Learning Is Not A Race: Improving Retrieval in Language Models via Equal Learning

Wanqian Yang, Aahlad Manas Puli, Rajesh Ranganath
New York University
New York, NY 10012, USA
{wanqian,apm470}@nyu.edu, rajeshr@cims.nyu.edu

Abstract

Many applications that modern large language models (LLMs) are deployed on are retrieval tasks: the answer can be recovered from context and success is a matter of learning generalizable features from data. However, this is easier said than done. Overparametrized models trained on cross-entropy loss can overfit on noise. We argue that such overfitting is prone to happen when the model can identify mechanisms that rapidly drive down the loss of certain tokens early on in training. Fitting some tokens early reduce gradient signals in later iterations, as such, remaining tokens are more vulnerable to noise overfitting. We dub this phenomenon unequal learning and show that LLMs with longer contexts or larger embedding sizes are prone to this failure mode. In this work, we argue that learning training samples at an equal rate helps counter such biases. We highlight two mechanisms that promote equal learning: (i) loss functions that regularize uniform margins across training samples, (ii) small learning rates (e.g. by warming up) at the start of training. We demonstrate these approaches on various synthetic and natural language datasets.

1 Introduction

If it looks like a duck, swims like a duck, and quacks like a duck, the maximum likelihood principle tells us we should expect — in all likelihood — a duck. This same principle underpins modern large language models (LLMs), which are largely transformer-based (Vaswani, 2017) neural networks p_θ trained to carry out next-token prediction by minimizing cross-entropy loss

$$\mathcal{L}(\theta) = \mathbb{E}_{\mathbf{x}_{1:T}\sim p_{\text{train}}}\Big[\sum_t \log p_\theta(\mathbf{x}_t|\mathbf{x}_{<t})\Big]. \quad (1)$$

One key task that these models are often deployed on is **retrieval from context**. Question answering (Rajpurkar, 2016; Yang et al., 2018), summarization (Nallapati et al., 2016; Narayan et al., 2018), and sentiment analysis (Maas et al., 2011; Socher et al., 2013) are amongst the many downstream language tasks whose success relies on the model's ability to retrieve and output the correct piece of information from context when given the appropriate prompt.

When is a duck not a duck? In this work, we crystallize the insight that retrieval is not a corollary of doing language modeling well. LLMs that can achieve zero loss on the training objective do not necessarily learn generalizable features. Accordingly, a well-trained transformer can fail to retrieve the correct information from context even when evaluated on samples drawn from the *same* distribution. Let us briefly survey how such failure modes might arise.

LLMs are overparametrized models and can learn multiple solutions from finite data. The solution (local optimum) that the model lands on depends on inductive biases arising from the complex interplay between factors such as model architecture, loss function, and optimization hyperparameters (Li et al., 2018; Bietti et al., 2024). These biases can cause the model to overfit, learning erroneous solutions that achieve zero loss on training samples but do not generalize at test time.

Recent work highlights one such bias: the ubiquitous cross-entropy loss. Linear models trained on cross-entropy learn a maximum-margin solution. To arrive at such a solution, the model provably exploits imperfect predictors ("shortcuts") to correctly classify *some* training samples — as long as enough noisy features exist to overfit the remaining samples (Puli et al., 2023).

We show that LLMs are susceptible to the same pitfall. Modern LLM design increasingly favors long context lengths (Lee et al., 2024; Achiam et al., 2023; Team et al., 2024). However, when trained on cross-entropy, a transformer's ability to retrieve the correct content quickly erodes as context length increases. We observe that *simply reducing batch size by squeezing samples into the same context window (thus keeping the overall number of samples per batch* ***constant****) can cause the model to overfit by increasing the amount of noisy features available.*

Biases can arise from other design choices. For example, LLMs are commonly trained via next-token prediction, i.e. teacher-forcing. This process can itself induce bias (Bachmann and Nagarajan, 2024). If earlier tokens in the response reveal information about the latter tokens in the sequence, a next-token predictor will inevitably leverage this correlation. This creates failure on "lookahead" tasks where the correct predictor requires planning later tokens ahead of earlier tokens.

How, then, do we encourage our models to learn generalizable features for retrieval? In this work, we argue that an useful auxiliary objective is **equal learning** of

Findings of the Association for Computational Linguistics: EMNLP 2025, pages 23200–23211
November 4-9, 2025

the individual conditional distributions $p_{\text{train}}(\mathbf{x}_t|\mathbf{x}_{<t})$ in (1). In the failure modes identified above, the model exploits correlations between tokens to quickly drive down the loss — that is, it learns to model certain parts of the sequence (or certain samples altogether) *faster* than others. This differential in learning is precisely how the model quickly learns a shortcut mechanism for part of the sequence. Once those terms have been fitted, they no longer contribute to the batch gradient signal, allowing the model to more easily fit the remaining terms with noise. To combat this tendency, we argue that enforcing uniform learning is a useful inductive bias for learning generalizable features — in other words, all tokens should be learnt at an equal rate.

How do we teach the model to learn all terms uniformly? We propose **explicit regularization**, by designing a family of loss functions that regularize for small margins. As the regularization term contradicts the cross-entropy term (which prefers maximizing margins), optimizing the overall objective requires the model to learn a fixed value for the margin and prevents it from fitting selective tokens too early on in training. Furthermore, we recommend **warming up the learning rate schedule**. Keeping learning rates small at the start of optimization dampens the model's ability to rapidly fit certain parts of the sequence. This allows all terms in the sequence to contribute to the gradient signal for more iterations, preventing the model from noise overfitting.

We show that our approach can learn generalizable features on a number of diverse tasks that a standard cross-entropy-trained transformer fails on.

We summarize our contributions as the following:

1. We design a synthetic retrieval task and train a transformer using the standard cross-entropy loss. We demonstrate empirically that the transformer fails to learn generalizable features despite modeling training data perfectly. We show that this failure mode is especially acute with longer context window lengths and larger embedding sizes.

2. We argue that these failure modes can be prevented by enforcing an additional objective, which is the equal learning of all tokens in the batch.

3. We identify an approach — explicit regularization, in conjunction with learning rate warmup — that is effective at enforcing equal learning, and demonstrate its efficacy on a number of retrieval tasks.

2 Related Work

Inductive Biases in Neural Networks Understanding how various inductive biases influence learning in deep neural networks is a well-explored subject, and existing literature has explored phenomena such as spectral bias (Yang and Salman, 2019; Ronen et al., 2019; Rahaman et al., 2019; Cao et al., 2019; Xu et al., 2019), simplicity bias (Jo and Bengio, 2017; Baker et al., 2018; Shah et al., 2020; Vasudeva et al., 2024), and the role of gradient dynamics in learning (Gidel et al., 2019; Advani et al., 2020; Nagarajan et al., 2021; Puli et al., 2023).

Feature Learning and Biases in Transformers As the core architecture underpinning modern LLMs, understanding how transformers (Vaswani, 2017) learn is a key research topic and much existing work has sought to precisely identify and describe the internal mechanisms that govern feature learning (Elhage et al., 2021; Nanda et al., 2023; Power et al., 2022; Hanna et al., 2024). More specifically, recent work has also studied biases that occur in transformers (Bhattamishra et al., 2022; Vasudeva et al., 2024; Bachmann and Nagarajan, 2024). Our work broadly falls in this category, however, we specifically focus on improving feature learning for retrieval tasks where the correct mechanism can be learnt from context. We discuss a few closely-related papers below and how our approach differs from theirs:

Bachmann and Nagarajan (2024) show that next-token prediction itself is a paradigm that can learn to learning features that do not generalize. More specifically, teacher-forcing can fail on "lookahead tasks", which are tasks that require the model to compute future tokens ahead of earlier ones in order. They demonstrate this insight on a constructed graph search problem, where the prefixes of the response are strongly correlated to the rest of response. As such, a model trained via teacher-forcing can achieve zero training loss simply by conditioning on earlier tokens, but will fail to generalize at test time. This work's chief contribution lies in identifying the problem; the authors provide a bespoke solution to the graph problem that relies on prior knowledge of the true solution. Instead, we propose black-box solutions that do not require domain knowledge.

Focusing on shortcut learning, Nagarajan et al. (2021) and Puli et al. (2023) study a similar setting where the label is a deterministic function of the inputs, however, a shortcut mechanism exists that is strongly correlated to the label under the training distribution only. They show that empirical risk minimization (ERM) with the cross-entropy loss will prefer to learn the shortcut — provably so in the case of linear classifiers. Both works attribute this phenomenon to the ERM's bias towards learning a maximum-margin solution (Soudry et al., 2018), and as a fix Puli et al. (2023) propose losses to control the margin for binary classification. Our work extends their findings to the transformer architecture, showing that the same phenomenon can be observed in transformers under aggravating conditions such as long contexts. We relate the max-margin bias to unequal token learning and implement a set of solutions that operate over multi-class discrete sequences.

3 Failure Modes on Retrieval Tasks

In this section, we probe the conditions under which transformers fail to learn generalizable features when deployed on retrieval tasks. Under specific model and data conditions, we show that training on the standard cross-entropy loss creates learning dynamics that can

lead to high generalization error. As such, the cross-entropy loss can provide poor inductive biases.

We first formalize the problem setting. We assume that there is a mechanism (function) that deterministically maps a context to the correct output tokens. Formally, letting $\mathbf{x}, \mathbf{y}$ denote the context and the output tokens, the conditional distribution of the output given context is a delta function:

$$p_{\text{train}}(\mathbf{y} \mid \mathbf{x}) = \delta_{f(\mathbf{x})} \tag{2}$$

where f is the correct mechanism and $f(\mathbf{x})$ the answer.

On infinite data, the only global minimum of cross-entropy loss over output token prediction is the mapping f. However, on finite data, many other solutions can achieve low or zero loss. A model with sufficient capacity can, of course, trivially fit all training samples with noise (Yun et al., 2019). Depending on the task, there can be other mechanisms that predict output tokens correctly most of the time. For example, it is well-known that question answering (QA) datasets can be biased towards having answers in specific positions, e.g. the subject of the first sentence (Ko et al., 2020). A QA model can easily reach a local minimum by learning a mechanism that relies on positional cues.

3.1 When do transformers fail to generalize?

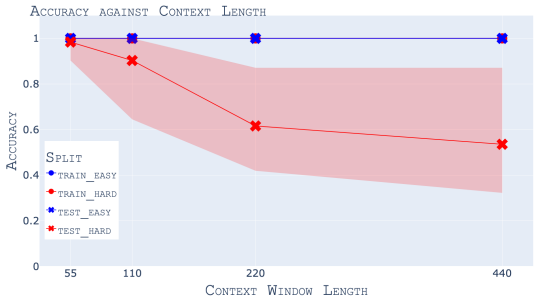
Between multiple low-loss solutions, can we characterize the optimization path that the transformer prefers? To shed some light on this question, we construct a running example throughout Sections 3 and 4.

Padded-Shifted Copy Task We consider a retrieval task as follows: the training distribution consists of strings of the form "`[seq][padding]<CLS>[seq]`". `[seq]` is a substring of length L_c, with each character uniformly and independently sampled from the English alphabet: $\texttt{[seq]}_p \sim \text{Unif}(\{\texttt{a}, \ldots, \texttt{z}\})$. `[padding]` is a substring of length L_p. With probability $p_s = 0.8$, the padding consists of a rightward Caesar shift of `[seq]` (e.g. `gravity` $\rightarrow$ `hsbwjuz`) repeated $\frac{L_p}{L_c}$ times. With probability $1 - p_s$, the padding consists of L_p characters uniformly and independently sampled. Foreshadowing the subsequent results, we call the 80% **easy** strings and the 20% **hard** strings. The test distribution is identical to the training distribution. The attention mask is 1 only for the L_c tokens following `<CLS>`.[1]

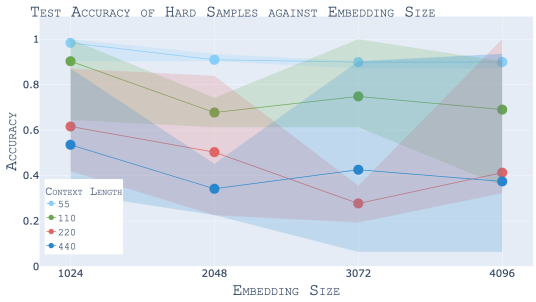
The goal is to copy `[seq]` from the start of the string to the end, i.e. the model is evaluated on its output of the L_c tokens following `<CLS>`. We measure **exact match accuracy** (EM) to `[seq]` as well as **character-level accuracy** (CL), i.e. fraction of the L_c tokens that match the corresponding position in `[seq]`. The dataset is generated once and fixed for all experiments.

We train a 4-layer GPT-NeoX (Andonian et al., 2023) with 4-head attention. For each experimental setting, we report results across 10 different random initializations. We report these two metrics on both the train and test sets, and on the easy and hard subsets separately. Further details can be found in Appendix B.

[1]We ablate for having an attention mask on all tokens, finding no difference in results; we choose to keep the restricted attention mask for ease of learning.



(a) Accuracy across train/test splits for both easy and hard groups. Across all context lengths, training accuracy is perfect for all samples, as is test accuracy on easy sequences. However, test accuracy on hard sequences degrade as context window increases, despite the total number of samples staying constant.



(b) For each context length in (a), we plot the test accuracy on the hard sequences while varying embedding size. On average, we see a slight decrease in the mean accuracy as embedding size increases. Variance (across runs) also increases markedly, with the worst run on `context_length = 440, embed_size = 3072 or 4096` performing no better than random (1/26).

Figure 1: Evaluation plots for $L_c = 1$ on the copy task.

In this section, we consider the $L_c = 1$ case (single-token retrieval) and $L_p = 50$. Figure 1 shows accuracy plots against various context lengths and embedding sizes. *Note that exact-match (EM) and character-level (CL) accuracies are identical here since $L_c = 1$, hence we will simply refer to both as "accuracy" below.*

In Figure 1a, we set the embedding size to 1024 and vary the context length. Crucially, we also modify batch size such that the ***total number of strings per batch remains constant*** for all experiments, i.e. a batch with fewer samples contain longer samples (more strings packed into each context). As such, the model receives the same information (i.e. same strings) every batch. Training accuracy is perfect on all strings across all experiments, proving that the model has optimized correctly. However, we see a divergence in *test* accuracy — the model is perfect on all easy test strings, however, its performance on hard test strings deteriorates as context length increases.

In Figure 1b, we consider the same context lengths, but this time, we increase the model embedding size to multiples of 1024. Once again, accuracy on all train strings and easy test strings are perfect. However, as embedding size increases, we observe a slight drop in the test accuracy of the hard strings. Furthermore, the variance in accuracy has also inflated — for the largest embedding sizes and the longest context windows, the worst performing runs have an accuracy of $\sim$1/26 on hard strings, i.e. no better than random.

Learning an Imperfect Mechanism On the surface, it is clear where the model went wrong. The correct mechanism f is simply to copy the first token following the <BOS> token, which a 4-layer transformer can provably learn (Elhage et al., 2021). However, the divergence in train and test accuracy indicates that the model has overfitted. Furthermore, the divergence in easy and hard test strings indicates that the model overfitted only on the hard strings. Since the easy and hard strings differ only on the padding, it is clear that the model has learnt a suboptimal mechanism g, where it retrieves a *padding* token and learns to reverse the Caesar shift.

3.2 Why do transformers fail to generalize?

This explanation, however, does not fully resolve our questions. On a noiseless dataset, the model can achieve zero loss by learning the true mechanism f: a simple task for a 4-layer transformer considering that the true token is always in the same relative position. What prompts it to learn g instead, which additionally requires the model to learn the Caesar shift function? Furthermore, why does increasing context length or embedding size lead to worse performance? The dataset and batching are identical across all experiments and the model receives the same information every iteration.

To understand this failure, we begin by examining the loss curves, as shown in Figure 2. We see a significant discrepancy between easy and hard strings. Whereas all easy strings are immediately optimized and reach zero loss within a single epoch (the first 50 iterations), the loss of hard strings decrease slowly, with a long tail. This phenomenon becomes more pronounced with increased context length or embedding size.

From Figure 2, we hypothesize that there are three separate regimes of training occurring:

- **Stratification:** At the start of training, batch gradients are dominated by the easy strings, which comprise the majority of samples in any given batch. Within a few steps of optimization, the model quickly learns the imperfect mechanism g, which is sufficient to reduce loss of all easy strings to zero.

  At this point, only hard strings have non-zero loss and contribute to the gradient signal for the rest of the training process.

- **Memorization** or **Rectification:** The model needs to reduce loss on the hard strings without forgetting the easy strings. As such, it has two choices: it can either overfit to the remaining hard strings by simply memorizing each individual sequence (Carlini et al., 2021), or it can recover from the suboptimal mechanism and instead learn the correct one. The two regimes are mutually exclusive because training loss goes to zero once memorization happens, preventing the model from rectifying.

Figure 2: Training loss for three different settings of conte xt length and embedding size trained across 400 epochs. In all settings, easy sequences are optimized almost immediately. The loss of hard sequences decreases more gradually, especially with longer context length or embedding size.

What we see is that larger embedding sizes and context lengths lead the model towards memorization rather than rectification, resulting in a gradual descent in loss as shown in the bottom two plots of Figure 2. In contrast, the top plot shows the rectification process, whereby learning the correct mechanism quickly reduces the loss of the remaining hard strings.

Let us consider how these stages of training arise. In any given batch, more easy strings exist than hard ones. For an easy string, attending to any padding token in the sequence (or any combination of padding tokens) is sufficient for retrieval. Since there are many padding tokens and only a single correct token, and since there are many more easy strings than hard strings, the model is far more likely to attend to a padding token than the correct token. As such, the model is predisposed

towards learning an imperfect mechanism g within the first few gradient steps.

Having learnt g, how does the model decide whether to memorize or rectify? The key factor here is the early stratification. Because easy strings were learnt and optimized to zero loss at the onset of training, *they contribute little gradient signal for the rest of the process*. As such, easy strings no longer provide a contrasting signal against noise overfitting in the latter phases, and it becomes far easier for the model, with enough capacity, to memorize the hard strings.

This process is exacerbated by longer context windows — individual strings drawn from the training distribution are uncorrelated, as such, packing more strings into the same window provide more sources of noise. Similarly, it is exacerbated by larger embedding sizes as the model simply has more capacity to memorize training data. The signal-to-noise ratio (relative lengths of `[seq]` and `[padding]`) as well as the ratio of hard and easy strings are also factors that affect how likely the model is to learn the incorrect mechanism g and stratify at the start of training.

3.3 The Pitfall of Unequal Learning

We crystallize the failure mode outlined above as the consequence of **unequal learning**. Formally, consider the objective (1) of the likelihood-maximizing autoregressive transformer. In a given batch $\{\mathbf{x}_{1:T}\}_{i=1}^{B}$, we can decompose the loss $\mathcal{L}(\theta)$ into individual conditional distributions for each of the $B \times T$ tokens

$$\sum_{i=1}^{B}\sum_{j=1}^{T} \log p_\theta(\mathbf{x}_{i,j} \mid \mathbf{x}_{i,<j}).$$

Unequal learning refers to the failure mode where the model learns these tokens at different rates. Some terms are optimized faster than others, and as individual terms' losses go down to zero, they stop contributing to the gradient signal. As training progresses, the transformer receives fewer and fewer signals. Accordingly, it is far more likely to memorize and overfit the later tokens than rectify and learn a generalizable mechanism that would fit all $B \times T$ distributions correctly.

As exemplified by the copy task, unequal learning is a consequence of gradient-based learning. All $B \times T$ tokens are equally weighted in the objective and contribute equally to the gradient. The direction of steepest descent (i.e. the negative gradient, with respect to the ℓ_2-norm) is simply the one that increases the overall likelihood of all tokens the most within a given ℓ_2-norm budget. This leads to stratification if suboptimal mechanisms exist that can quickly drive down the loss of a majority of tokens early on in training. In the case of the copy task, terms $p_\theta(\mathbf{x}_{i,j} \mid \mathbf{x}_{i,<j})$ corresponding to easy samples dominate the batch gradient signal.

Unequal learning at the start of training makes later rectification *harder*. The fewer terms remain in the gradient signal, the more likely these tokens are memorized by the model, especially with exacerbating factors such as sufficient noisy features (long context windows) or sufficient model capacity (large embedding sizes), as demonstrated in Figure 1.

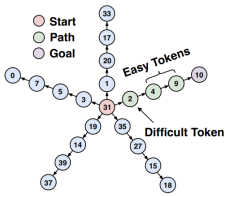


Figure 3: Graph problem from Bachmann and Nagarajan (2024). The task is to output a sequence to the start node to the goal node. The start node is the center node and the goal node is randomly chosen from the set of leaf nodes. The model is given the adjacency list of the graph representation and the start and goal nodes.

The presence of strong correlations between output and context tokens, as exemplified by the copy task, is one way unequal learning can arise. We highlight another setting that can give rise to this failure mode:

Example: Teacher-Forcing Bachmann and Nagarajan (2024) considers shortcut mechanisms that arise naturally from doing next-token prediction. Specifically, an autoregressive model trained via teacher-forcing learns strong correlations between earlier and later tokens of the model's output. This can lead to a suboptimal mechanism if the model fails to model earlier tokens correctly, as it guarantees that the rest of the output sequence is incorrect. They demonstrate this pitfall using a graph search problem (Figure 3). The directed graph is a star, as such, every non-start node is connected only to one other node. A model that outputs the first node wrongly will also output the rest of the sequence incorrectly.

Unlike the copy task, the model fails on all samples equally — there is no distinction between easy or hard samples. So where is the unequal learning taking place?

In our synthetic copy task, the model learns tokens of different samples unequally. Here, the model is learning tokens at different *positions* unequally. Specifically, for all samples i, the first token $\log p_\theta(\mathbf{x}_{i,1} \mid \mathbf{x}_{i,0})$ is being learnt far slower than later ones. This is because the start node is the only node with a degree greater than one. For tokens later in the sequence, the model only sees a single answer in the context, as such, they are easier distributions to model and are learnt faster. This discrepancy is the source of the failure mode described in the original paper; here, we show that it is also an example of unequal learning.

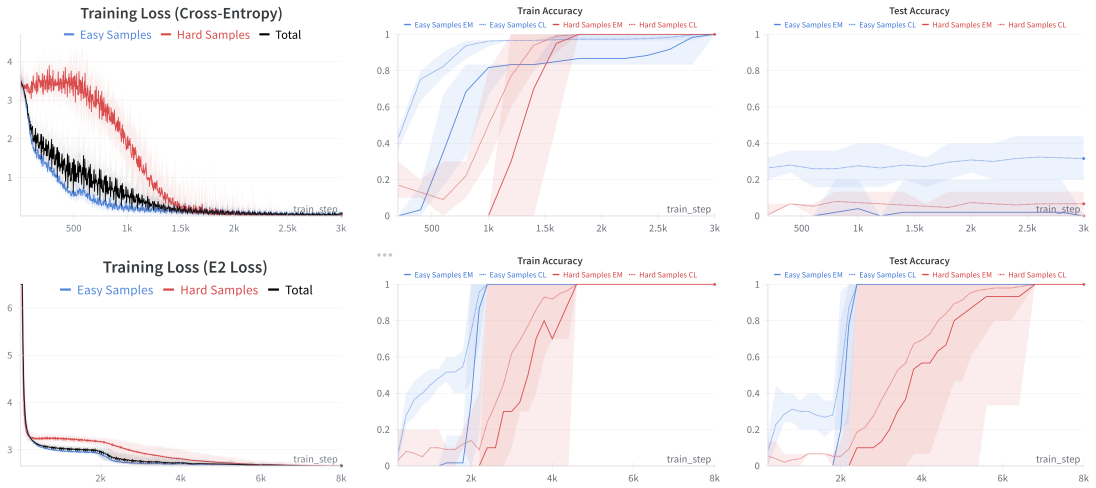


Figure 4: Evaluation plots (training loss, training accuracy, test accuracy) for $N_c = 5$ for different loss functions, showing metrics for easy and hard samples separately. We plot both exact-match and character-level accuracy. **(Top row)** Cross-entropy loss. **(Bottom row)** E2 loss with $\lambda = 10$ and 1000 warmup steps.

3.4 A way out?

Unequal learning stems from a data-generating process whereby optimizing a subset of tokens coincides with the likelihood-maximizing direction at the onset of training. Gradient descent ensures that the model then learns a suboptimal mechanism that drives the loss of certain tokens to zero faster than others.

For a retrieval task, unequal learning would not happen if the model had learnt the true mechanism instead. The true mechanism is equally correct on all samples and tokens, as such, a model in the process of learning correct features should expect the loss of all terms $\log p_\theta(\mathbf{x}_{i,j} \mid \mathbf{x}_{i,j})$ to decrease at equal rates. This intuition provides us a way out of the quagmire — at training time, we additionally bias all tokens to be learnt at the same rate. Doing so explicitly prevents the model from imperfect mechanisms, and hopefully tilts the model towards learning generalizable features.

4 Equal Learning

How can we promote the equal learning of terms in (1) during training? The idea is to constrain the model from learning specific terms too fast, especially during the early iterations of training. Intuitively, we want to ensure that all tokens contribute to gradient signals for as long as possible during training, which makes noise overfitting harder since the model would need to overfit more tokens. Our approach intervenes on the training loss by regularizing for this additional objective.

4.1 Explicit Regularization

Recall that a flexible model trained on cross-entropy loss is biased towards **maximizing margins** between the correct token and the wrong ones. For a given token $\mathbf{x} := \mathbf{x}_t$ with label $k := \mathbf{x}_{t+1}$, we see that

$$\begin{aligned}\mathcal{L}_{\text{ce}}(\theta, \mathbf{x}, k) &= -\log \frac{e^{f_k}}{\sum_{c=1}^{V} e^{f_c}} \\ &= g\Big(f_{c^*} - f_k + \log\Big[\sum_{c \neq k} e^{f_c - f_{c^*}}\Big]\Big) \quad (3)\end{aligned}$$

where $f_c := f_\theta(\mathbf{x})_c$ is the output logit of class c, c^* is the non-label class with the largest logit, and $g(z) = \log(1 + e^z)$ is a monotonically increasing function.

In the process of minimizing the cross-entropy loss, the model will learn to quickly maximize margins — in so doing, it can overfit specific tokens even in the early stages of training. As such, the question we pose becomes: how can we avoid learning a maximum-margin solution while still modeling the text correctly?

One alternative is to target a *specific but positive* margin by implementing an additive regularization term. We combine a term that wants arbitrarily large margins (the cross-entropy term) with one that wants to *shrink* the margin (the regularization term). As these two terms will compete, the model will need to learn a bounded, positive value of the margin in order to minimize the sum of both loss terms.

Whereas a model trained on cross-entropy alone can rapidly minimize loss by learning large margins, a model trained on the sum of both terms will find it harder to learn the correct solution, as taking gradient steps in either direction of the intended margin will cause loss to increase. To this end, we introduce a family of losses, termed E-LOSSES:

- **E1:** We penalize the mean-squared average margin between the correct class k and all other classes c.

$$\begin{aligned}\mathcal{L}_{E1}(\theta, \mathbf{x}, k) = \mathcal{L}_{\text{ce}}(\theta, \mathbf{x}, k) + \quad (4)\\ \lambda \cdot \log\Big(1 + \frac{1}{V-1}\sum_{c \neq k}(f_k - f_c)^2\Big)\end{aligned}$$

where λ is a hyperparameter and V is the number of classes, i.e. the vocabulary size.

- **E2:** We penalize variance of the margin:

$$\mathcal{L}_{E1}(\theta, \mathbf{x}, k) = \mathcal{L}_{\text{ce}}(\theta, \mathbf{x}, k) + \lambda \cdot \text{Var}_c(f_k - f_c) \tag{5}$$

- **E3:** Unlike E1and E2, we penalize mean-squared error of each output logit individually instead of penalizing the log-likelihood. This loss function implicitly combines both the cross-entropy and regularization objective into a single term:

$$L(f, y) = \|\mathbf{f} - \mathbf{v}\|^2 \tag{6}$$

where $\mathbf{v} = \left[-b, \ldots, b, \ldots\right]$ is the vector that is $-b$ everywhere except at the position corresponding to k, where it is b. b is a hyperparameter, as such, E3 trades off one kind of optimization difficulty (choosing λ) for another (choosing b).

All E-LOSSES work by introducing two competing incentives at training time: the cross-entropy term pushes the model towards maximizing margins whereas the regularization term *penalizes* the model for having large margins. Unlike the cross-entropy loss alone, where taking large gradient steps at the start of training allows the model to rapidly increase margins, our losses require the model to achieve specific margin values to achieve zero loss on any given token.

Which loss to use? We primarily use E1 in our experiments, which scales well to larger model and vocabulary sizes. We design E2 to show that there are other choice of regularization terms that can work so long as they shrink margins. On the other hand, E3 represents a vastly different design choice that targets a hard value for the margin rather than allow the model to learn a specific margin via soft regularization. This tradeoff makes E3 more effective on smaller vocabulary sizes, however, it is less effective as $|\mathcal{V}|$ increases since the $-b$ terms dominate the loss.

Connections to Uniform Margins Our choice to regularize margins was motivated by existing work in the field of out-of-distribution generalization. Notably, Puli et al. (2023) show that linear classifiers provably learn a shortcut mechanism assuming that enough noisy features exist and the scaling factor on the shortcut feature is large enough. They prove that enforcing uniform margins on all samples prevent shortcut learning from happening and demonstrate empirical results on the overparametrized setting (ResNet-50s with vision datasets).

Learning Rate Warmup In addition to using the E-LOSSES, we find empirically that limiting the learning rate at the start of training (i.e. setting a warmup schedule) is a simple but effective intervention. We hypothesize that smaller gradient steps prevent the model from reducing the loss of certain tokens to zero too early (stratification). Since all terms contribute to the gradient signal for more training iterations, the model is more likely to come across future batches of data that allows it to recover from the suboptimal mechanism (rectification).

We observe that controlling the learning rate alone is less effective than the E-LOSSES since the model is still biased towards learning the suboptimal mechanism — a smaller learning rate only affords the model more chances at rectification. As such, merely having a learning rate schedule is insufficient. However, E-LOSSES and learning rate warmup are not mutually exclusive interventions. On all our experiments below, our approach use an E-LOSS with learning rate warmup.

4.2 Empirical Validation on the Copy Task

We validate E-LEARN on the same copy task introduced in Section 3. Here we consider a more difficult setting, where $L_c = 5$ and $L_p = 2000$ (a signal-to-noise ratio of 400). The context length is 2013 (the length of a single string) and the embedding size is 1024. Figure 4 shows loss and accuracy plots for the standard cross-entropy loss as well as our E2 loss with 1000 warm-up steps. Due to space constraints, we show the other E-LOSSES in Appendix A.

Figure 4 show that while both losses achieve perfect exact-match (EM) and character-level (CL) accuracies on all training samples, only the E2 loss achieve perfect EM and CL accuracy on test samples at the end of training. As such, the key takeaway here is that **E-LEARN has learnt the correct copy mechanism instead of overfitting to training samples**. The training loss plots validate the conclusion that equal learning has taken place, as the E2 plot shows a far smaller gap between loss curves for the easy and hard samples. For the E2 loss, note also that the training accuracy of easy samples rises at a far slower rate in the first 1000 iterations, showing that the smaller learning rates has prevented the model from learning the easy samples too early.

Notably, the baseline cross-entropy model fails to generalize perfectly on even the *easy* test strings. Observe that the test CL accuracy is only $\sim$0.3 (average across 10 runs), in other words, the baseline model fails to learn even the wrong mechanism perfectly with this limited amount of data. However, E-LEARN generalizes perfectly on the same amount of data, showing that **equal learning is more sample-efficient**.

Ablations In considering how our approach can scale to real-life datasets and tasks, we ablate for two key factors. We summarize these findings below, and report them in greater detail in Appendix A:

- **Vocabulary size:** E1 and E2 scale up well to larger vocabulary sizes.

- **Separating the effect of E-LOSSES and learning rate warmup:** E-LOSSES are robust without needing warmup (with only a slight dip in perfor-

mance), however, warmup alone only provides a slight boost to performance.

Hyperparameter Selection E1 and E2 introduce an additional hyperparameter. We note that selecting this hyperparameter is not costly. For the synthetic task in Section 4.2, we set this hyperparameter through cross-validation by searching over powers of 2. For larger experiments in Section 5, we tried the value of 1, which generally works well in practice. Furthermore, it is pretty easy to see that the value of λ is wrong early in optimization by picking the largest value of λ where the negative log-likelihood term improves. We provide ablations for hyperparameters in Appendix A.

5 Experimental Results

In this section, we demonstrate the validity of our approach on two large-scale datasets: (i) the path-star graph task originally introduced in Bachmann and Nagarajan (2024), and (ii) two natural language texts on which we devise copy tasks susceptible to the same pitfalls introduced in Section 3. Experimental details can be found in Appendix B.

5.1 Path-Star Graph Problem

We consider the same graph task as Bachmann and Nagarajan (2024), which we described earlier in Section 3. We consider various graphs $G_{d,l}$ where d is the degree of the starting node (i.e. number of possible goal nodes) and l is the length og each path. Similar to the original authors, we train a GPT-Mini (Radford et al., 2019) from scratch. For each experiment, we report results across 5 different random initializations. We evaluate **accuracy** in predicting the goal node correctly. We do not compare to the methods (teacherless training and reverse-encoding) used in the original paper — we note that the solutions that Bachmann and Nagarajan (2024) propose are bespoke to the path-star graph problem and require knowing what the true solution is (namely, lookahead to the goal node). In contrast, E-LEARN are black-box loss functions that can be applied to any autoregressive task without knowing what the imperfect or true mechanisms are.

Table 1 displays the results. We can see that **E-LEARN performs significantly better than the baseline model trained on cross-entropy loss**. Notably, on $G_{2,3}$, the best-performing run of the cross-entropy model still performs worse than the worst-performing run of the E1 loss.

5.2 Natural Language Datasets

We devise a copy task on two natural language texts. Each train or test sample consists of a single sentence of the text, modified as follows: For each sentence, we assign a "label", which is a single word, related to the sentence (e.g. the title of the book that sentence originates from), that is injected into the sentence at two possible places: (1) the start of each sentence, and (2) in the middle of a sentence. The task is to output this label at the end of the sentence. Easy samples correspond to those containing this word both at the start and the middle of the sentence, whereas hard samples correspond to those without the word at the start.

	Accuracy
$G_{2,3}$	
Cross-Entropy	0.289 *(0.123, 0.472)*
E1 Loss with LR warm-up	**0.655** *(0.486, 0.976)*
$G_{5,3}$	
Cross-Entropy	0.001 *(0.000, 0.005)*
E1 Loss with LR warm-up	**0.117** *(0.094, 0.139)*

Table 1: Results on the path-star graph search problem. We report both the mean accuracy (across 5 independent runs) as well as the maximum and minimum accuracy *(in brackets)*.

	Accuracy
King James Bible	
Cross-Entropy	0.863
E1 Loss with LR warm-up	**0.981**
Complete Works of Shakespeare	
Cross-Entropy	0.703
E1 Loss with LR warm-up	**0.766**

Table 2: Results of a copy task on two natural language texts: the King James Bible and the Complete Works of Shakespeare.

Table 2 displays the results on pretrained GPT-2. In both datasets, the E1 loss function outperforms standard cross-entropy. These results highlight that our approach is not only **viable on natural language text, but also that it is useful for fine-tuning on a model already pretrained on natural language.** In particular, we note that both texts that we use are part of multiple language corpora and highly likely to be in GPT-2's training dataset (Radford et al., 2019).

6 Discussion

As LLMs gain widespread traction and usage in society, understanding how and when they can fail is crucial to safe and robust performance. We believe that our work: (i) identifies an important pitfall that modern transformer-based LLMs are susceptible to — namely, overfitting due to unequal learning, (ii) provides empirical evidence for this problem, and (iii) proposes a working solution in the form of new loss functions.

Limitations

The key limitation of our approach is that our loss functions are more challenging to optimize well. Adding a regularization term involves a hyperparameter sweep on a heldout set in order to find the correct values of λ and b. It is also more computationally intensive because we need to compute an additional term in the loss function,

although the margin is relatively cheap to compute.

Potential Risks We do not believe there are any notable risks associated with our work besides risks that are associated with LLMs in general and their potential for misuse.

Acknowledgements

This work was partly supported by the NIH/NHLBI Award R01HL148248, NSF Award 1922658 NRT-HDR: FUTURE Foundations, Translation, and Responsibility for Data Science, NSF CAREER Award 2145542, ONR N00014-23-1-2634, NSF Award 2404476 and Apple. This work was also supported by IITP with a grant funded by the MSIT of the Republic of Korea in connection with the Global AI Frontier Lab International Collaborative Research (No. RS-2024-00469482 + RS-2024-00509279).

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Madhu S Advani, Andrew M Saxe, and Haim Sompolinsky. 2020. High-dimensional dynamics of generalization error in neural networks. *Neural Networks*, 132:428–446.

Alex Andonian, Quentin Anthony, Stella Biderman, Sid Black, Preetham Gali, Leo Gao, Eric Hallahan, Josh Levy-Kramer, Connor Leahy, Lucas Nestler, Kip Parker, Michael Pieler, Jason Phang, Shivanshu Purohit, Hailey Schoelkopf, Dashiell Stander, Tri Songz, Curt Tigges, Benjamin Thérien, Phil Wang, and Samuel Weinbach. 2023. GPT-NeoX: Large Scale Autoregressive Language Modeling in PyTorch.

Gregor Bachmann and Vaishnavh Nagarajan. 2024. The pitfalls of next-token prediction. In *Forty-first International Conference on Machine Learning*.

Nicholas Baker, Hongjing Lu, Gennady Erlikhman, and Philip J Kellman. 2018. Deep convolutional networks do not classify based on global object shape. *PLoS computational biology*, 14(12):e1006613.

Satwik Bhattamishra, Arkil Patel, Varun Kanade, and Phil Blunsom. 2022. Simplicity bias in transformers and their ability to learn sparse boolean functions. *arXiv preprint arXiv:2211.12316*.

Alberto Bietti, Vivien Cabannes, Diane Bouchacourt, Herve Jegou, and Leon Bottou. 2024. Birth of a transformer: A memory viewpoint. *Advances in Neural Information Processing Systems*, 36.

Yuan Cao, Zhiying Fang, Yue Wu, Ding-Xuan Zhou, and Quanquan Gu. 2019. Towards understanding the spectral bias of deep learning. *arXiv preprint arXiv:1912.01198*.

Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, et al. 2021. Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2633–2650.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, et al. 2021. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 1(1):12.

Gauthier Gidel, Francis Bach, and Simon Lacoste-Julien. 2019. Implicit regularization of discrete gradient dynamics in linear neural networks. *Advances in Neural Information Processing Systems*, 32.

Michael Hanna, Ollie Liu, and Alexandre Variengien. 2024. How does gpt-2 compute greater-than?: Interpreting mathematical abilities in a pre-trained language model. *Advances in Neural Information Processing Systems*, 36.

Jason Jo and Yoshua Bengio. 2017. Measuring the tendency of cnns to learn surface statistical regularities. *arXiv preprint arXiv:1711.11561*.

Miyoung Ko, Jinhyuk Lee, Hyunjae Kim, Gangwoo Kim, and Jaewoo Kang. 2020. Look at the first sentence: Position bias in question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1109–1121, Online. Association for Computational Linguistics.

Jinhyuk Lee, Anthony Chen, Zhuyun Dai, Dheeru Dua, Devendra Singh Sachan, Michael Boratko, Yi Luan, Sébastien MR Arnold, Vincent Perot, Siddharth Dalmia, et al. 2024. Can long-context language models subsume retrieval, rag, sql, and more? *arXiv preprint arXiv:2406.13121*.

Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. 2018. Visualizing the loss landscape of neural nets. *Advances in neural information processing systems*, 31.

Andrew Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, pages 142–150.

Vaishnavh Nagarajan, Anders Andreassen, and Behnam Neyshabur. 2021. Understanding the failure modes of out-of-distribution generalization. In *International Conference on Learning Representations*.

Ramesh Nallapati, Bowen Zhou, Caglar Gulcehre, Bing Xiang, et al. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023*.

Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. 2023. Progress measures for grokking via mechanistic interpretability. *arXiv preprint arXiv:2301.05217*.

Shashi Narayan, Shay B Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. *arXiv preprint arXiv:1808.08745*.

Alethea Power, Yuri Burda, Harri Edwards, Igor Babuschkin, and Vedant Misra. 2022. Grokking: Generalization beyond overfitting on small algorithmic datasets. *arXiv preprint arXiv:2201.02177*.

Aahlad Manas Puli, Lily Zhang, Yoav Wald, and Rajesh Ranganath. 2023. Don't blame dataset shift! shortcut learning due to gradients and cross entropy. *Advances in Neural Information Processing Systems*, 36:71874–71910.

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

Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. 2019. On the spectral bias of neural networks. In *International conference on machine learning*, pages 5301–5310. PMLR.

P Rajpurkar. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Basri Ronen, David Jacobs, Yoni Kasten, and Shira Kritchman. 2019. The convergence rate of neural networks for learned functions of different frequencies. *Advances in Neural Information Processing Systems*, 32.

Harshay Shah, Kaustav Tamuly, Aditi Raghunathan, Prateek Jain, and Praneeth Netrapalli. 2020. The pitfalls of simplicity bias in neural networks. *arXiv preprint arXiv:2006.07710*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Daniel Soudry, Elad Hoffer, Mor Shpigel Nacson, Suriya Gunasekar, and Nathan Srebro. 2018. The implicit bias of gradient descent on separable data. *The Journal of Machine Learning Research*, 19(1):2822–2878.

Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, et al. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*.

Bhavya Vasudeva, Deqing Fu, Tianyi Zhou, Elliott Kau, Youqi Huang, and Vatsal Sharan. 2024. Simplicity bias of transformers to learn low sensitivity functions. *arXiv preprint arXiv:2403.06925*.

A Vaswani. 2017. Attention is all you need. *Advances in Neural Information Processing Systems*.

Zhi-Qin John Xu, Yaoyu Zhang, Tao Luo, Yanyang Xiao, and Zheng Ma. 2019. Frequency principle: Fourier analysis sheds light on deep neural networks. *arXiv preprint arXiv:1901.06523*.

Greg Yang and Hadi Salman. 2019. A fine-grained spectral perspective on neural networks. *arXiv preprint arXiv:1907.10599*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*.

Chulhee Yun, Srinadh Bhojanapalli, Ankit Singh Rawat, Sashank J Reddi, and Sanjiv Kumar. 2019. Are transformers universal approximators of sequence-to-sequence functions? *arXiv preprint arXiv:1912.10077*.

A Additional Results

A.1 Copy Task in Section 4

Figure 5 extends Figure 4 of Section 4 and show the training plots for the E1 and E3 losses. Similar to the E2 loss reported in the main paper, both of these loss functions generalize well on test samples, achieving high accuracy (on average, across 10 initialization runs) on test hard samples. The loss curves also show that both of these losses achieve equal learning across all samples.

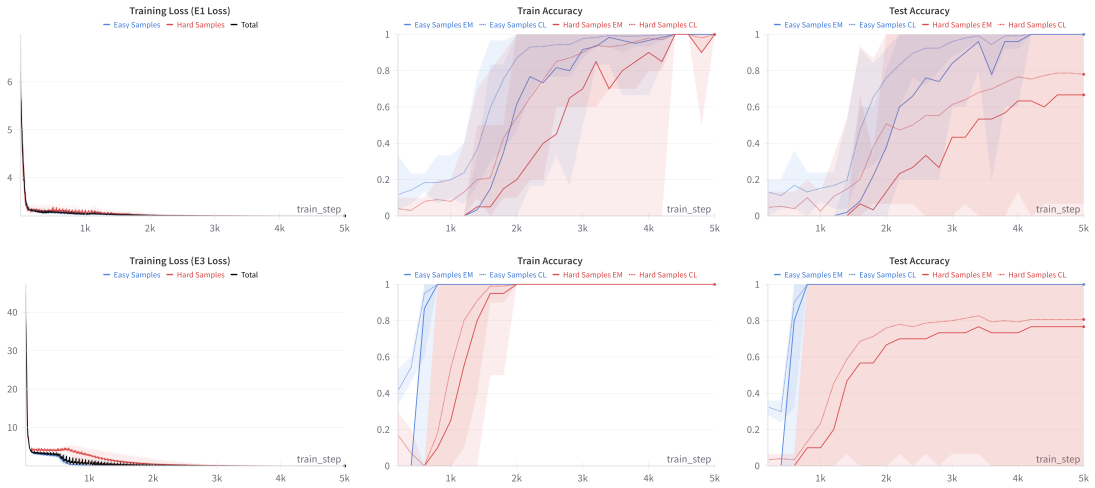


Figure 5: Evaluation plots (training loss, training accuracy, test accuracy) for $N_c = 5$ for different loss functions, showing metrics for easy and hard samples separately. We plot both exact-match and character-level accuracy. **(Top row)** E1 loss with 1000 warm-up steps. **(Bottom row)** E2 loss with 500 warmup steps.

A.2 Ablations in Section 4

Table 3 shows the results of the copy task in Section 4 as we ablate for various factors. We report only CL accuracy on the easy test samples (averge of 10 random initializations) since the model performance on this subset is the key indicator of generalization.

	CL Accuracy on Easy Test Samples
E1 Loss with no LR warm-up and $\lambda = 2$	0.08
E1 Loss with no LR warm-up and $\lambda = 4$	0.593
E1 Loss with LR warm-up and $\lambda = 4$	0.780
E1 Loss with LR warm-up and $\lambda = 1$ and $V = 500$	1.00
E1 Loss with LR warm-up and $\lambda = 8$	0.693
Only LR warm-up	0.246

Table 3: Hyperparameter sweep for the different E-LOSSES on the copy task.

Hyperparameter Sensitivity For the E1 loss, we can see that a minimum value of λ is generally desired to ensure that the regularization term is strong enough.

Vocabulary Size Our approach E-LEARN is generally robust as we scale up the vocabulary size. For example, the E1 achieves perfect accuracy as we increase $V = 26$ to $V = 500$. However, we note that we need to scale training data as well. From $V = 26$ to $V = 500$, we also increased data by twice as much.

Separating E-LOSSES and learning rate warm-up We see that E1 is generally robust without needing learning rate warm-up, even though performance degrades slightly. Having a learning rate schedule alone, without the regularization, is only slightly effective.

B Experimental Details

B.1 Padded-Shifted Copy Task

We train a 4-layer GPT-NeoX from scratch with 4-head attention. For each experimental setting, we report results across 10 different random initializations. We report these two metrics on both the train and test sets, and on the easy and hard subsets separately. The training size is 400 and the test size is 1000.

B.2 Path-Star Graph Problem

We train a GPT-Mini from scratch. For each experimental setting, we report results across 5 different random initializations. The training size is 200000 and the test size is 20000, similar to the original paper.