf{z}^{(\mathbf{i})^\top} \tag{1}$$

where we will denote $f^{(j+i)}$ as the model with parameter $\mathbf{W}_{\text{proj}}$. Then, we have that the logit on the correct next token for memorized example $\mathbf{z}^{\text{mem}}$ is

$$(\mathbf{e}^{\text{mem}})^\top f^{(i+m)}(\mathbf{z}^{\text{mem}}) = (\mathbf{e}^{\text{mem}})^\top f^{(i)}(\mathbf{z}^{\text{mem}}) + (\mathbf{e}^{\text{mem}})^\top \gamma \sum_{j=1}^{m} (\mathbf{e}^{(j)} - (\mathbf{z}^{\text{mem}})\sigma(f^{(j+i)}(\mathbf{z}^{(\mathbf{i})}))\mathbf{z}^{(\mathbf{i})^\top}(\mathbf{z}^{\text{mem}})$$

Now, since we have that the token embeddings are orthogonal, we can rewrite this as

$$(\mathbf{e}^{\text{mem}})^\top f^{(i+m)}(\mathbf{z}^{\text{mem}}) = (\mathbf{e}^{\text{mem}})^\top f^{(i)}(\mathbf{z}^{\text{mem}}) - (\mathbf{e}^{\text{mem}})^\top \gamma \sum_{j=1}^{m} \sigma(f^{(j+i)}(\mathbf{z}^{(\mathbf{i})}))\mathbf{z}^{(\mathbf{i})^\top}(\mathbf{z}^{\text{mem}})$$

Note that by the assumption of bounded norm for $\mathbf{W}_{\text{proj}}$. we have that $(\mathbf{e}^{\text{mem}})^\top \sigma(f^{(j+i)}(\mathbf{z}^{(\mathbf{i})})) \ge c_{min}$ (defined earlier). Note also the assumption that $\mathbf{z}^{(\mathbf{i})^\top}(\mathbf{z}^{\text{mem}}) \ge \epsilon\ \forall i$. This implies that

$$(\mathbf{e}^{\text{mem}})^\top f^{(i+m)}(\mathbf{z}^{\text{mem}}) \le (\mathbf{e}^{\text{mem}})^\top f^{(i)}(\mathbf{z}^{\text{mem}}) - \gamma \sum_{j=1}^{m} \epsilon c_{\min} \tag{2}$$

This immediately yields our desired claim. □

Next, we will show that the seqTD accumulates memorization in the memorization neurons, as formalized in the following theorem. This theorem also crystalizes some key quantities relating to gradient interference. First of all, we see that the forgetting depends on the number of *further gradient steps* taken after seeing $\mathbf{s}^{\text{mem}}$. Secondly, we observe that the impact of forgetting dynamics is influcned by how *similar* the activation of neurons are amongst different examples: controlled by ϵ. The first observation immediately suggests that if some neurons were activated less often, then those neurons would be effectively "store" more memorization.

E.3 ANALYSIS OF `SEQTD`

Theorem 3 (`SeqTD` Accumulates Memorization in Memorization Neurons)**.** *Consider training `SeqTD`, where the memorization neurons are activated on a p fraction of non-repeated examples. We will assume that the model is trained from 0 initialization. Denote the MLP $f_{\textit{mem-dropped}}$ as the model with memorization neurons dropped out and $f_{\textit{gen-dropped}}$ as the model with the generalization neurons dropped out. Suppose that the model is trained for N total steps and the repeated sequence $\mathbf{s}^{mem}$ is observed k times. Then we have at the end of training*

1. $(\mathbf{e}^{mem})^\top f^{(n)}_{\textit{gen-only}}(\phi(\mathbf{s}^{mem})) \leq \gamma k(1 - c_{\textit{min}}) - \gamma(N-k)\epsilon_{\textit{shared}} c_{\textit{min}}$
2. $(\mathbf{e}^{mem})^\top f^{(n)}_{\textit{mem-only}}(\phi(\mathbf{s}^{mem})) \geq \gamma k(1 - c_{max}) - \gamma(N-k)\rho\epsilon_{\textit{mem}} c_{\textit{max}}$

where c_{min} and c_{max} are constants depending on an upper bound of the parameter norm of $\mathbf{W}_{\textit{proj}}$.

Proof. Our argument resembles the proof of Theorem 2, and we will rely on the intuition therein. For reference, we will write the gradients for the components of $\mathbf{W}_{\text{proj}}$ below.

$$\frac{\partial L}{\partial \mathbf{W}^{\text{shared}}_{\text{proj}}} = (\mathbf{e} - \sigma(f(\mathbf{z}))\mathbf{z}^\top_{\text{shared}}$$

and likewise

$$\frac{\partial L}{\partial \mathbf{W}^{\text{mem}}_{\text{proj}}} = (\mathbf{e} - \sigma(f(\mathbf{z}))\mathbf{z}^\top_{\text{mem}}$$

We will first examine $(\mathbf{e}^{\text{mem}})^\top f^{(n)}_{gen-only}(\mathbf{z}^{\text{mem}})$. At any point in training, recall that we can upper and lower bound the value $c_{min} \leq (\mathbf{e}^{(\text{i})})^\top \sigma(f(\mathbf{z}^{\text{mem}})) \leq c_{max}$. As such, observe that $(\mathbf{e}^{(\text{i})})^\top \sigma(f(\mathbf{z}^{\text{mem}}))$ received k updates upper bounded by $\gamma(1 - c_{min})$ (from the k obervations of $\mathbf{z}^{\text{mem}}$ and $(N-k)$ updates upper bounded by $\gamma\epsilon_{\text{shared}} c_{min}$ (from the remaining $(N-k)$ observations of the $\mathbf{z}^{(i)}$. This yields the desired claim for (1).

Now, for claim (2) observe that the component $(\mathbf{e}^{\text{mem}})^\top f^{(n)}_{mem-only}(\mathbf{z}^{\text{mem}})$ receives k updates lower bounded by $(1 - c_{max})$ (again, from the k observations of $\mathbf{z}^{\text{mem}}$, but only $p(N-k)$ updates from other observations, which can likewise be lower bounded by $\gamma\epsilon_{\text{mem}} c_{max}$ This immediately implies the desired claim in (2) □

This theorem formalizes the notion that memorization "accumulates" in the memorization neurons when they are shielded from the interference of other sequences sufficiently. In our theory, the extent to which this occurs is dependent on two quantities (1) the fraction of *non-repeated* sequences for which the memorization neurons are active and (2) the similarity of activations of the repeated example and non-repeated example in the memorization neurons. Relative to algorithm design, however, we will generally only have control over ρ and so we will consider $\epsilon_{\text{shared}} = \epsilon_{\text{mem}}$ out of convenience. Our analysis demonstrates that when ρ is set appropriately low. Some calculation demonstrates that when $\rho < \frac{c_{min}}{c_{max}} - \frac{k}{(N-k)}(c_{max} - c_{min})$, then we will have a seperation in the logits of $\mathbf{s}^{\text{mem}}$ where the memorization neurons primarily contain the memorized example.