Language Model Memory and Memory Models for Language

Benjamin L. Badger
IBM
ben.badger@ibm.com

Abstract

The ability of machine learning models to store input information in hidden layer embeddings, a form of model 'memory', is widely employed but not well characterized. We find that single causal language model embeddings typically contain relatively little input information regardless of data and compute scales during training. In contrast, embeddings from autoencoders trained for input regeneration are capable of nearly perfect memory formation. The substitution of memory embeddings for token sequences leads to computational efficiencies, motivating the introduction of a parallelizable encoder-decoder memory model architecture. Upon causal training these models contain information-poor embeddings incapable of arbitrary information access, but by combining causal and information retention objective functions they learn to form and decode information-rich embeddings. Training can be further streamlined by freezing a high fidelity encoder followed by a curriculum training approach where decoders first learn to process memories and then learn to additionally predict next tokens. We conclude that next token prediction training alone is poorly suited for accurate memory formation, motivating the use of combined objective functions for models where the entire input is not exposed.

1 Introduction

A fundamental feature of machine learning models is their ability to store and use information. The nature of this information storage and processing varies depending on the model in question, but for transformer-based large language models information is stored in the model's parameters as well as the activations of hidden layers. With some oversimplification, one can think of the information attained during training as existing in the model's weights and the information attained from any given input as existing in the model's activations, and we focus on the latter in this work. Here we restrict the notion of a model's 'memory' to the information contained in the last hidden layer embedding of the last token of an input sequence, as this embedding is directly responsible for language tasks such as token prediction and retrieval.

The substitution of memory embeddings for sequences of tokens gives clear benefits to many aspects of inference computation, and has been attempted in various forms because of this. To date memory models have shown promise for single-lookup (phone book) style tasks [Bulatov et al., 2022] and associative retrieval [Rodkin et al., 2025] and variable-length compression [Dai et al., 2019], but to our knowledge have not been widely adopted for general language tasks. In this work we provide a basis for why memory models have underperformed their full-context counterparts, namely that in their current state they are incapable of the arbitrary information access characteristic of full-context Transformers, which we argue is a necessary element in their emergent abilities for tasks they have not been explicitly trained to fulfill (see [Radford et al., 2019]).

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

1.1 Related Work

An investigation into model memory that is particularly relevant is [Bulatov et al., 2022], where the authors incorporate one or more embeddings from a causal decoder into that decoder recurrently. Somewhat analogously, [Dai et al., 2019] uses stop-gradient transformations to allow multiple hidden layer embeddings from past tokens to inform a model's prediction of future tokens. [Lin et al., 2025] optimizes retrieval-augmented generative search by augmenting causal decoder inputs with memory embeddings. [Sukhbaatar et al., 2021] investigate the creation and destruction of memories for very long-context modeling with fixed memory usage.

Contemporaneously with this work, [Prakash et al., 2025] introduced an architecture nearly identical to one presented here (the non-frozen encoder version of the Transformer memory model) and explore the benefits of training this model at somewhat larger compute and data scales. In that work the authors detail the inference time compute benefits inherent in this type of parallelizable memory model in terms of wall clock latency and memory and introduce an optimized decoding cache. Here we make two contributions to this architecture: firstly we find that these models when trained for token prediction actually store very little information in their memory embeddings (Section 6.1) and introduce a new objective function to ameliorate this, and secondly we introduce a frozen encoder variant of this architecture that yields more efficient training computation, particularly for long context inputs (Section 5.2).

LLMs have been found to be invertible in certain scenarios: [Morris et al., 2023] observed accurate conversational prompt input regeneration using last token last hidden layers or even logits. [Weller et al., 2025] found that single-embedding retrieval models far underperform both their theoretical potential as well as multi-embedding or classic search algorithms when matching a query to lists, and we provide an explanation for this observation: retrieval of one element in a list requires near-lossless compression of those list elements into the embeddings, but most causal and retrieval embeddings do not form lossless compressions.

2 Our Contribution

This work seeks to address the following question: *How accurate are causal language model memories, or in other words how much input information do these models retain?* and we answer this as "not very much" for memories composed of single embeddings. The need to access arbitrary input elements for modern language tasks makes such models unsuitable for providing memory embeddings to other models, which combined with the observation that autoencoders are capable of highly accurate memory formation motivates the following question: *Can models be trained to form accurate memories, and can causal decoders learn to use information from perfect-memory encoders?* where we find that the answer is "yes".

We introduce the following methods and approaches: 1) Information extraction methods via inversion for two architectures, 2) An information quantification approach for cross-model comparison, 3) Parallelizable memory model architectures with combined objective functions for accurate memory, and 4) Frozen-memory architectures for arbitrary information retention via curriculum training.

3 Memory and Useful Memory

The most common application of deep learning language models today is autoregressive generation, where a sequence of tokens is fed to the model and tokens are predicted one at a time. In order to predict any single next token it may generally be assumed that only a small fraction of previous tokens are required, although it cannot be determined *a priori* which tokens these will be due to the inherent freedom in language such that *any* previous token's information may be requested to predict a next token. A prediction step for these models may be described as a function $f : X \to Y$ where $X \in \mathbb{Z}^n$ is the sequence of n input tokens and $Y \in \mathbb{Z}^1$ is the single output token, where f is by many-to-one function. Many-to-one functions are typically non-invertible and as such offer no guarantees of how much input information will be stored in any intermediate state. In our parlance, there is no guarantee that causal models will have good internal memory because they are exposed to the entire input (or at least a token-by-token cache of the input) and are optimized on an objective of predicting a single next token at a time, which normally requires the knowledge of only a handful of

previous tokens. Therefore it can be argued that causal models are unlikely to form high-information embeddings, as they would not be expected to need to do so to minimize training loss.

Understanding memory informational content is important when models are not exposed to the entire input, if for example a model were to use an embedding in place of a sequence of tokens, which we call a 'Memory' Model. Memory model inference is made more difficult by the need to be able to extract any of the relevant information in memory embeddings, which leads to the requirement that embeddings contain *all* input information because which tokens will and will not be used in the future is unpredictable. The benefits of such models are reduced computation at inference, explored in the next section.

3.1 Computational Complexity

The design of effective memory models is motivated by computational efficiencies at inference: for a given sequence, memory models requires less cache memory and computation, leading to lower time-to-first-token, large context window capability, and higher token throughput than full-sequence models. For Transformer memory models for n tokens the computation for full-context models is proportional to n^2, whereas for a memory model embedding with s sequences ('chunks') of n/s tokens each the computation is proportional to $s(n/s)^2$ for the encoder and s^2 for the decoder. Upon solving for the minimum value of the maximum of these functions, the ideal value of s in terms of n is $s = n^{2/3}$ which results in computation proportional to $n^{4/3}$. We investigate memory models in which encoder forward passes may be parallelized during both training and inference, which for full parallelization reduces the encoder's computation to be proportional to $n^{2/3} = s$ and does not change the decoder's complexity. In the case where global memory bandwidth is limiting, memory models offer another substantial speedup: rather than loading $n * d$ elements per layer per token predicted, $n^{2/3} * d$ elements are loaded for each token predicted assuming $s = n^{2/3}$. For this work we typically use $s < n^{2/3}$, specifically $s = 4$ for $n = 1024$ or $n = 2048$, raising the computational complexity but lowering the number of cache elements loaded.

3.2 Theory: Model Invertibility and Information Retention

What does it mean for a model's memory of an input to be good or bad? We assume that a good memory is one that is accurate, in that it can be used to accurately reconstruct that input, or equivalently that a good memory contains all the information necessary to uniquely identify any given input, and for information-rich inputs such as language such memories are necessarily also information-rich. Also equivalently, a good memory is one that allows the model to be inverted.

Invertibility in the context of deep learning models is a concept subtle enough to require further exploration. Given a model θ and input X we obtain an output embedding Y as $Y = O(X, \theta)$, where Y can then be used for token prediction or retrieval. Strictly speaking, model θ is invertible if and only if that equality may be represented by a function $f : X \to Y$ that is bijective, that is, both injective (one-to-one) and surjective (onto). Language model architectures are typically not invertible in this strict sense because they are composed of functions that are themselves not invertible (specifically non-injective): for example, not a single linear layer, normalization, or attention module in the original Transformer [Vaswani et al., 2017] is invertible. None of the models or architectures presented in this work are invertible to the degree that unrecognizable inputs may be generated that are, to the model, indistinguishable from data samples [Badger, 2025].

On the other hand, suppose X is a non-infinite set such as a collection of dataset samples, and Y the non-infinite set of model embeddings of these inputs: in this case, the model is invertible with respect to the dataset if and only if it is representable by a function $f : X_n \to Y_n$ that maps each input to a unique output, which we index by $n \in \{0, 1, ..., N\}$ for dataset size N. Testing for invertibility over a finite dataset requires only that the number of unique outputs equals the number of unique inputs, and over practically all dataset and model combinations used today this is guaranteed to be true due in part to the dimensionality of the outputs and the non-linear nature of θ (for empirical evidence of this on image data, see [Badger, 2023]). In this input-restricted sense, language models are all invertible.

These considerations motivate a third definition of model invertibility: an invertible model is one in which for a given set of inputs, each model output may be mapped to the corresponding input in that set with a mapping that is computationally feasible to find and perform. If we allow the inverse function to be trainable and assuming the the inverse function mirrors the model in some

way, finding an inversion function will likely be much more difficult than actually computing it. Equivalently, a model with a good memory is one where some method can be feasibly found that can accurately reconstruct any input on a given corpus given the memory embedding. Computational feasibility being subjective, we define it to be around 96 V100 hours (or around 24 H100 hours) which is a maximum theoretical total of around $170 * 10^{18}$ FLOP, and more training details are given in section A.1. We perform experiments with Transformers as well as Masked Mixers [Badger, 2025], Transformer-like models that substitute attention for masked convolutions, in order to test whether any given observation is specific to attention or not.

4 Language Model Memory Measurement

Prior work found that the aforementioned architectures exhibited different degrees of invertibility as judged by a model-free decoding approach [Badger, 2025]. Like other inversion approaches that don't involve training a decoder model, that method was found to be generally incapable of accurately inverting single embeddings. We therefore replace gradient descent with a trainable decoder model, inspired by [Morris et al., 2023]. We first tested language model inversion for relatively small models over the same dataset that these models were trained on (the FineWeb dataset, FineWeb-edu [Penedo et al., 2024], which is itself filtered from Common Crawl snapshots). The experimental approach we take is as follows: first a model is trained for a given language task, then its parameters are frozen and a trainable decoder is added, the decoder is trained to invert the model's embeddings over its training dataset, and a measurement of this inversion in terms of information retention is obtained. In order to apply this technique to models of different types, we use two metrics that yield tokenizer size-independent estimates: an entropy ratio that measures the fraction of information of the input the embedding contains, and a token identification accuracy measurement.

The rationale behind the use of the entropy ratio H_r is that cross-entropy loss is equivalent to the number of bits per token required to convert the models' distribution to the target, where the model's output is a sequence of vectors of size $|t|$, the size of the tokenizer. We now define upper and lower bounds for this metric: the upper bound is simply zero as a perfect model requires no change to its output distribution to match the target, and the lower bound we assume to be the average of the cross entropy between an informationless output (the uniform distribution $\mathcal{U}(|t|)$, which has maximum entropy) and sample t from any target distribution. We then take the complement of the ratio of achieved loss to upper bound loss (Equation 1). The lower bound may also be thought of as normalizing for the differences between tokenizers of various sizes, as it is clearly more difficult to guess a token from a very large tokenizer than one with only a few possible tokens. For the 8k-size tokenizer most commonly used in this work, Equation 1 simplifies to $H_r = 1 - H(p,q)/9.03$.

It should be noted that H_r is a relatively forgiving metric as it only measures a ratio relative to an informationless *output* distribution, not an informationless model *input* distribution. If an inversion decoder is trained to predict FineWeb tokens given input vectors $a_i : i \in 0, 1, ..., n_{ctx}$ each is assigned a random normal distribution, $a_i \sim \mathcal{N}(0, I_d)$, for the same tokenizer we find inversion loss $\mathbb{L} = 7.33$ as the decoder is able to form a statistical expectation of token distributions without actually receiving any input information. A more strict definition of entropy ratio would take this value to be the upper limit and define $H_r = 1 - H(p,q)/7.33$, which for Table 1 results in each entropy ratio measurement approximately halving (in particular, causal Transformer in this case exhibits $H_r = 0.152$).

$$H_r = 1 - \frac{H(p,q)}{H(p_0,q)} = 1 - \frac{H(p,q)}{\frac{1}{n}\sum_n \mathbb{L}\left(\mathcal{U}(|t|), t\right)} = 1 - \frac{-\sum_x q(x)\log(p(x))}{-\sum_x q_0(x)\log(p(x))} \tag{1}$$

We also use a Hamming metric-based measurement of proportion of input tokens that the decoder has successfully inverted, which is the complement of the fraction of non-pad tokens that are correctly identified, and formally given in Equation 2 for j non-pad tokens. In this work we report $h(x,y)$ as 'Accuracy' in terms of % tokens correct for disambiguation. An informationless input ($a_i \sim \mathcal{N}(0, I_d)$ as before) results in an accuracy of $h(x,y) = 3.41$, which is in a sense the lower bound for token recovery accuracy for this dataset and tokenizer combination.

$$h(x, y) = 1 - \frac{1}{j}\text{Card}\left(\{x_i \neq y_i\}\right) : x_i \neq t_{pad}, i \in \{0, 1, ..., j\} \tag{2}$$

4.1 Information Retention Measurement

We test the amount of information contained in a model's last token last hidden layer as follows: given an arbitrary model M, we first discard classification or language modeling heads such that it becomes an encoder mapping input tokens to a single (last token) embedding, $\mathcal{E} : \mathbb{Z}^n \to \mathbb{R}^d$. We then freeze the parameters of $\mathcal{E}$ and add a decoder $\mathcal{D} : \mathbb{R}^{n \times d} \to \mathbb{Z}^n$ that takes as input he encoder's last hidden layer, last token embedding $\mathcal{E}(X, \theta)_{[:,-1]}$. The encoder's embedding is unrolled via a trainable projection which maps $P : \mathbb{R}^d \to \mathbb{R}^{n \times d}$ after [Morris et al., 2023]. The unrolling process is described in more detail in [Badger and Neligeorge, 2025] and essentially amounts to applying a trainable linear projection from a sliding (in the embedding dimension) window of the the encoder's embedding to form all decoder inputs. The complete encoder-decoder information retention model is described by Equation 3 and is depicted in Figure 1.

$$Y = \mathcal{D}(P(\mathcal{E}(X, \theta)_{[:,-1]}), \theta_{\mathcal{D}}) \tag{3}$$

4.2 Causal Language and Retrieval Models but not Autoencoders exhibit Poor Memory Accuracy

Upon training (on FineWeb for 13 billion tokens) we find inversion loss tends to plateau and decrease only marginally as the number of training samples increases (Figure 1). We find that models trained for next token prediction and retrieval exhibit little increase in information retention compared to untrained models, but that autoencoders trained for input regeneration have substantially more information (Table 1). We confirmed that the decoder-only training approach for information retention measurement does not lead to substantial under-estimation of information retention in pretrained autoencoders, where the encoder's information content is already known (Figure A1, Table 1).

To restate, we find that for models trained to predict next tokens or retrieve passages, the last token's last hidden layer contains relatively littleinput information and is incapable of recovering more than a small fraction of input tokens. This does not mean that hidden layers in these models are low-information, however, and we observe that perfect input token recovery is trivially easy to train when the inversion decoder is fed *all* token's last hidden layers: a mere 1/50th of the model's pretraining data and compute is required to train an inversion decoder to near-perfect (>99%) input token identification accuracy (Figure A2). This indicates that internally causal and retrieval models maintain nearly all input information across their embeddings of all tokens, but that when restricted to one single token's embedding only a very small amount of this information remains.

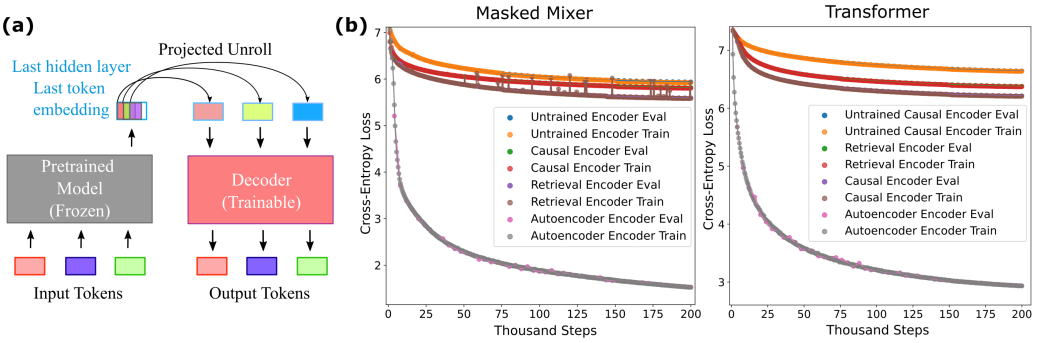


Figure 1: (a) Information retention experimental approach and (b) Example Masked Mixer (left) and Transformer (right) training runs, $n_{ctx} = 512, n_l = 8$ Mixers $d_m = 1024$ and Transformers $d_m = 512$ for 200k steps, 13.1B tokens.

Table 1: Information Retention measurements for models trained with the given objective. 'Causal -> Retrieval' signifies causal pretraining followed by summary-to-text retrieval InfoNCE-based training. Validation models have trainable encoders as well as decoders, all others are frozen.

	Masked Mixer			Transformer		
Encoder Model	Loss	H_r	Accuracy	Loss	H_r	Accuracy
Autoencoder ($n_{ctx} = 256$ validation)	0.01	0.999	99.6 %	0.09	0.99	95.4 %
Autoencoder (validation)	0.292	0.968	91.84 %	2.73	0.698	43.3 %
Autoencoder	0.435	0.952	76.96 %	2.935	0.624	33.5 %
Untrained	5.937	0.343	4.67 %	6.643	0.264	4.67 %
Causal	5.815	0.356	5.48 %	6.214	0.31	5.01 %
Causal -> Retrieval	5.594	0.381	5.45 %	6.380	0.29	5.29 %
Autoencoder -> Retrieval	5.64	0.375	5.46 %	-	-	-

Table 2: FineWeb (unless otherwise noted) LLM information retention for the given context windows. Entropy ratio H_r and token accuracies in terms of % correctly identified are reported.

Model	$n_{ctx} = 16$	$n_{ctx} = 64$	$n_{ctx} = 256$	$n_{ctx} = 512$
BERT large	0.719 (40.2%)	0.481 (11.6 %)	0.406 (6.6 %)	0.376 (5.96 %)
Llama 3.1 1b	0.665 (34.8 %)	0.490 (11.2 %)	0.434 (6.0 %)	0.419 (5.30 %)
Qwen 3 0.6b	0.614 (35.7 %)	0.459 (12.7 %)	0.414 (6.62 %)	0.402 (5.61 %)
BERT large (FineMath)	0.891 (72.3 %)	0.641 (26.8 %)	0.528 (12.6 %)	0.491 (9.6 %)

4.3 Autoencoder Memories are Non-Trivial

We reasoned that the high memory fidelity, or equivalently high compression, in autoencoders could be caused by the encoder learning to embed the input in a non-data-dependent manner, which we call a 'trivial' autoencoding. One example of such an encoding is if encoder's embedding dimension is at least as large as the number of tokens in the input, $d_m \geq n_{ctx}$, and the encoder learned to simply encode one token per embedding element. We tested the data dependence of trained autoencoders by applying these models to sequences of uniform randomly distributed tokens, and observe that this result in large increases in loss compared to in-distribution samples (Figure A3) and furthermore that trivial embeddings are not practically trainable with our methods (Figure A3) and that marginally out-of-distribution data is generalized well (Figure A4).

4.4 Information Retention does not Increase with Causal Training Scale

We tested whether increases in model size and pretraining compute and data would result in greater information retention by training decoders using embeddings from off-the-shelf LLMs. We investigated BERT large [Devlin et al., 2019] with 5x the parameters and trained with 10x the compute as the models detailed abvove, as well as Qwen 3 [Yang et al., 2025] 0.6b with 9x the parameters and Llama3.1 1b at 14x the parameters [Grattafiori et al., 2024], both trained with >1000x the compute. For this experiment, we freeze the pretrained model's token embedding and transformer model layers and feed the last hidden layer output to a decoder after unrolling. As shown in Table 2, there is a modest increase in token accuracy and entropy ratio for Transformers with the same context as investigated previously ($n_{ctx} = 512$).

These results contrast with those obtained in [Morris et al., 2023], where conversational prompts were able to be recovered from causal LLMs. We hypothesized that there may be two reasons for this; firstly that shorter ($n_{ctx} \leq 64$) inputs should be easier to invert as they contain less information, and for the same reason less-diverse inputs should be easier to invert. We tested the first hypothesis by observing input regeneration for smaller contexts, and we find an inverse correlation between prompt length and regeneration accuracy (Table 2). The second hypothesis was tested by training decoders to invert BERT outputs on a math-specific subset of the FineWeb, FineMath 4+ [Allal et al., 2025], and we observe double the token recovery accuracy on this dataset.

5 Memory Models

5.1 Architecture and Implementation

Are memory embeddings useful for causal modeling? We introduce a memory architecture where an encoder is responsible for mapping a sequence of tokens to an embedding, and a decoder takes any number of these embeddings in addition to any additional tokens of a sequence. This architecture is similar to one that has been independently introduced by [Prakash et al., 2025] and has notable advantage over decoder-only memory models [Bulatov et al., 2022] in that it is capable of parallelizable processing of input sequences at inference, as the formation of any single embedding requires no input from any other embedding such that the encoder may be applied to all s chunks in parallel. We train for causal modeling and in order to control for differences in decoder context length we also train memory models without memory embeddings. We explore two varieties of memory model embedding introduction, one where a variable number of memory embeddings are introduced and token embeddings follow immediately, and the other where memory embeddings are introduced and padded such that the embeddings from the same memory chunks are always found in the same decoder positions. We find little difference in causal training efficiency, and in the rest of this work use the fixed embedding approach (Figure A5).

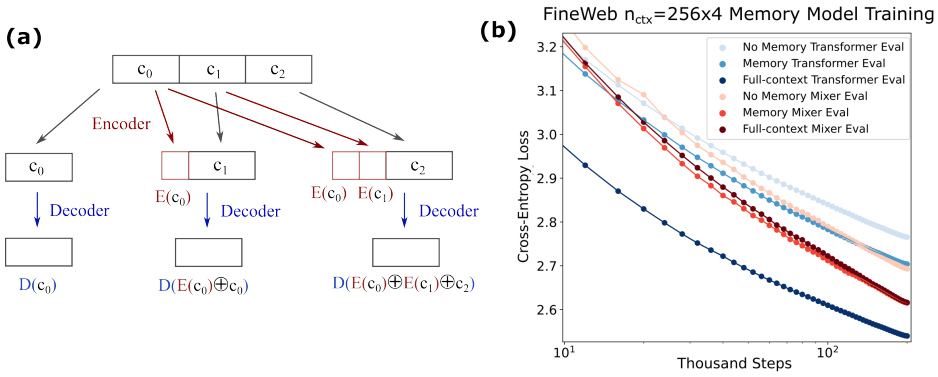


Figure 2: (a) Memory Model Architecture and (b) $n_{ctx} = 256$ per chunk, $s = 4$ chunk causal training characteristics on FineWeb. Mixers with $d_m = 512$ for encoders, $d_m = 1024$ for decoders and Transformers $d_m = 256$ and $d_m = 512$ respectively for compute equivalence.

5.2 Causal Language Training Efficiencies of Memory Models

Causal training efficiencies of memory models are given in Figure 2. We find that memory models can be nearly as efficient to train as full-context models but that this is somewhat architecture- and context window chunk size- dependent. We compared these causal training efficiencies to those from an architecture where decoders recurrently form and process memory embeddings [Bulatov et al., 2022], and we find that the parallelized encoder-decoder memory models are slightly more efficient to train on FineWeb (Figure A6).

Viewing the sole task of a memory model encoder as capturing as much input information as possible, we can first training encoders to capture information and then train decoders to predict next words using that information. This allows for increased parallelism and throughput during training, as encoders can be trained, frozen, and can applied iteratively during decoder training and embeddings can even be entirely precomputed if desired. As shown in Figure A8, frozen encoders memory model training efficiency approaches that of memory models with both encoder and decoder trainable.

5.3 Pretrained LLMs as Memory Model Decoders

We next investigated whether or not it would be possible to adapt an LLM trained for next token prediction to use memory model embeddings. Using Llama 3.2 (1b) [Grattafiori et al., 2024] as the memory model decoder, we trained autoencoders to use the Llama 3 tokenizer, attached encoders

from these models to trainable decoders and trained on the $n_{ctx} = 256, s = 4$ copy task depicted in Figure 3. We find that these models require relatively small learning rates to prevent divergence due to difficulties backpropagating gradients into the encoder, and as a result train relatively slowly but do minimize the copy loss metric (Figure A7), indicating successful memory information use. We next investigated whether memory model training would interfere with the abilities the LLM gained during pretraining via benchmark evaluation of the decoder: we find that benchmark accuracy decreases somewhat after memory model training but curiously is most changed for tasks testing pretrained (non-input) information retention (Tables A1 and A2).

6 Combined Objectives for Accurate Memories

To be able to make use of arbitrary input information, memory models must be capable of both encoding token information into embeddings and processing this information in the decoder, so we explore the potential for memory models to learn to do so in this section. We use three methods for evaluating memory accuracy in memory models: the encoder-decoder information retention test detailed in Section 4.1, a copy objective originally introduced by [Hochreiter and Schmidhuber, 1997] in which a sequence of tokens is sampled from the dataset, the first half of the sequence is copied and the model is evaluated on its ability to accurately predict these copied tokens, and what we term a 'blank copy' objective in which the tokens are masked and only memory embeddings are available for the decoder when attempting to predict the copied tokens (Figure 3). The copy task as applied to a memory model is a test of the model's ability to transfer sufficient information from the first $s/2$ chunks for a s chunk model to the decoder. This is a less information-intensive task than autoencoder information task in that all previous tokens not encoded by a memory embedding are given to the decoder, such that a model with no information retention ability achieves a copy accuracy equivalent to its next token prediction accuracy (which from Table 3 is around 42%).

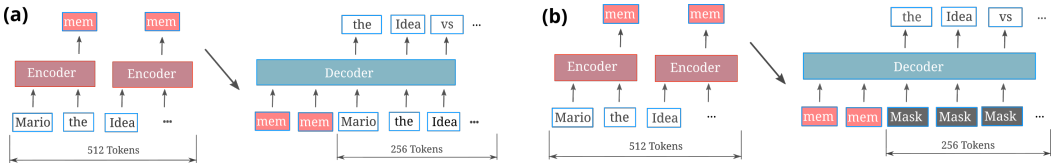


Figure 3: (a) Copy and (b) Blank Copy experimental schematics

6.1 Causal Memory Model Embeddings have Little Information

Standard causal architectures such as Transformers contain hidden layers that retain practically all input information across all token embeddings (Figure A2), such that the model can effectively look up input information when predicting any next token from these intermediate states (or from a key-value cache storing these states). As there is much less information present in single embeddings for causal-trained models, the same lookup-style information access cannot be reasonably expected to be possible given only these embeddings. Being that memory models are trained using the same causal objective, we wondered whether the embedding memories in these models would be similarly low-information.

We first measured memory embedding information via copy task training in which the memory model encoder is frozen and the decoder trained, before measuring performance on the more information-intensive encoder-decoder information extraction and blank copy tasks. Causal memory models exhibit minimal increases in copy accuracy over no-memory controls (Figure 4 (d)), do not exhibit substantially higher information retention than full-context decoders (Figure 4 (b)), and are incapable of learning the blank copy task (Figure 4 (c)). These results suggest that in their current state memory models cannot be expected to be capable of accurate information lookup over their memories, only over the non-memory input tokens.

6.2 Curriculum Learning for Decoding Frozen Encoder Memory

We were surprised to observe that frozen encoder embeddings conferred no benefit to copy model training (Figure 4 (a)), despite these embeddings containing a near totality of input information

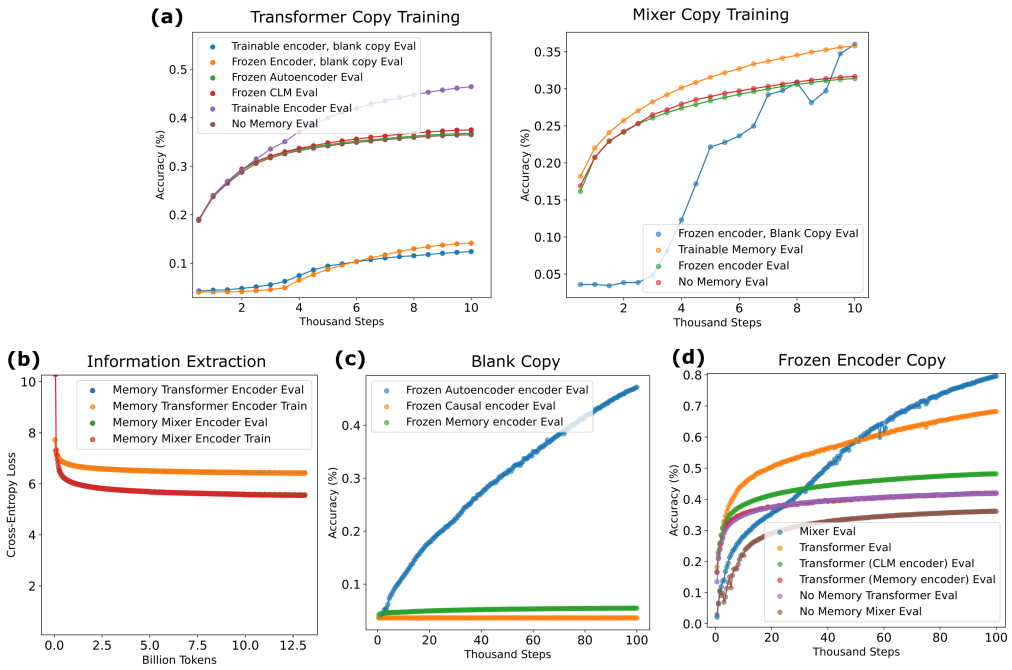


Figure 4: Memory Model Information Content. (a) Memory model training for the copy task, (b) $d_m = 256, n_{ctx} = 512$ Transformer and $d_m = 512, n_{ctx} = 256$ Mixer memory model encoder information, (c) Memory Transformer training on the blank copy task, (d) Frozen encoder copy training, all curriculum trained except CLM encoder and Memory encoder. All models $d_m = 512, n_l = 16, n_{ctx} = 256, s = 4$ unless otherwise noted.

Table 3: Combined and causal trained memory model performance on causal and copy tasks.

Objective	Combined		CLM		Combined (Blank Copy)		
Metric	H_r	Accuracy	H_r	Accuracy	H_r	Accuracy	
CLM Eval	0.675	42.07 %	0.684	43.15 %	0.669	41.5 %	All Trainable
Copy Eval	0.947	86.1 %	0.688	43.65 %	0.756	50.6 %	
CLM Eval	0.664	40.8 %	0.677	42.32 %	0.660	40.4 %	Frozen Encoder
Copy Eval	0.927	82.7 %	0.677	42.63 %	0.759	52.6 %	

(Table 1). Reasoning that this might be because predicting next tokens while disregarding memory embeddings may be easier than decoding information-rich embeddings, we tested a curriculum learning approach where memory model decoders first learn to process information-rich memories without token embeddings followed by introduction of token embeddings. After training frozen autoencoder encoder memory models on the blank copy task (Figures 4 (c), A9) we then trained the models on the copy task and find that copy accuracy further increases (Figure 4 (d)), indicating that the memory model decoders are capable of using sufficient information from frozen autoencoder encodings for copying. In contrast, causal-trained encoders are unsuitable for this curriculum approach as they are poorly trainable to copy without token information (Figure 4 (c)) and result in a modest increase in copy accuracy when trained without this curriculum (Figure 4 (d)).

6.3 Causal and Copy Combined Objectives for Accurate Memory Formation

Given that memory models are capable of next token prediction and accurate memory formation in isolation, we investigated whether these models are capable of both in conjunction. We trained memory models using a linear combination of cross-entropy loss of next token prediction and copy training (X_s denotes shifted inputs, X_c denotes copied inputs with a copy delimiter added, and X_{sc} shifted copied inputs) given in Equation 4. As shown in Table 3, memory models are capable of

Table 4: Causal versus copy and causal combined objective benchmark accuracy. Both models are frozen encoder-based memory transformers with $d_m = 512, n_l = 16, n_{ctx} = 256, s = 4$.

Information Benchmarks	**Causal**	**Combined**
IFEval (strict instance)	0.2482	**0.2506**
LongBench		
Code Completion	13.17±0.31	**16.48±0.33**
Few-Shot Learning	0.80±0.11	**1.57±0.12**
Multi-Document QA	0.54±0.09	0.51±0.09
Single Document QA	0.81±0.07	**1.07±0.08**
Summarization	2.89±0.11	**3.41±0.12**
Information Retrieval	0.86±0.34	0.83±0.37
SWDE (contains)	0.0	**0.81**
Other Benchmarks	**Causal**	**Combined**
ARC easy	44.2±1.02	43.73±1.02
WikiText (BPB)	**1.6752**	1.6850
Lambada-OpenAI	**18.65±0.54**	17.21±0.53
HellaSwag	27.71±0.45	27.52±0.45

learning to accurately copy and predict next tokens. For the more difficult task of copying without tokens provided, training via the combined objective function yields lower accuracies, and curriculum-trained frozen encoders have higher copy accuracy than memory models with trainable encoders (Table 3). Training memory models to form and decode information-rich embeddings results in a modest decrease in causal prediction efficiency, as does the use of frozen autoencoder embeddings although the latter comes with the benefit of increased throughput. We find that training on the combined objective function leads to improvements in input information-related but not general language performance compared to causal training alone (Table 4), suggesting that recall training applied to the encoder alone is sufficient to increase memory formation and use fidelity, even though the decoder was not directly trained to use this information for token prediction.

$$\mathbb{L}' = \mathbb{L}(O(X, \theta), X_s) + \mathbb{L}(O(X_c, \theta), X_{sc}) \tag{4}$$

7 Discussion

7.1 Information retention in Masked Mixers and Transformers

We find that Masked Mixers are typically more efficient encoders whereas Transformers are more efficient decoders, but that embeddings from Masked Mixers are incompatible with Transformer decoders as introducing Mixer memory model embeddings to Transformer decoders leads to a decrease in training efficiency relative to non-memory controls regardless of whether the encoder is frozen or trainable (Figure A10), implying that the learned embedding manifolds differ substantially between these architectures, but why this would be is currently unclear.

7.2 How much information is enough?

In this work we have examined information retention using various methods (autoencoder input regeneration, copy, blank copy) and metrics (entropy ratio and token accuracy). A natural question to ask is which method and metric most closely reflects the ability of a model to perform arbitrary input information access. In the context of memory models it may be hypothesized that copy accuracy is most relevant to tasks where less information needs to be accessed, but this remains to be explored.

This answer is likely different for retrieval, which is typically performed by computing the cosine similarity between the query embedding and target embeddings and finding pairs with the highest similarity. Modern retrieval approaches often use multiple embeddings, reranking models, and approaches such as BM25 to increase the likelihood that the retrieved text matches the target. We find that embedding models contain only a fraction of input information, explaining some of the findings of recent work [Weller et al., 2025] observing that retrieval model accuracy is often lower than what is theoretically possible for information-rich inputs. It remains to be seen whether standard retrieval training methods are compatible with maximal information retention.

Acknowledgments and Disclosure of Funding

The author would like to thank IBM for support during the writing of this paper.

References

Aydar Bulatov, Yuri Kuratov, and Mikhail S. Burtsev. Recurrent memory transformer, 2022. URL https://arxiv.org/abs/2207.06881.

Ivan Rodkin, Yuri Kuratov, Aydar Bulatov, and Mikhail Burtsev. Associative recurrent memory transformer, 2025. URL https://arxiv.org/abs/2407.04841.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context, 2019. URL https://arxiv.org/abs/1901.02860.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Xiaoqiang Lin, Aritra Ghosh, Bryan Kian Hsiang Low, Anshumali Shrivastava, and Vijai Mohan. Refrag: Rethinking rag based decoding, 2025. URL https://arxiv.org/abs/2509.01092.

Sainbayar Sukhbaatar, Da Ju, Spencer Poff, Stephen Roller, Arthur Szlam, Jason Weston, and Angela Fan. Not all memories are created equal: Learning to forget by expiring, 2021. URL https://arxiv.org/abs/2105.06548.

Jatin Prakash, Aahlad Puli, and Rajesh Ranganath. Attention and compression is all you need for controllably efficient language models, 2025. URL https://arxiv.org/abs/2511.05313.

John X. Morris, Wenting Zhao, Justin T. Chiu, Vitaly Shmatikov, and Alexander M. Rush. Language model inversion, 2023. URL https://arxiv.org/abs/2311.13647.

Orion Weller, Michael Boratko, Iftekhar Naim, and Jinhyuk Lee. On the theoretical limitations of embedding-based retrieval, 2025. URL https://arxiv.org/abs/2508.21038.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Benjamin L. Badger. Masked mixers for language generation and retrieval, 2025. URL https://arxiv.org/abs/2409.01482.

Benjamin L. Badger. Adversarial examples from dimensional invariance, 2023. URL https://arxiv.org/abs/2304.06575.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. The fineweb datasets: Decanting the web for the finest text data at scale, 2024. URL https://arxiv.org/abs/2406.17557.

Benjamin L. Badger and Matthew Neligeorge. Know your limits: Entropy estimation modeling for compression and generalization, 2025. URL https://arxiv.org/abs/2511.10618.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019. URL https://arxiv.org/abs/1810.04805.

An Yang, Anfeng Li, Baosong Yang, and Beichen Zhang et al. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Aaron Grattafiori, Abhimanyu Dubey, and Abhinav Jauhri et al. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Hynek Kydlíček, Agustín Piqueres Lajarín, Vaibhav Srivastav, Joshua Lochner, Caleb Fahlgren, Xuan-Son Nguyen, Clémentine Fourrier, Ben Burtenshaw, Hugo Larcher, Haojun Zhao, Cyril Zakka, Mathieu Morlon, Colin Raffel, Leandro von Werra, and Thomas Wolf. Smollm2: When smol goes big – data-centric training of a small language model, 2025. URL https://arxiv.org/abs/2502.02737.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8): 1735–1780, 1997. doi:10.1162/neco.1997.9.8.1735.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi 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 M. Rush. 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, October 2020. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/2020.emnlp-demos.6.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 07 2024. URL https://zenodo.org/records/12608602.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *ArXiv*, abs/1803.05457, 2018.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence?, 2019. URL https://arxiv.org/abs/1905.07830.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The lambada dataset: Word prediction requiring a broad discourse context, 2016. URL https://arxiv.org/abs/1606.06031.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding, 2021. URL https://arxiv.org/abs/2009.03300.

A Appendix

Code for this work is available at https://github.com/blbadger/memorymodels

A.1 Training Details

We train using models by integrating Pytorch [Paszke et al., 2019] defined modules into the Huggingface transformers Trainer [Wolf et al., 2020] utility, adding preprocessing or evaluation processing as required. We train using the torch implementation of AdamW [Loshchilov and Hutter, 2017], typically with a 500 step linear warm up, peak learning rate of $\lambda = 2 * 10^{-4}$ for Transformers and $\lambda = 5 * 10^{-4}$ for Mixers (tuned to each model), and with a linear decay for the duration of training, and $n_{ctx} = 512$ was used for the context window unless otherwise noted. Causal full-context models, memory models, and autoencoders were typically trained with a global $b = 128$ batch size

via torch-native distributed data parallel. Models with a context window of $n_{ctx} = 1024$ (as was typical for copy model training) were trained with a batch size of $b = 64$. For variable-context LLM information retention tests, a constant total number of tokens trained by assigning the batch size to be a function of context length, $b = \lfloor 32768/n_{ctx} \rfloor$, which equates to $b = 64$ for $n_{ctx} = 512$.

Note that for information measurement experiments frozen token embedding layer with a trainable transformation results in less decrease to validation model accuracy, and this is performed unless otherwise noted.

A.2 Figure Details

Raw plots were obtained using matplotlib or seaborn, plots were assembled into figures using Inkscape, and plot titles were added and plot legends and axis titles were moved or enlarged for clarity. No other modifications (besides size and position) were applied to legends and axis titles, and no modifications were made to axis demarcations, data points, or any other aspect of each plot.

A.3 Benchmark Details

Benchmark evaluations were obtained by modifying Eleuther AI's Language Model Evaluation Harness [Gao et al., 2024]. Code for applying memory models to the benchmark suite was integrated into a fork of this repository, and is available at https://github.com/blbadger/lm-evaluation-harness. All benchmarks are zero-shot unless otherwise noted.

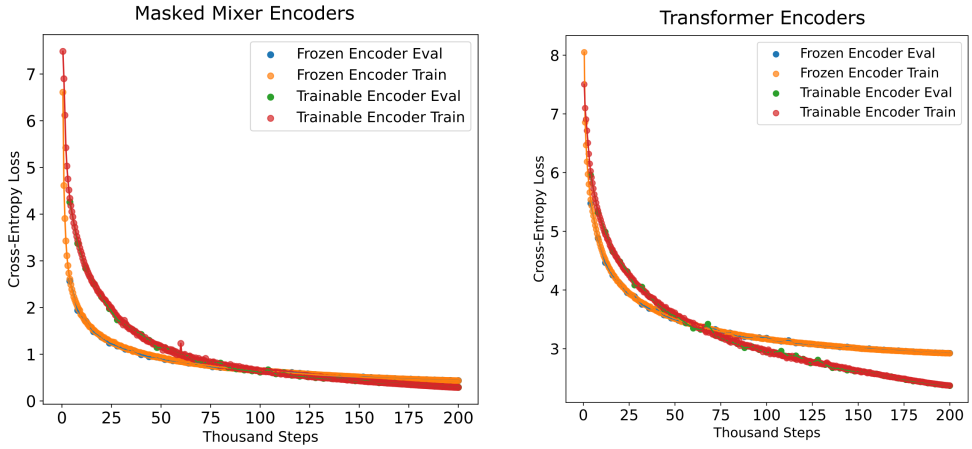


Figure A1: Mixer and Transformer information retention validation training curve examples on FineWeb. Transformers are $d_m = 512, h = 4$ and mixers $d_m = 1024$ non-headed, both with $n_{ctx} = 512, n_l = 16$.

A.4 Pretrained LLM memory model training

We benchmark for general question answering via Arc-Easy [Clark et al., 2018], sentence completion using HellaSwag [Zellers et al., 2019] and Lambada (OpenAI processed) [Paperno et al., 2016], and pretraining information retention via MMLU [Hendrycks et al., 2021] using Eleuther-AI's evaluation harness [Gao et al., 2024]. Before memory model training, the pretrained Llama 3.2 achieves the benchmark accuracies shown in Table A1, and after training for 32k steps the decoder's benchmark accuracies are shown in Table A2.

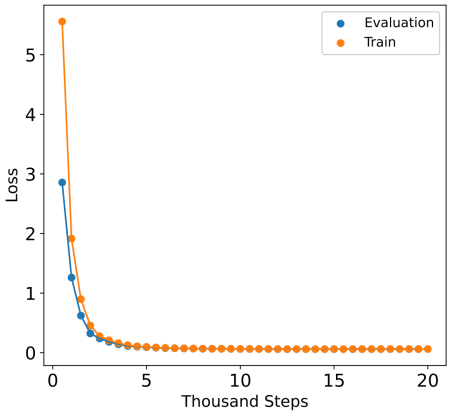


Figure A2: Causal transformer ($\mathbb{L}_{CEL} = 2.64$ on FineWeb) $d_m = 512, h = 4$, with $n_{ctx} = 512, n_l = 16$ model inversion, inverter model is $n_l = 8, d_m = 512$. These inversion losses correspond to 96.5% input token identification accuracy at 2k steps, 99.1% accuracy at 4k, and 99.5% accuracy at 10k steps.

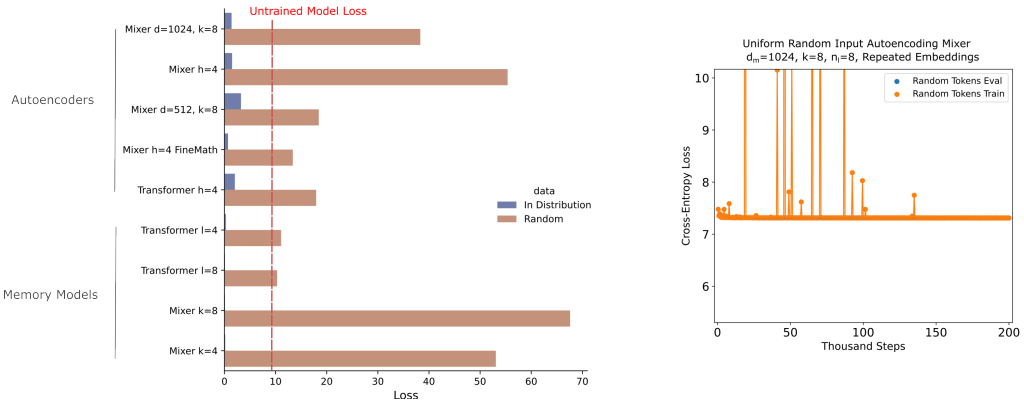


Figure A3: Autoencoders and Memory Model Encoder distribution dependence. Left, Autoencoder and Memory Model generalization to OOD uniform random distribution inputs versus hold-out FineWeb in-distribution inputs. Right, Masked Mixer autoencoders do not learn to model uniform random inputs. Memory models are 'oracle' memory models where a full-context encoding is added to a causal decoder and trained for next token prediction, resulting in very small loss.

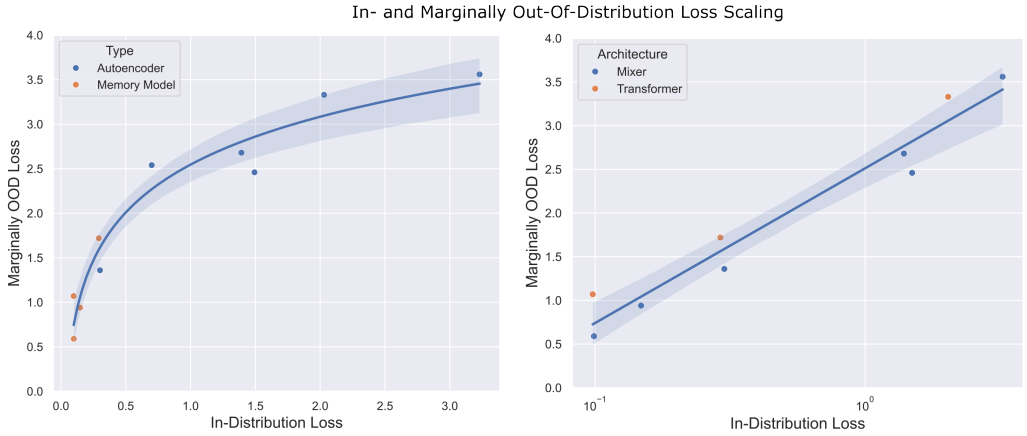


Figure A4: Autoencoder and Memory Model (oracle) encoder generalization. Models were trained on FineWeb-edu with FineMath 4+ used for marginally out-of-distribution samples.

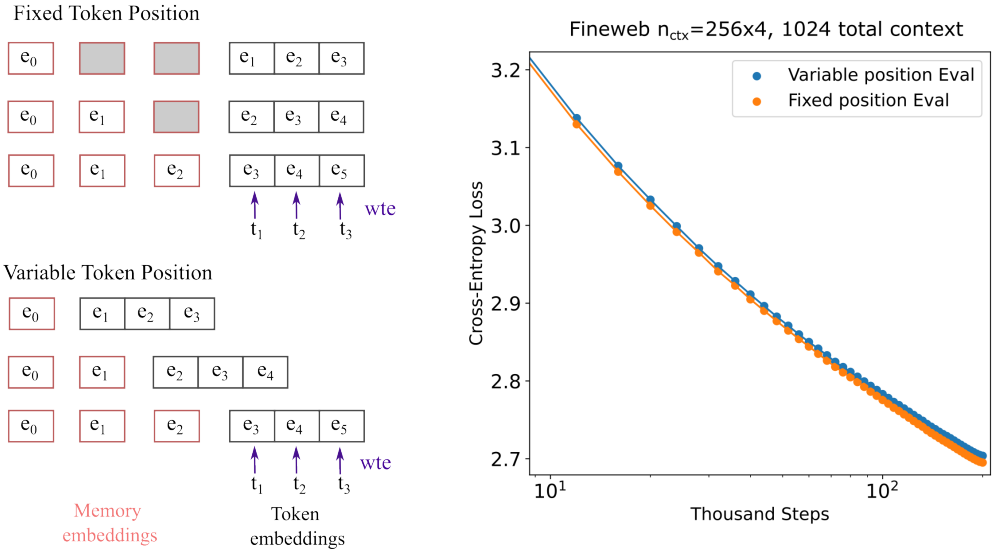


Figure A5: Fixed versus variable position memory Transformer training on FineWeb.

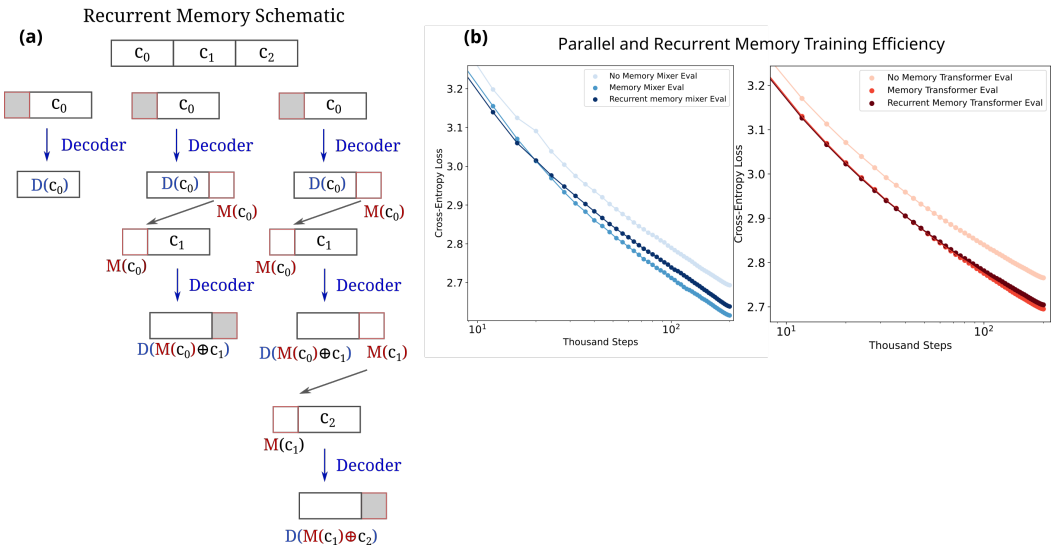


Figure A6: Parallel and recurrent memory model training efficiency on FineWeb.

Table A1: Llama 3.2 (1b) benchmarks before copy memory training.

Tasks	Version	Filter	n-shot	Metric		Value		Stderr
arc_easy	1	none	0	acc	↑	0.6633	±	0.0097
		none	0	acc_norm	↑	0.6170	±	0.0100
hellaswag	1	none	0	acc	↑	0.4805	±	0.0050
		none	0	acc_norm	↑	0.6427	±	0.0048
lambada_openai	1	none	0	acc	↑	0.6222	±	0.0068
		none	0	perplexity	↓	5.4344	±	0.1288
mmlu	2	none		acc	↑	0.3775	±	0.0040
- humanities	2	none		acc	↑	0.3515	±	0.0069
- other	2	none		acc	↑	0.4303	±	0.0088
- social sciences	2	none		acc	↑	0.4030	±	0.0087
- stem	2	none		acc	↑	0.3394	±	0.0083

Table A2: Llama 3.2 (1b) Memory Model Decoder benchmark evaluations at 32k copy training steps.

Tasks	Version	Filter	n-shot	Metric		Value		Stderr
arc_easy	1	none	0	acc	↑	0.6646	±	0.0097
		none	0	acc_norm	↑	0.5976	±	0.0101
hellaswag	1	none	0	acc	↑	0.4638	±	0.0050
		none	0	acc_norm	↑	0.6159	±	0.0049
lambada_openai	1	none	0	acc	↑	0.5960	±	0.0068
		none	0	perplexity	↓	6.4017	±	0.1668
mmlu	2	none		acc	↑	0.2872	±	0.0038
- humanities	2	none		acc	↑	0.2837	±	0.0066
- other	2	none		acc	↑	0.3238	±	0.0083
- social sciences	2	none		acc	↑	0.2912	±	0.0082
- stem	2	none		acc	↑	0.2525	±	0.0077

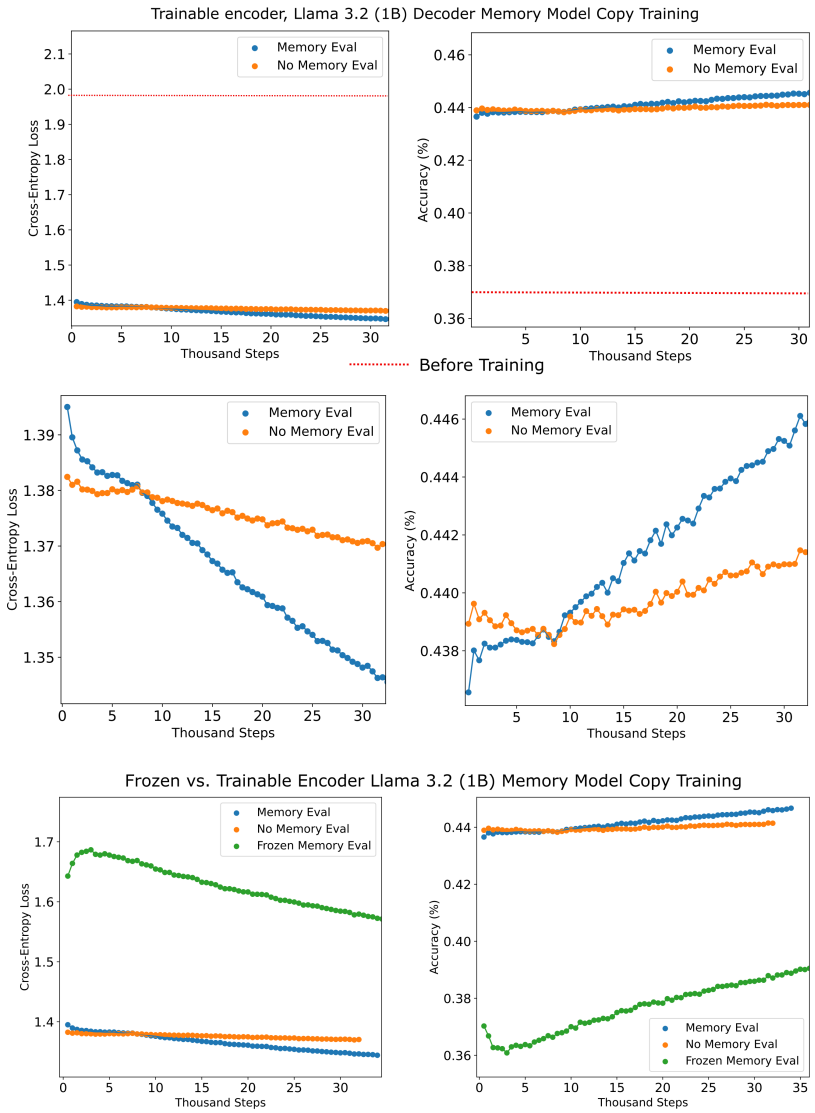


Figure A7: Llama 3.1 (1b) decoder memory model copy training. Models are $n_{ctx} = 256, s = 4$ with 512 tokens copied from FineWeb sampled inputs.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Each claim in the abstract and introduction is detailed and evidence supporting each claim is presented, and every effort has been made to avoid any claims that are not exactly supported by experimental evidence.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.

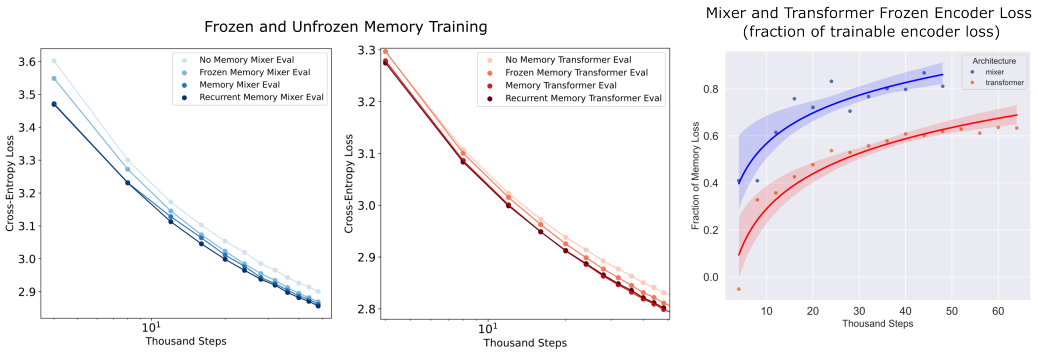


Figure A8: Memory model ($n_{ctx} = 512$, $s = 4$ chunks) causal training on FineWeb.

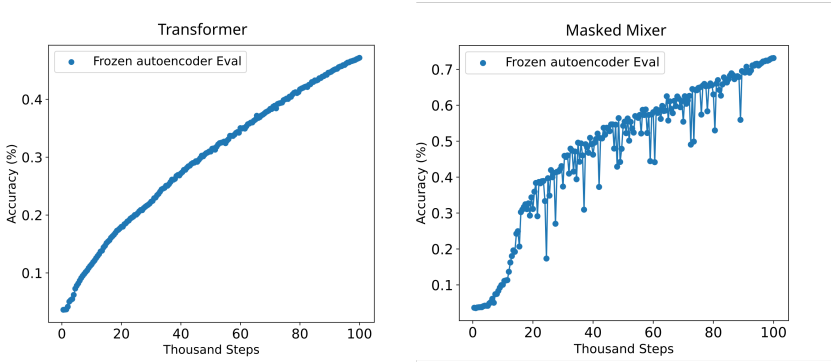


Figure A9: Blank Copy Training on FineWeb, 512 copy tokens, $d_m = 512, n_l = 16, n_{ctx} = 1024$

- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Limitations to specific experiments are provided throughout this work, and larger limitations are discussed in the Conclusion.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.

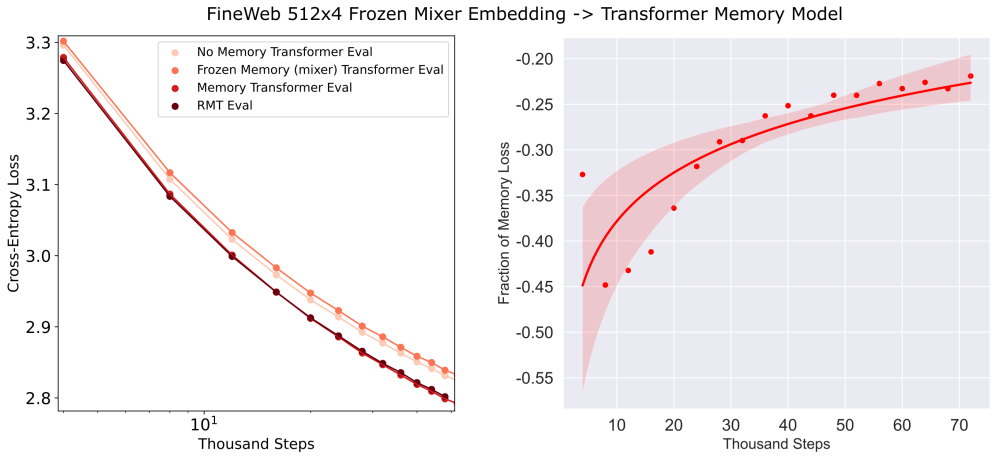


Figure A10: Memory Model Training on FineWeb, Masked Mixer encoder and Transformer decoder ($n_{ctx} = 512, s = 4, d_m = 512, n_l = 16$).

- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: No proofs are provided in this work.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer:

Justification: All code for reproducing experiments is provided, and datasets are public.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The code, main, and supplementary experimental results together provide comprehensive details on exact experimental replication.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.

- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Provided in the Appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: The experiments provided are largely representative examples of replicates for which significant variance was not observed, due to the additive nature of stochastic processes (for example, training replicates typically converged to within 1% of the values given in this paper). Errors are given where appropriate and computationally feasible.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: Compute resources are explicitly stated in the main body (no more than 24 hours on a single H100 for example) and in the Appendix.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Does conform.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: This work was deemed to be of too small scope to have significant societal impacts: the models trainable by the experiments described have no abilities for deepfakes, privacy, or other considerations.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.

- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: No high risk items introduced.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Citations are provided for all code, data, models, and tools as required.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: Code is released and documented in that repo and here.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.

- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: No crowdsourced work in this study.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: No participants in this study.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [No]

    Justification: No LLMs used at any time.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.