Same Pre-training Loss, Better Downstream: Implicit Bias Matters for Language Models

Hong Liu [1] **Sang Michael Xie** [1] **Zhiyuan Li** [1] **Tengyu Ma** [1]

Abstract

Language modeling on large-scale datasets improves performance of various downstream tasks. The *validation* pre-training loss is often used as the evaluation metric for language models since the pre-training loss tends to be well-correlated with downstream performance (which is itself hard to evaluate comprehensively). Contrary to the conventional wisdom, this paper shows that 1) pre-training loss cannot fully explain downstream performance and 2) flatness of the model is well-correlated with downstream performance where pre-training loss is not. We identify three ways to produce models with the same pre-training loss but different downstream performance: continue pre-training after convergence, increasing the model size, and changing the pre-training algorithms. These experiments demonstrate the existence of implicit bias of pre-training algorithms—among models with the same minimal pre-training loss, they implicitly prefer more transferable ones. Toward understanding this implicit bias, we prove that SGD with standard mini-batch noise implicitly prefers flatter minima of pre-training loss in language models, and empirically observe a strong correlation between flatness (measured by trace of Hessian) and downstream performance among models with the same pre-training loss. We also prove in a synthetic language setting that among models with the minimal pre-training loss, the flattest model transfers to downstream tasks.

1. Introduction

Large language models (LLMs) pre-trained on internet-scale data have improved performance on a wide array of downstream tasks (Devlin et al., 2018; Yang et al., 2019; Radford et al., 2019; Raffel et al., 2020; Brown et al., 2020). These models are trained with a language modeling pre-training loss to "fill in the blanks"—either predicting the next token/word (autoregressive language modeling loss, or perplexity) or masked tokens (masked language modeling (MLM) loss).

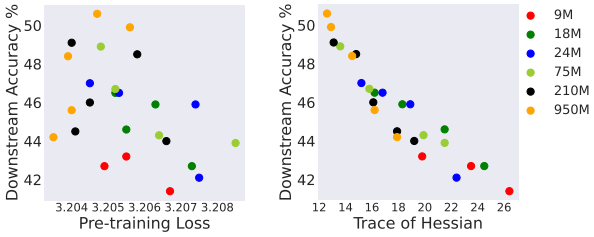


Figure 1. **The relationship between pre-training loss, flatness, and linear probe accuracy on PCFG Task B.** Each dot corresponds to a model. Dots with the same color are models of the same size but trained with different number of steps. All the models have quite similar pre-training validation losses between 3.204-3.208. The trace of Hessian of the pre-training loss correlates much better with the downstream performance than the pre-training loss.

In common practice, the *validation* pre-training loss is used to monitor the training process (Brown et al., 2020; Zhang et al., 2022a) and compare different models since the pre-training loss is generally strongly correlated with downstream performance (Hernandez et al., 2021). Moreover, theoretical works on understanding LLMs also focus on how the pre-training loss affects downstream performance. Saunshi et al. (2020); Wei et al. (2021b); Xie et al. (2021) show that good pre-training loss, or fitting the language modeling conditional probability well, is a main reason for downstream success of LLMs. Their analyses generally treat the language models as blackboxes and do not take into account *how* the models represent the conditional probability.

In this paper, we question the conventional wisdom on the correlation between the validation pre-training loss and downstream performance for language modeling. Recent works have demonstrated that models with different architectures may have the same pre-training loss but different performance (Saunshi et al., 2022; Tay et al., 2021). Due to the expressivity of modern neural nets, many parameter configurations even within the *same* architecture can still have the same pre-training loss. A priori, it is unclear why all these config-

*Equal contribution [1]Department of Computer Science, Stanford Univerisity. Correspondence to: Hong Liu <hliu99@stanford.edu>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

urations should have the same downstream performance.

We find that different parameter configurations with the same pre-training loss can indeed have different downstream performance. Concretely, with synthetic and real datasets, we find three situations that demonstrate such a phenomenon:

- Models pre-trained with standard optimizers and regularization have better downstream performance than models pre-trained with adversarial algorithms, improper regularization, or a hypothetical "look-up table" model that outputs the *exact* conditional probabilities, *even when they all have with the same pre-training loss.*
- Even after the pre-training loss *converges*, models at a later time step still tend to perform better.
- Larger models perform better downstream than smaller models even when they have the *same pre-training loss*.

In each of the first two cases above, we find models with the *same* pre-training loss and the *same* architecture; but some have better performance than others. They only differ by the pre-training algorithms. Therefore, this suggests the pre-training algorithms have an *implicit bias* toward certain types of models—standard algorithms with more training steps biases towards parameter configurations that transfer better to downstream tasks. The third case has a more subtle but similar interpretation. There exists a hypothetical large model that represents the smaller model with worse downstream performance (by filling zeros in the weights or replicating the weights of the smaller model). The training algorithm on the large architecture could have chosen it, but did not. This suggests the algorithm has an implicit bias against the hypothetical model (which has an equally good pre-training loss).

In supervised settings, optimizers are known to have an implicit bias toward selecting generalizable models among all models with small *empirical* loss. Recently, Damian et al. (2021); Li et al. (2021) suggest that, among all the minimizers of the empirical loss, stochastic optimizers implicitly prefer *flatter* ones. Past works have suggested theoretically and empirically that encouraging flatness tends to improve generalization (Keskar et al., 2016; Dziugaite & Roy, 2017; Neyshabur et al., 2017; Jastrzębski et al., 2017; Jiang et al., 2019; Wei & Ma, 2019a;b; Wu et al., 2020; Foret et al., 2021; Norton & Royset, 2021; Zheng et al., 2021) (even though flatness is not a necessary condition for generalization (Dinh et al., 2017)).

However, the role of implicit bias in self-supervised learning has not been studied. We also claim that its role is conceptually different. Unlike in supervised learning, the gap between empirical and population self-supervised losses is typically small in self-supervised learning, and thus implicit bias does *not* contribute to bridging this gap. Instead, the implicit bias selects models that transfer better to downstream tasks.

Why do pre-training algorithms bias toward some type of models? In Section 3, we provide a first-cut theoretical analysis of the implicit bias in language modeling. Fortunately, despite the conceptual differences, mathematical tools from supervised settings can be straightforwardly adapted to language modeling. We prove that mini-batch SGD prefers flatter minima of population pre-training loss. Interestingly, we obtain cleaner theoretical results for the standard mini-batch SGD, without the artificial label noise introduced in prior works (Damian et al., 2021; Li et al., 2021), partly because the mini-batch noise for LLMs does not vanish even at convergence.

We corroborate our theory with empirical evidence in Section 4. We show that for models with the same pre-training loss in the three situations above, the trace of Hessian of the pre-training loss strongly correlates with the downstream performance (See Figure 1). In addition, removing regularization from standard pre-training algorithms makes both flatness and downstream performance worse, while adding explicit flatness regularization can recover both both of them.

Finally, to complement the theory and experiments, we rigorously formalize the connection between flatness and downstream performance in a simplified Dyck language setting in Section 5. We prove that although there are many models with good MLM pre-training loss; among them, the flattest model learns the most useful features for downstream tasks.

2. Implicit Bias Affects Downstream Accuracy

In this section, we investigate the relationship between pre-training loss and downstream performance. We find that models with the same pre-training loss but different training procedures can have different downstream performance. Code is provided in `https://github.com/Liuhong99/implicitbiasmlmcode`.

2.1. Formulations

Masked language modeling. Consider a vocabulary $W = \{0, 1, \dots, c\}$, where 0 is a special token for the mask. Let $x = [x_1, \dots, x_T]$ denote the input sequence, and $x_{-t} = [x_1, \dots, x_{t-1}, 0, x_{t+1}, \dots, x_T]$ denote the masked sentence, where t is sampled uniformly randomly and independently.[1] The MLM conditional probability refers to the probability of x_t given the rest of the sequence $\Pr(x_t \,|\, x_{-t})$. We use $\Pr(\cdot \,|\, x_{-t})$ to denote the c-dimensional probability vector $\Pr(\cdot \,|\, x_{-t}) := [\Pr(x_t = 1 \,|\, x_{-t}), \dots, \Pr(x_t = c \,|\, x_{-t})] \in \mathbb{R}^c$. In MLM pre-training, the model outputs the predicted MLM conditional probability vector $f_\theta(x_{-t}) \in \mathbb{R}^c$. The model is trained to predict the masked token x_t given the rest of the sentence x_{-t} with cross entropy loss,

[1] For simplicity, in the formulation we mask one token per sentence. Empirically, we use standard 15% mask rate on real datasets.

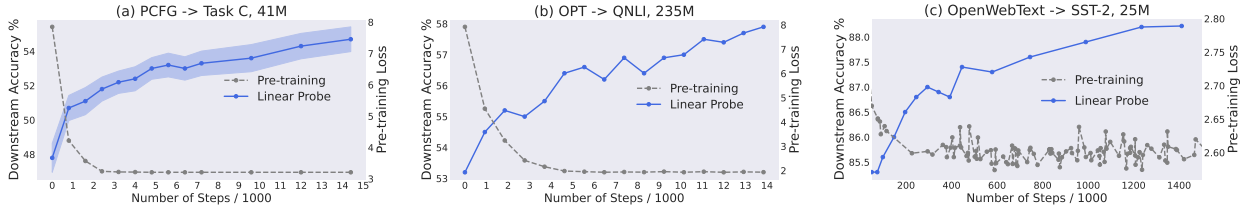


Figure 2. Models at a later time step perform better, even after the pre-training loss converges. (a) A model with 41M parameters pre-trained on the PCFG-generated dataset, and evaluated on task C. (b) A model with 235M parameters pre-trained on the OPT-generated dataset, and evaluated on QNLI. (c) A model with 25M parameters pre-trained on the OpenWebText, and evaluated on SST-2. Note that the pre-training loss approaches its minimal value (3.196 for the PCFG and 1.865 for OPT) in (a) and (b) as we increase the number of steps.

$L(\theta) = \mathbb{E}_{x,t}[\ell(f_\theta(x_{-t}), x_t)] = \mathbb{E}_{x,t}[-\log([f_\theta(x_{-t})]_{x_t})]$.

Downstream evaluation. The language model f_θ is composed of a feature extractor h_ψ, which outputs a sequence of contextual representations, and a linear classifier that outputs the conditional probability at every position. On downstream tasks, we use a randomly initialized g_ϕ on top of the pre-trained h_ψ. In fine-tuning, both g_ϕ and h_ψ are trained, while in linear probe, only g_ϕ is updated. For fine-tuning, we use the contextual representations of the `cls` token. For linear probe, we concatenate the contextual representations of all the tokens together.

Saturation regime. In order to factor out the impact of the pre-training loss, we introduce the saturation regime, a family of models with the same (optimal) pre-training loss. We say a model is in the saturation regime if its output equals the *true* conditional probability, $f_\theta(x_{-t}) = \Pr(\cdot \mid x_{-t})$, and the MLM loss equals the entropy of the true conditional probability $L(\theta) = \mathbb{E}_{x,t}[-\log(\Pr(x_t \mid x_{-t}))] = \frac{1}{T}\sum_{t=1}^{T} H(x_t \mid x_{-t})$, which is also the lower bound of the pre-training loss. Thus, *all* models in the saturation regime have the same optimal pre-training loss. When the architectures are sufficiently expressive, multiple parameter configurations can compute the true conditional probability and thus be in the saturation regime. As will be shown in Section 2.3, interestingly, models in the saturation regime can still have varying downstream performance (which is not caused by the pre-training loss because it is the same.) Perhaps more interestingly, this phenomenon also holds with linear probing on contextualized presentations instead of fine-tuning. Thus, even though the predicted conditional probabilities of two models are the same (and correct), the contextualized representations can behave differently.

We also conduct experiments on real large-scale data which are typically not in the saturation regime due to limited training time and architecture size, which validate our findings in the setup that is more representative of real-world use.

2.2. Experimental Setup

We design controlled experiments to study the correlation between pre-training loss and downstream performance. In particular, we find a set of models with almost the same pre-training loss. We effectively use the same architecture family so that the main difference between the models only stems from pre-training algorithms. More details and additional results are provided in Section A.

Datasets. We introduce three datasets produced by generative models. With the knowledge of the true generative models, we can compute the true conditional probability and scale up the models until they approach the saturation regime to ensure they have almost the same pre-training loss. We also consider two real-world large language modeling corpora, with which we can demonstrate our findings in practical setups.

1) *PCFG*-generated dataset. PCFG (Chomsky, 1956) generates sentences with probabilistic trees and is widely used to understand natural language (Roark & Bacchiani, 2003; Kim et al., 2019). The symbols in the parse trees are intrinsic quantities of the sentences such as syntax. We design three downstream tasks A, B, and C to classify symbols at different positions of the parse trees.
2) *HMM-generated dataset.* HMM samples hidden variables from the transition probabilities and tokens from the emission probabilities, which is used to analyze the properties of pre-trained language models (Wei et al., 2021b; Xie et al., 2021). The downstream task is classifying hidden variables in the sentences. We use task-k to refer to classifying the k-th hidden variable.
3) *OPT-generated dataset.* OPT is an autoregressive language model (Zhang et al., 2022a). Starting from the `bos` token, we sample tokens from the output of OPT. For computational feasibility we only allow the top-2000 frequent tokens in the OPT vocabulary. We use QNLI and SST-2 from GLUE (Wang et al., 2018) as downstream tasks.
4) *OpenWebText.* OpenWebText (Gokaslan et al., 2019) is an open version of the large language corpus used to train GPT-2 (Radford et al., 2019). It contains 40GB of text.
5) *BookCorpus.* BookCorpus (Zhu et al., 2015) is a 4GB collection of novel books used to train BERT (Devlin et al., 2018). We consider GLUE (Wang et al., 2018) as downstream tasks for the two real-world datasets.

Note that the true conditional probability can be computed efficiently for the three synthetic datasets given the generative

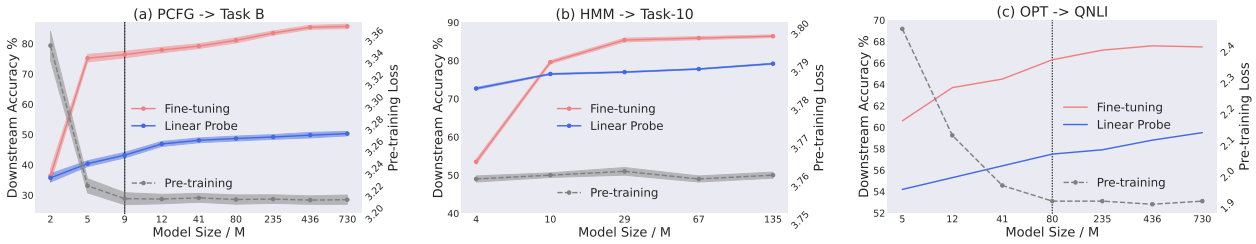


Figure 3. Larger models perform better downstream than smaller models, even with almost the same pre-training loss. (a) Pre-train on the PCFG-generated dataset and evaluate on task B. (b) Pre-train on the HMM-generated dataset and evaluate on task-10. (c) Pre-train on the OPT-generated dataset and evaluate on QNLI. For PCFG- and OPT-generated datasets, the vertical dashed line indicates the model size where the pre-training loss saturates. For HMM, the smallest model is already close to the minimal loss. See Section 2 for details.

models. For PCFG- and HMM-generated datasets, we can compute the true conditional probability with the inside algorithm (Lari & Young, 1990) and the Viterbi algorithm (Forney, 1973), respectively. For the OPT-generated dataset, we can calculate the MLM conditional probability from the joint probability, and the joint probability can be decomposed into the autoregressive conditional probability of the OPT model.

Models and algorithms. For PCFG, OPT and real language datasets, we use transformers (Vaswani et al., 2017) from 2M to 950M. For the HMM-generated dataset, we use LSTM (Hochreiter & Schmidhuber, 1997b) from 10M to 135M. All the models are pre-trained with AdamW following the protocol of Izsak et al. (2021) with 0.1 dropout and 0.01 weight decay. Besides varying training steps and model sizes, we also consider two other approaches that produce models with the same pre-training loss. First, inspired by Liu et al. (2020); Raghu et al. (2021), we pre-train models with an additional meta-learning objective that messes up the downstream performance while keeping the same MLM loss. The second approach is to virtually implement a model that produces the true conditional probabilities as the representations of the sentences, and we call this model a *look-up table*. Note that such a model always has a perfect pre-training loss, and can be implemented by a sufficiently large transformer because transformers are universal approximators (Yun et al., 2019; Wei et al., 2021a). Since we know the true conditional probability from generative models, we can evaluate the downstream performance of the "look-up table" by linear probe on top of the concatenated conditional probabilities, without explicitly constructing parameters of the transformer that implements the look-up table. On BookCorpus, we compare models pre-trained with standard 0.1 dropout and 0.01 weight decay to models without these regularizations. Additionally, we evaluate models pre-trained with sharpness-aware minimization (SAM) (Foret et al., 2020), which explicitly encourages flatness.

2.3. Role of implicit bias on downstream accuracy

We compare the downstream performance of models with the same pre-training loss in the following situations: (1) training for different numbers of steps after the pre-training loss converges, (2) pre-training using normal algorithms with regularization vs. pre-training using adversarial algorithms or without proper regularization, and (3) using different model sizes. We plot the pre-training loss and downstream accuracy of the models we obtain on PCFG datasets in Figure 1 (Left). Our results demonstrate examples of implicit biases that affect downstream accuracy when pre-training loss *fails* to correlate with downstream accuracy.

Training longer improves downstream performance despite obtaining the optimal pre-training loss. In Figure 2, we plot the validation pre-training loss and the downstream performance of different models checkpoints during one pre-training run. After the pre-training loss converges, although the pre-training loss does not improve, the downstream accuracy continues increasing.

Table 1. Comparison of different pre-training algorithms.

PCFG	Pre-training	Task A %	Task B %
AdamW	3.204	89.9 ±0.3	49.2±0.8
Adversarial	3.206	83.1 ±0.6	42.3±1.5
Lookup table	3.196	71.2	39.7
BookCorpus	**Pre-training**	**QNLI %**	**SST-2 %**
AdamW (w/ dropout, WD)	1.85	84.4	90.8
-dropout-WD	1.85	83.0	89.5
-dropout-WD+SAM	1.89	84.4	90.1

Natural pre-training algorithms have better implicit bias than adversarial algorithms despite reaching the optimal pre-training loss. In Table 1, we evaluate the 235M transformers on PCFG tasks A and B with different pre-training algorithms. Although the adversarially trained transformer has almost the same pre-training loss as the AdamW trained 235M transformer, it is more than 6% worse than the AdamW model. Thus, these two models have very different downstream accuracy with the same pre-training loss and architecture. Similarly, the transformer model that implements a lookup table has perfect pre-training loss, but it performs worse than all other models in Figure 3(a) on task B.

Regularization matters for downstream accuracy and

SAM can serve as a substitute. On BookCorpus, we train a 330M-parameter transformer with and without explicit regularization (dropout and weight decay). Both models achieve the same *validation* pre-training loss, but the model without regularization performs worse after fine-tuning on QNLI and SST-2 (Table 1). SAM recovers the decrease in downstream performance, even with a slightly worse pre-training loss.

Larger models are better than smaller models even with the same pretraining loss. In Figure 3, we plot the pre-training loss and the downstream performance of models with different sizes. As we increase the model size, the pre-training loss approaches the entropy of the true conditional probability. With the same pre-training loss, scaling up the models improves linear probe performance by 6.9%, 4.5%, and 2.0%, on PCFG, HMM, and OPT generated data, respectively.

The experiments indicate that for models with the same architecture family and the same pre-training loss, the choice of training algorithms, model sizes, and the number of steps that the optimizer works affect downstream performance. This indicates that *implicit bias* of the pre-training algorithms plays a crucial role towards choosing more transferable models among those with the same pre-training loss and architecture.

3. SGD Prefers Flatter Minima in Language Modeling

Recent works (Damian et al., 2021; Li et al., 2021) show that SGD with label noise prefers converging to the flattest local minima in the supervised setting. In this section, we extend these results to the language modeling setting. Interestingly, the results here apply to the standard mini-batch SGD without the artificial label noise in Damian et al. (2021); Li et al. (2021) because SGD on language modeling loss (which is a cross-entropy loss) with sufficient data still has non-vanishing noise around the local minima, and thus the artificial label noise is no longer needed.

Concretely, we analyze SGD on the population cross-entropy loss $L(\theta) = \mathbb{E}_{x,t}[-\log([f_\theta(x_{-t})]_{x_t})]$ with freshly sampled data at every iteration, because, as argued, the difference between empirical and population pre-training loss is not our focus. For simplicity, we present the results for batch size $= 1$, though they can be generalized to arbitrary batch size (see discussion below Theorem 3.3). Let η be the learning rate and let θ_k^η denote the parameter at step k. We drop the superscript η when there is no ambiguity. We will show that the implicit bias kicks in when SGD reaches a global minimizer—it drives the iterate towards global minimizers with smaller trace of Hessian. For simplicity of demonstration, we analyze the process starting from a global minimizer $\overline{\theta}$, *i.e.*, we assume that $\theta_0^\eta = \overline{\theta}$ (for all η). At each iteration k, we get a fresh sample (x,t), where x is a sentence and t is the position of the masked token, and update the parameter θ by $\theta_{k+1} = \theta_k - \eta \nabla_\theta \ell(f_{\theta_k}(x_{-t}), x_t)$. We assume the network is sufficiently expressive such that there are many fundamentally different global minimizers of the pre-training loss L. As a (non-trivial) regularity condition, following prior works (Fehrman et al., 2020; Li et al., 2021; Arora et al., 2022), we also assume that the minimizers of the loss function L are connected and form a smooth manifold.

Assumption 3.1. *Assume that the loss L is a $\mathcal{C}^4$-smooth function and that $\min_{\theta\in\mathbb{R}^d} L(\theta) = \mathbb{E}_{x,t}[-\log(\Pr(x_t \mid x_{-t}))]$. We also assume that the set of global minimizers, Γ, is a $(d-M)$-dimensional $\mathcal{C}^2$-submanifold of $\mathbb{R}^d$ for some integer $1 \le M \le d$, where for all $\theta \in \Gamma$, $\operatorname{rank}\big(\nabla^2 L(\theta)\big) = M$.*

A key observation for language models is that even if the model reaches the saturation regime, that is, the model reaches a point on the manifold Γ of the minimizers, the optimization process still has non-vanishing gradient noise, because the cross-entropy loss is typically *non-zero* at the global minimizers and thus the stochastic gradient variance is also non-zero.[2] Therefore, the dynamics of SGD do not completely stop; instead, the iterate oscillates around the manifold Γ. It turns out that this oscillation in turn encourages the parameter to move in a certain direction along the manifold, determined by the covariance structure of the stochastic gradient. The following lemma shows that the covariance of stochastic gradient for language models in the saturation regime has a favorable property, *i.e.*, it is equal to the Hessian of pre-training loss.

Lemma 3.2 (Bartlett identity)**.** *Suppose $\Sigma(\theta)$ is the covariance of the stochastic gradient at θ, that is, $\Sigma(\theta) = \mathbb{E}_{t,x}\left[\nabla_\theta \log[f_\theta(x_{-t})]_{x_t} (\nabla_\theta \log[f_\theta(x_{-t})]_{x_t})^\top\right] - \nabla L(\theta)^\top \nabla L(\theta)$. For any $\theta \in \Gamma$, we have $\Sigma(\theta) = \nabla^2 L(\theta)$.*

Though we give a proof of the lemma in Appendix F for completeness, the formula holds for the MLE loss of any well-specified probabilistic models at a global minimizer, and both the gradient covariance and the Hessian equal to the Fisher information matrix.

With Lemma 3.2, we can invoke Corollary 5.2 of Li et al. (2021) to derive the following theorem which says that SGD will locally decrease the trace of Hessian along the solution of ordinary differential equation (1) defined below.

$$\mathrm{d}\hat{\theta}(t) = -1/4 \cdot \nabla_\Gamma \operatorname{Tr}[\nabla^2 L(\hat{\theta}(t))]\mathrm{d}t, \quad \hat{\theta}(0) = \overline{\theta} \qquad (1)$$

where $\nabla_\Gamma = P_\Gamma^\perp \nabla$ is the Riemannian gradient on manifold Γ, or just the ordinary gradient projected back to the tangent space of Γ at θ. In other words, the ODE (1) is essentially projected gradient descent with loss function $\operatorname{Tr}[\nabla^2 L(\theta)]$, the constraint set Γ, and infinitesimal learning rate. We show

[2] This is in contrast with supervised setting where the empirical 0-1 or cross-entropy loss can achieve zero and consequently the mini-batch noise vanishes. Such a difference enables us to prove cleaner results without the label noise than the supervised setting.

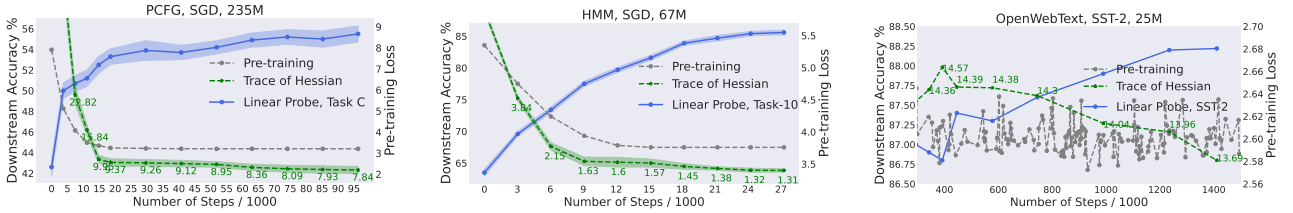


Figure 4. The trace of Hessian correlates with downstream performance for model at different number of steps after the pre-training loss converges. Left: A 235M model pre-trained on the PCFG-generated dataset, and evaluated on task C. Middle: A 67M model pre-trained on the HMM-generated dataset, and evaluated on task-10. Right: A 25M model pre-trained on OpenWebText, and evaluated on SST-2.

that SGD effectively minimizes the trace of the Hessian $\mathrm{Tr}[\nabla^2 L(\theta)]$ with the constraint set Γ similarly to ODE in (1).

Theorem 3.3. *Suppose the loss function L and the manifold of global minimizers Γ satisfy Assumption 3.1. For any $K>0$ such that ODE (1) has a solution $\{\hat{\theta}(t)\}_{t=0}^{K}$, it holds that θ^{η}_{K/η^2} converges in distribution to $\hat{\theta}(K)$ as $\eta \to 0$.*

Finally, we note that the above result can be extended to an arbitrary batch size B. The covariance of stochastic gradient at θ with batch size, denoted by $\Sigma_B(\theta)$, satisfies that $\Sigma_B(\theta) = \frac{1}{B}\Sigma(\theta)$. Therefore $\Sigma_B(\theta) = \frac{1}{B}\nabla^2 L(\theta)$ and we can again invoke Corollary 5.2 of Li et al. (2021) to derive the same result as in Theorem 3.3 but with the coefficient $\frac{1}{4}$ in equation (1) replaced by $\frac{1}{4B}$.

4. The Correlation between Flatness and Downstream Performance

The previous section proves that SGD prefers flatter models among all global minima with the same pre-training loss. In this section, we empirically validate the positive correlation between flatness of the model and downstream performance among models with the same (minimal) pre-training loss. Note that flatness has been shown to have a strong correlation with the generalization performance in the supervised settings (Keskar et al., 2016; Neyshabur et al., 2017; Jastrzębski et al., 2017; Jiang et al., 2019). For language models, the empirical and validation pre-training loss are nearly identical, and we focus on the transferability to downstream tasks.

We demonstrate a strong correlation between flatness and downstream performance in the models found in Section 2. As illustrated in Figure 1 (Right), the trace of Hessian is a good indicator of downstream performance when the pre-training loss becomes near optimal and stops being a reliable indicator. Moreover, we show that regularization techniques that encourage flatness such as SAM (Foret et al., 2020) and dropout (Srivastava et al., 2014) improve the downstream performance (without changing the validation pre-training loss).

Evaluation of flatness. Similar to the pre-training loss, the trace of Hessian of pre-training loss can also be calculated as a function of the model and the pre-training data. Also note that smaller trace of Hessian indicates flatter minima. Inspired by Lemma 3.2 and Wei et al. (2020), we can unbiasedly estimate the trace of Hessian with random samples. Details are provided in Section B.1.

Table 2. Comparison of pre-training algorithms.

PCFG	Pre-training	Task C %	$\mathrm{Tr}[\nabla^2 L(\theta)]$
AdamW	3.204	55.7±0.6	8.01±0.73
Adversarial	3.206	50.2±1.0	19.34±0.92
BookCorpus	**Pre-training**	**RTE %**	$\mathrm{Tr}[\nabla^2 L(\theta)]$
AdamW (w/ dropout, WD)	1.85	62.9	24.55±0.18
-dropout-WD	1.85	60.2	34.91±0.40
-dropout-WD+SAM	1.89	62.5	22.15±0.26

Results. In Figure 4, we compare the downstream accuracy and the trace of Hessian of different checkpoints obtained at different times during pre-training. On PCFG- and HMM-generated datasets, the trace of Hessian demonstrates a clear decreasing trend after the validation pre-training loss converges, following the prediction of Theorem 3.3. Furthermore, as the trace of Hessian decreases, the downstream performance improves by 1.6% and 4.0% on the PCFG- and HMM-generated datasets, respectively. When we pre-train on OpenWebText, we can observe a 1.25% increase in SST-2 accuracy and a 0.67 decrease in trace of Hessian as we continue to train the models from 400K to 1400K steps.

On the PCFG-generated dataset, we compare the trace of Hessian of the models pre-trained with adversarial algorithm and standard AdamW in Table 2. The trace of Hessian of the adversarially pre-trained model is 2 times larger than the normally pre-trained model, and has a drop of 5.5% in downstream performance.

We also study the impact of regularization on flatness and downstream performance. Note that dropout is known to encourage flatter models (Wei et al., 2020). We remove dropout and weight decay from pre-training on BookCorpus. Compard to the baseline (row "AdamW" in Table 2), we observe almost the same *validation* pre-training loss (despite the generalization gap of pre-training loss increase to 0.13), a 2.7% worse downstream performance on RTE, and a 40% larger trace of Hessian (see row "-droput-WD" in Table 2) Adding SAM (Foret et al., 2020) does not improve the pre-training loss, but improves the flatness and recovers the degradation of downstream performance compared to the model without

dropout (see last row of Table 2). These results suggest that explicit flatness regularization can improve downstream performance without changing the validation pre-training loss.

In Figure 5, we compare the downstream accuracy and the flatness of transformers with different sizes. For a fair comparison, we view transformers of various sizes as parameter configurations within a large transformer architecture, and compare the trace of Hessian of the new views of these models. Intuitively, we can view a small model as a large one by filling zeros into additional parameters or replicating the parameters of the small model. For MLPs, this maintains the input-output functionality and the trace of Hessian. Although the layer-norm in transformers causes subtleties, we can still keep the functionality and the trace of Hessian with respect to most parameters unchanged by replicating the parameters properly (See Section B.4 for details). Therefore, the views of these models have the same parameterization/architecture and the same pre-training loss (because they have the same representations as the corresponding original smaller models), and we evaluate the relationship between the their trace of Hessian and their downstream performance.

On the dataset generated by PCFG, the pre-training loss is almost the same for models (technically, the new views of these models) that are larger than 9M. As we increase the model size, the trace of Hessian of the pre-training loss decreases from 19.8 to 12.6, correlating with the increase of linear probe accuracy from 40.4% to 50.5%. On the OPT-generated dataset, we can also observe an increase in linear probe accuracy that co-occurs with a sharp decrease in the trace of Hessian, as we increase the model size.

Interaction between implicit bias and model size. Intuitively, the implicit bias drives both larger models and smaller models toward flat minima. As justified above, the smaller transformer architecture is a subset of the larger transformer architecture. Thus, the implicit bias drives the trace of Hessian to smaller value on a larger transformer compared with a smaller transformer, and performs better downstream.

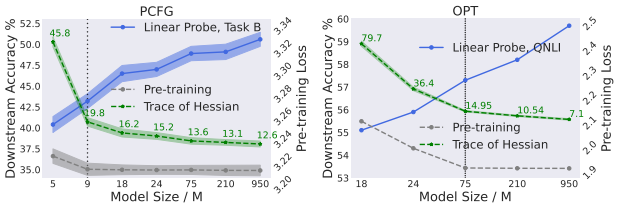


Figure 5. The trace of Hessian correlates with downstream performance for models with different sizes and almost the same pre-training loss. As we increase the model size, the trace of Hessian continues to decrease, as the downstream performance increases.

5. Flatness Regularization Identifies Transferable Models on Synthetic Language

Toward formally proving the connection between flatness and downstream performance, we consider a setting with synthetic Dyck language. The simplicity of the data allows us to sharply analyze the internal working of a single-layer transformer (with an attention layer and an MLP layer) for masked language modeling. We show that multiple parameter configurations can predict the conditional probability well, including one ideal model that learns the correct representations capturing the intrinsic structure of the sentence, and many "cheating" models that memorize the ground truth using random features. We will prove that the model with the smallest trace of Hessian is the desired model that transfers to downstream tasks.

Pre-training Distribution. Consider a variant of the Dyck language (Nivat, 1970) consisting of matching brackets. The vocabulary of the language has two brackets $\langle$ and $\rangle$. Each sentence is composed of a sequence of tokens such that the total numbers of $\langle$'s and $\rangle$'s are equal. To sample from the pre-training distribution P, we first draw a sentence uniformly over all valid sentences with even length T, and randomly select one position to replace the bracket with a mask.

Downstream Task. The most intrinsic property about the synthetic language is the difference in the number of left and right brackets, and thus we use it as the downstream task. Concretely, for any sequence x in $\{\langle,\rangle\}^*$ of length T, let $g^*(x)$ count the number of mismatches in x:

$$g^*(x) \triangleq \#\text{ of } \rangle\text{'s in } x - \#\text{ of } \langle\text{'s in } x \tag{2}$$

Thus, the sentence x is a valid string in the language if and only if $g^*(x) = 0$. For MLM, the masked token can also be recovered from $g^*(x)$: $g^*(x) = 1$ if the masked token is $\langle$, and $g^*(x) = -1$ if the masked token is $\rangle$. To evaluate if the model learns the structure, we consider a downstream distribution where each token is sampled from $\{\langle,\rangle\}$ uniformly, randomly, and independently.

Encoding of the Inputs. With a slight abuse of notation, we also denote by x_t the encoding of the t-th token. We encode the input as a one-hot vector in dimension $d = 2T$, where the index of the nonzero element encodes the position and the sign encodes the bracket. Concretely, let $e_t \in \mathbb{R}^d$ be the natural basis vector where the t-th entry is 1. Let $x_t = e_t$ if the t-th token is $\langle$ and $x_t = -e_t$ otherwise. If the position t is a mask, we set x_t to v, where $v \sim \text{Unif}(\{\pm e_{t+T}\})$. Note that the target function can be expressed as $g^\star(x) = -\langle \mathbf{1}_T, [\sum_{t=1}^T x_t]_{1:T} \rangle$ with this input encoding, where $\mathbf{1}_T$ is the all one vector in $\mathbb{R}^T$ and $[a]_{1:T}$ refers to the first T coordinates in a.

Models and Algorithms. Suppose $Q, K \in \mathbb{R}^{k\times d}$ are the query and key matrices, $V \in \mathbb{R}^{m\times d}$ is the value matrix and $u \in \mathbb{R}^m$ is the parameter of the output

layer. Let $\psi = (Q, K, V)$. A single-layer transformer is composed of an attention layer and an MLP layer. $[\text{Attn}_{\psi,u}(x)]_t = \frac{1}{m} u^\top \sigma(\sum_{j=1}^T a_{t,j} V x_j)$, where the attention score $a_{t,1:T} = \text{softmax}(\langle Qx_t, Kx_t\rangle, \cdots \langle Qx_t, Kx_T\rangle)$. $\sigma(x) = \max\{x, 0\}$ is the relu activation. We use the output of the first token, $f_{\psi,u}(x) = [\text{Attn}_{\psi,u}(x)]_1$.

We use the squared loss for both MLM and downstream adaptation. The loss function of MLM is $L(\psi, u)$. In downstream adaptation, we have a finite dataset $\{x^{(i)}\}_{i=1}^n$ sampled i.i.d. from P_{ds}. The training loss with n data is $\widehat{L}^{P_{\text{ds}}}(\psi, u)$, and the population loss for the downstream task is $L^{P_{\text{ds}}}(\psi, u)$.

Main Intuitions. We are interested in two kinds of parameter configurations both with good pre-training loss: (1) learning the natural and transferable features $\mathbf{1}_T$ and (2) fitting the pre-training task by memorizing the masked sentences. We construct the two solutions as follows. For solution (1), first note that the softmax attention layer can take the average of all the token encodings $[x_t]_{t=1}^T$ in a sentence. Let us denote the sum by $z \in \mathbb{R}^d$, $z = \sum_{t=1}^T x_t$. Note that the first T coordinates in z are ± 1 indicating the bracket type and the last T coordinates indicate the position of the mask. On top of z, two neurons can predict the masked token in MLM perfectly. Consider the two neurons $V_1 = [\mathbf{1}_T; \mathbf{0}_T]$, $V_2 = [-\mathbf{1}_T; \mathbf{0}_T]$. Then $g^*(x) = \sigma(V_2^\top z) - \sigma(V_1^\top z)$, which is the transferable solution. For solution (2), we set the entries in V to i.i.d. samples from $\mathcal{N}(0, T)$. If m is sufficiently large, we can find the coefficient u to express $g^*(x)$ with random Gaussian features, i.e. $g^*(x) = u^\top \sigma(Vz)$.

We observe that the trace of Hessian of configuration (1) is smaller than configuration (2), due to a main difference between them–the cancellation between activated neurons. In configuration (1), for every possible input, only one of the neurons $\sigma(V_1^\top z)$ and $\sigma(V_2^\top z)$ is activated. In contrast, in configuration (2), many neurons can be activated at the same time. Among them, the output coefficient u_i's contain both positive and negative values, leading to cancellation between activated neurons. In Lemma G.1, we link the trace of the Hessian with the cancellation between neurons. Indeed, we show that the mimimum of trace of the Hessian can be achieved only if there is no such cancellation. Therefore solution (1) is also the minimizer of the trace of Hessian. The intuitions are formalized in Theorem 5.1.

Theorem 5.1. *Suppose $m \geq 2$ and $T \geq 6$. Consider minimizing the trace of Hessian among all the solutions to the MLM pre-training task:* $\text{minimize}_{\psi,u} \text{Tr}[\nabla_\psi^2 L(\psi, u)] + \text{Tr}[\nabla_u^2 L(\psi, u)]$, *subject to $L(\psi, u) = 0$. The flattest solution $\hat{\psi}, \hat{u}$ are defined as the solution of the optimization problem above. Let $\tilde{u}$ be the minimizer of downstream training loss on top of $\hat{\psi}$, that is, $\tilde{u} \in \text{argmin}_u \|u\|_2$ subject to $\widehat{L}^{P_{\text{ds}}}(\hat{\psi}, u) = 0$. Then with probability at least $1 - 2^{-n}$, $L^{P_{\text{ds}}}(\hat{\psi}, \tilde{u}) = 0$.*

6. Related Work

Language modeling and downstream adaptation. Starting from Devlin et al. (2018), a line of works improve the downstream performance various tasks with increasing model size and data amount (Yang et al., 2019; Radford et al., 2019; Raffel et al., 2020). LLMs even exhibit unexpected emergent behaviors, such as in-context learning (Xie et al., 2021; Min et al., 2022), step-by-step reasoning (Wei et al., 2022), and zero-shot learning (Brown et al., 2020). Kaplan et al. (2020); Hernandez et al. (2021) study the behavior of language models with increasing size, and find out that the pre-training loss is typically correlated with downstream performance as model size increases. The pre-training loss is widely used as an evaluation metric. For example, efficient transformer works benchmark the pre-training loss given the same computation constraint (Dai et al., 2020; Wang et al., 2020; Choromanski et al., 2020; Liu et al., 2021).

Understanding the success of language modeling. Empirical works on understanding MLM find out that representations encode rich semantic and syntactic information (Peters et al., 2018; Htut et al., 2019; Hewitt & Manning, 2019; Mamou et al., 2020). Zhang & Hashimoto (2021) shows MLM recovers latent variables in graphical models. Recently, (Saunshi et al., 2022) show that models with the same pre-training loss but different architectures can have different downstream performance. Tay et al. (2021) find out that a narrow but deep transformer is better than a wide but shallow one with the same pre-training loss. Zhang et al. (2022b) demonstrate that Albert (Lan et al., 2019) generalizes better OOD than Bert on a synthetic reasoning task. These works indicate that architecture is an important factor for downstream performance beyond pre-training loss. This paper discovers the role of implicit bias in language modeling, which happens with models in the same architecture.

Implicit bias in supervised learning. The training algorithm chooses solutions with certain properties and usually leads to better generalization (Gunasekar et al., 2018; Soudry et al., 2018; Li et al., 2017b; Ji & Telgarsky, 2018; Arora et al., 2019a; Lyu & Li, 2019; Vaskevicius et al., 2019; Li et al., 2020; Yun et al., 2020; Amid & Warmuth, 2020a;b; Woodworth et al., 2020; HaoChen et al., 2020; Lyu et al., 2021; Azulay et al., 2021; Arora et al., 2022; Li et al., 2022; Lyu et al., 2022). Recently, Blanc et al. (2019); Damian et al. (2021); Li et al. (2021; 2022) demonstrate label noise SGD biases the models toward flatter minima. However, the setting of implicit bias in supervised learning is different from language modeling. In language modeling, we have access to gigantic corpus, and cannot interpolate the pre-training dataset. Moreover, we care about the adaptability of the solution on downstream tasks instead of generalization in distribution.

Flatness and regularization. The idea that flatness is related to generalization dates back to Hochreiter & Schmid-

huber (1997a). Jiang et al. (2019) demonstrated through experiments that flatness strongly correlates with generalization. Flatness regularization improves generalization in supervised learning (Foret et al., 2020; Zhang et al., 2021; Norton & Royset, 2021) and fine-tuning (Bahri et al., 2021). Recent work (Wen et al., 2022) theoretically shows that batch size can affect the exact notion of flatness being regularized. Fradkin et al. (2022) finds out contrastive pre-training favors flatter solutions compared with supervised pre-training and generalizes better. On the other hand, Dinh et al. (2017) showed flatness is not a necessity for generalization by constructing an artificially scaled network, while naturally trained models are unlikely to have such a scaling because the weight of different layers tends to keep balanced throughout training with balanced initialization (Du et al., 2018)).

7. Conclusion

We study the relationship between pre-training loss and downstream performance on language models. We discover that implicit bias matters beyond pre-training loss, and explore the mechanism of implicit bias in language modeling. We wish this motivates future works on the relationship between implicit bias and the feature representations of neural networks, as well as designing pre-training algorithms which leverage implicit bias and work better on downstream tasks.

Acknowledgement

We thank Jeff Z. HaoChen, Yining Chen, Garrett Thomas and Ananya Kumar for valuable feedbacks. HL is supported by the Stanford Graduate Fellowship. SMX is supported by an NDSEG Fellowship. The authors would like to thank the support from NSF IIS 2045685.

References

Amid, E. and Warmuth, M. K. Reparameterizing mirror descent as gradient descent. *Advances in Neural Information Processing Systems*, 33:8430–8439, 2020a.

Amid, E. and Warmuth, M. K. Winnowing with gradient descent. In *Conference on Learning Theory*, pp. 163–182. PMLR, 2020b.

Arora, S., Cohen, N., Hu, W., and Luo, Y. Implicit regularization in deep matrix factorization. In *Advances in Neural Information Processing Systems*, pp. 7411–7422, 2019a.

Arora, S., Khandeparkar, H., Khodak, M., Plevrakis, O., and Saunshi, N. A theoretical analysis of contrastive unsupervised representation learning. In *International Conference on Machine Learning*, 2019b.

Arora, S., Li, Z., and Panigrahi, A. Understanding gradient descent on the edge of stability in deep learning. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., and Sabato, S. (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 948–1024. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/arora22a.html.

Azulay, S., Moroshko, E., Nacson, M. S., Woodworth, B. E., Srebro, N., Globerson, A., and Soudry, D. On the implicit bias of initialization shape: Beyond infinitesimal mirror descent. In *International Conference on Machine Learning*, pp. 468–477. PMLR, 2021.

Ba, J. L., Kiros, J. R., and Hinton, G. E. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Bahri, D., Mobahi, H., and Tay, Y. Sharpness-aware minimization improves language model generalization. *arXiv preprint arXiv:2110.08529*, 2021.

Bai, Y. and Lee, J. D. Beyond linearization: On quadratic and higher-order approximation of wide neural networks. *International Conference on Learning Representations (ICLR)*, 2020.

Blanc, G., Gupta, N., Valiant, G., and Valiant, P. Implicit regularization for deep neural networks driven by an ornstein-uhlenbeck like process. *arXiv preprint arXiv:1904.09080*, 2019.

Borkar, V. S. *Stochastic approximation: a dynamical systems viewpoint*, volume 48. Springer, 2009.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901, 2020.

Chomsky, N. Three models for the description of language. *IRE Transactions on information theory*, 2(3):113–124, 1956.

Choromanski, K., Likhosherstov, V., Dohan, D., Song, X., Gane, A., Sarlos, T., Hawkins, P., Davis, J., Mohiuddin, A., Kaiser, L., et al. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*, 2020.

Dai, Z., Lai, G., Yang, Y., and Le, Q. Funnel-transformer: Filtering out sequential redundancy for efficient language processing. *Advances in neural information processing systems*, 33:4271–4282, 2020.

Damian, A., Ma, T., and Lee, J. D. Label noise sgd provably prefers flat global minimizers. *Advances in Neural Information Processing Systems*, 34:27449–27461, 2021.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Dinh, L., Pascanu, R., Bengio, S., and Bengio, Y. Sharp minima can generalize for deep nets. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1019–1028. JMLR. org, 2017.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Du, S. S., Hu, W., and Lee, J. D. Algorithmic regularization in learning deep homogeneous models: Layers are automatically balanced. In *Advances in Neural Information Processing Systems*, pp. 384–395, 2018.

Du, S. S., Hu, W., Kakade, S. M., Lee, J. D., and Lei, Q. Few-shot learning via learning the representation, provably. *arXiv preprint arXiv:2002.09434*, 2020.

Duchi, J. C. and Ruan, F. Stochastic methods for composite and weakly convex optimization problems. *SIAM Journal on Optimization*, 28(4):3229–3259, 2018.

Dziugaite, G. K. and Roy, D. M. Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data. *arXiv preprint arXiv:1703.11008*, 2017.

Fehrman, B., Gess, B., and Jentzen, A. Convergence rates for the stochastic gradient descent method for non-convex objective functions. *Journal of Machine Learning Research*, 21:136, 2020.

Foret, P., Kleiner, A., Mobahi, H., and Neyshabur, B. Sharpness-aware minimization for efficiently improving generalization. *arXiv preprint arXiv:2010.01412*, 2020.

Foret, P., Kleiner, A., Mobahi, H., and Neyshabur, B. Sharpness-aware minimization for efficiently improving generalization. In *International Conference on Learning Representations*, 2021.

Forney, G. D. The viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278, 1973.

Fradkin, P., Atanackovic, L., and Zhang, M. R. Robustness to adversarial gradients: A glimpse into the loss landscape of contrastive pre-training. In *First Workshop on Pre-training: Perspectives, Pitfalls, and Paths Forward at ICML 2022*, 2022. URL https://openreview.net/forum?id=-b3MEzI6N3.

Gokaslan, A., Cohen, V., Pavlick, E., and Tellex, S. Openwebtext corpus, 2019.

Gunasekar, S., Lee, J. D., Soudry, D., and Srebro, N. Implicit bias of gradient descent on linear convolutional networks. In *Advances in Neural Information Processing Systems*, pp. 9461–9471, 2018.

HaoChen, J. Z., Wei, C., Lee, J. D., and Ma, T. Shape matters: Understanding the implicit bias of the noise covariance. *arXiv preprint arXiv:2006.08680*, 2020.

HaoChen, J. Z., Wei, C., Gaidon, A., and Ma, T. Provable guarantees for self-supervised deep learning with spectral contrastive loss. *arXiv preprint arXiv:2106.04156*, 2021.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Hernandez, D., Kaplan, J., Henighan, T., and McCandlish, S. Scaling laws for transfer. *arXiv preprint arXiv:2102.01293*, 2021.

Hewitt, J. and Manning, C. D. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4129–4138, 2019.

Hochreiter, S. and Schmidhuber, J. Flat minima. *Neural Computation*, 9(1):1–42, 1997a.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997b.

Htut, P. M., Phang, J., Bordia, S., and Bowman, S. R. Do attention heads in bert track syntactic dependencies? *arXiv preprint arXiv:1911.12246*, 2019.

Izsak, P., Berchansky, M., and Levy, O. How to train bert with an academic budget. *arXiv preprint arXiv:2104.07705*, 2021.

Jastrzębski, S., Kenton, Z., Arpit, D., Ballas, N., Fischer, A., Bengio, Y., and Storkey, A. Three factors influencing minima in sgd. *arXiv preprint arXiv:1711.04623*, 2017.

Ji, Z. and Telgarsky, M. Gradient descent aligns the layers of deep linear networks. *arXiv preprint arXiv:1810.02032*, 2018.

Jiang, Y., Neyshabur, B., Mobahi, H., Krishnan, D., and Bengio, S. Fantastic generalization measures and where to find them. *arXiv preprint arXiv:1912.02178*, 2019.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., and Tang, P. T. P. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

Kim, Y., Dyer, C., and Rush, A. M. Compound probabilistic context-free grammars for grammar induction. *arXiv preprint arXiv:1906.10225*, 2019.

Kushner, H. and Yin, G. G. *Stochastic approximation and recursive algorithms and applications*, volume 35. Springer Science & Business Media, 2003.

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., and Soricut, R. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2019.

Lari, K. and Young, S. J. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer speech & language*, 4(1):35–56, 1990.

Lee, H. B., Lee, H., Na, D., Kim, S., Park, M., Yang, E., and Hwang, S. J. Learning to balance: Bayesian meta-learning for imbalanced and out-of-distribution tasks. In *International Conference on Learning Representations*, 2020.

Li, Q., Tai, C., and Weinan, E. Stochastic modified equations and adaptive stochastic gradient algorithms. In *International Conference on Machine Learning*, pp. 2101–2110. PMLR, 2017a.

Li, Q., Tai, C., and Weinan, E. Stochastic modified equations and dynamics of stochastic gradient algorithms i: Mathematical foundations. *The Journal of Machine Learning Research*, 20(1):1474–1520, 2019.

Li, Y., Ma, T., and Zhang, H. Algorithmic regularization in over-parameterized matrix sensing and neural networks with quadratic activations. *arXiv preprint arXiv:1712.09203*, pp. 2–47, 2017b.

Li, Z., Luo, Y., and Lyu, K. Towards resolving the implicit bias of gradient descent for matrix factorization: Greedy low-rank learning. *arXiv preprint arXiv:2012.09839*, 2020.

Li, Z., Wang, T., and Arora, S. What happens after sgd reaches zero loss?–a mathematical framework. *arXiv preprint arXiv:2110.06914*, 2021.

Li, Z., Wang, T., and Yu, D. Fast mixing of stochastic gradient descent with normalization and weight decay. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=sof8l4cki9.

Liu, H., HaoChen, J. Z., Wei, C., and Ma, T. Meta-learning transferable representations with a single target domain. *arXiv preprint arXiv:2011.01418*, 2020.

Liu, H., Dai, Z., So, D., and Le, Q. V. Pay attention to mlps. *Advances in Neural Information Processing Systems*, 34: 9204–9215, 2021.

Lyu, K. and Li, J. Gradient descent maximizes the margin of homogeneous neural networks. *arXiv preprint arXiv:1906.05890*, 2019.

Lyu, K., Li, Z., Wang, R., and Arora, S. Gradient descent on two-layer nets: Margin maximization and simplicity bias. *Advances in Neural Information Processing Systems*, 34: 12978–12991, 2021.

Lyu, K., Li, Z., and Arora, S. Understanding the generalization benefit of normalization layers: Sharpness reduction. *arXiv preprint arXiv:2206.07085*, 2022.

Mamou, J., Le, H., Del Rio, M., Stephenson, C., Tang, H., Kim, Y., and Chung, S. Emergence of separable manifolds in deep language representations. *arXiv preprint arXiv:2006.01095*, 2020.

Mandt, S., Hoffman, M. D., and Blei, D. M. Stochastic gradient descent as approximate bayesian inference. *Journal of Machine Learning Research*, 18:1–35, 2017.

Min, S., Lyu, X., Holtzman, A., Artetxe, M., Lewis, M., Hajishirzi, H., and Zettlemoyer, L. Rethinking the role of demonstrations: What makes in-context learning work? *arXiv preprint arXiv:2202.12837*, 2022.

Neyshabur, B., Bhojanapalli, S., McAllester, D., and Srebro, N. Exploring generalization in deep learning. In *Advances in Neural Information Processing Systems*, pp. 5947–5956, 2017.

Nivat, M. On some families of languages related to the dyck language. In *Proceedings of the second annual ACM symposium on Theory of computing*, pp. 221–225, 1970.

Norton, M. D. and Royset, J. O. Diametrical risk minimization: Theory and computations. *Machine Learning*, pp. 1–19, 2021.

Peters, M. E., Neumann, M., Zettlemoyer, L., and Yih, W.-t. Dissecting contextual word embeddings: Architecture and representation. *arXiv preprint arXiv:1808.08949*, 2018.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21: 1–67, 2020.

Raghu, A., Lorraine, J., Kornblith, S., McDermott, M., and Duvenaud, D. K. Meta-learning to improve pre-training. *Advances in Neural Information Processing Systems*, 34: 23231–23244, 2021.

Roark, B. and Bacchiani, M. Supervised and unsupervised pcfg adaptation to novel domains. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 205–212, 2003.

Saunshi, N., Malladi, S., and Arora, S. A mathematical exploration of why language models help solve downstream tasks. *arXiv preprint arXiv:2010.03648*, 2020.

Saunshi, N., Ash, J., Goel, S., Misra, D., Zhang, C., Arora, S., Kakade, S., and Krishnamurthy, A. Understanding contrastive learning requires incorporating inductive biases. *arXiv preprint arXiv:2202.14037*, 2022.

Soudry, D., Hoffer, E., Nacson, M. S., Gunasekar, S., and Srebro, N. The implicit bias of gradient descent on separable data. *The Journal of Machine Learning Research*, 19(1):2822–2878, 2018.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Su, W., Boyd, S., and Candes, E. A differential equation for modeling nesterov's accelerated gradient method: Theory and insights. In *Advances in Neural Information Processing Systems*, pp. 2510–2518, 2014.

Tay, Y., Dehghani, M., Rao, J., Fedus, W., Abnar, S., Chung, H. W., Narang, S., Yogatama, D., Vaswani, A., and Metzler, D. Scale efficiently: Insights from pre-training and fine-tuning transformers. *arXiv preprint arXiv:2109.10686*, 2021.

Tolstikhin, I. O., Houlsby, N., Kolesnikov, A., Beyer, L., Zhai, X., Unterthiner, T., Yung, J., Steiner, A., Keysers, D., Uszkoreit, J., et al. Mlp-mixer: An all-mlp architecture for vision. *Advances in Neural Information Processing Systems*, 34:24261–24272, 2021.

Vaskevicius, T., Kanade, V., and Rebeschini, P. Implicit regularization for optimal sparse recovery. In *Advances in Neural Information Processing Systems*, pp. 2968–2979, 2019.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. R. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.

Wang, S., Li, B. Z., Khabsa, M., Fang, H., and Ma, H. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

Wei, C. and Ma, T. Data-dependent sample complexity of deep neural networks via lipschitz augmentation. In *Advances in Neural Information Processing Systems*, pp. 9722–9733, 2019a.

Wei, C. and Ma, T. Improved sample complexities for deep networks and robust classification via an all-layer margin. *arXiv preprint arXiv:1910.04284*, 2019b.

Wei, C., Kakade, S., and Ma, T. The implicit and explicit regularization effects of dropout. *arXiv preprint arXiv:2002.12915*, 2020.

Wei, C., Chen, Y., and Ma, T. Statistically meaningful approximation: a case study on approximating turing machines with transformers, 2021a.

Wei, C., Xie, S. M., and Ma, T. Why do pretrained language models help in downstream tasks? an analysis of head and prompt tuning. *arXiv preprint arXiv:2106.09226*, 2021b.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Chi, E., Le, Q., and Zhou, D. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*, 2022.

Wen, K., Ma, T., and Li, Z. How does sharpness-aware minimization minimize sharpness? *arXiv preprint arXiv:2211.05729*, 2022.

Woodworth, B., Gunasekar, S., Lee, J. D., Moroshko, E., Savarese, P., Golan, I., Soudry, D., and Srebro, N. Kernel and rich regimes in overparametrized models. *arXiv preprint arXiv:2002.09277*, 2020.

Wu, D., Xia, S.-T., and Wang, Y. Adversarial weight perturbation helps robust generalization. *Advances in Neural Information Processing Systems*, 33:2958–2969, 2020.

Xie, S. M., Raghunathan, A., Liang, P., and Ma, T. An explanation of in-context learning as implicit bayesian inference. *arXiv preprint arXiv:2111.02080*, 2021.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., and Le, Q. V. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32, 2019.

Yun, C., Bhojanapalli, S., Rawat, A. S., Reddi, S. J., and Kumar, S. Are transformers universal approximators of sequence-to-sequence functions? *arXiv preprint arXiv:1912.10077*, 2019.

Yun, C., Krishnan, S., and Mobahi, H. A unifying view on implicit bias in training linear neural networks. *arXiv preprint arXiv:2010.02501*, 2020.

Zhang, J., Karimireddy, S. P., Veit, A., Kim, S., Reddi, S., Kumar, S., and Sra, S. Why are adaptive methods good for attention models? *Advances in Neural Information Processing Systems*, 33:15383–15393, 2020.

Zhang, S., Roller, S., Goyal, N., Artetxe, M., Chen, M., Chen, S., Dewan, C., Diab, M., Li, X., Lin, X. V., et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022a.

Zhang, T. and Hashimoto, T. On the inductive bias of masked language modeling: From statistical to syntactic dependencies. *arXiv preprint arXiv:2104.05694*, 2021.

Zhang, Y., Backurs, A., Bubeck, S., Eldan, R., Gunasekar, S., and Wagner, T. Unveiling transformers with lego: a synthetic reasoning task. *arXiv preprint arXiv:2206.04301*, 2022b.

Zhang, Z., Yang, J., Ji, X., and Du, S. S. Variance-aware confidence set: Variance-dependent bound for linear bandits and horizon-free bound for linear mixture mdp. *arXiv preprint arXiv:2101.12745*, 2021.

Zheng, Y., Zhang, R., and Mao, Y. Regularizing neural networks via adversarial model perturbation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8156–8165, 2021.

Zhu, Y., Kiros, R., Zemel, R., Salakhutdinov, R., Urtasun, R., Torralba, A., and Fidler, S. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pp. 19–27, 2015.

A. Details in Section 2

A.1. Generating Simplified Datasets

PCFG-generated dataset. We consider a PCFG with vocabulary size 200. The state space is S, and $|S|=50$. All the production rules have two symbols on the right side. The sentence length is limited to 32, which means the depth of the parse tree is limited to 6. We generate a total of 2×10^7 sentences, which is 3.4×10^8 tokens. The downstream tasks are classifying the non-terminal symbols in the parse tree of the PCFG (50-way classification). The label is defined as $y=\mathrm{argmax}_{s\in S}\Pr(s\,|\,x_1,x_2,...,x_{1+L})$. Tasks A, B and C are defined on the symbols corresponding to span length $L=32,16$ and 8, respectively. Each of the downstream task contains 0.1M examples. Examples of the generated trees are provided in Figure 6.

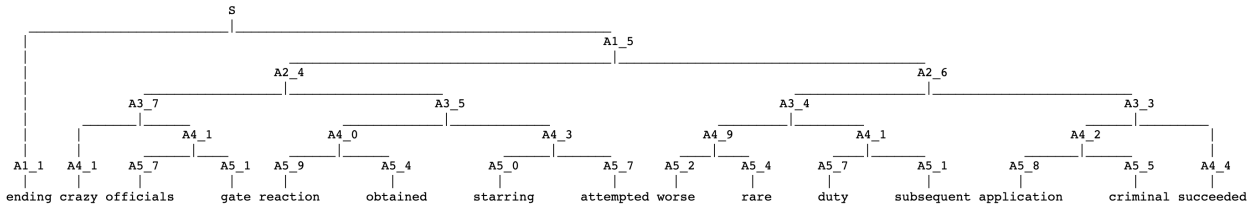


Figure 6. An example of the genearated PCFG sentence.

HMM-generated dataset. We consider an HMM with vocabulary size 200 and state space size 100. The sentence length is restricted to 16. We generate a total of 1×10^7 sentences, which is 1.6×10^7 tokens. The downstream task is to classify the latent variable in the HMM generative model. We consider task-6 and task-10, which classify the 6-th and 10-th hidden variables respectively. Each of the downstream task contains 0.1M examples.

OPT-generated dataset. We use the 125M OPT model to generate the training dataset. To simplify the dataset, we further process the logit of OPT to select only from the top-2000 tokens in the vocabulary. Starting from the bos token, we sample every token of the sentence from the predicted autoregressive LM probability. The sentence length is restricted to 24. We generate a total of 2×10^8 sentences, which is 3.2×10^9 tokens. Examples of the generated text are provided in Figure 7.

```
</s>I really don't either, so why do you feel it wouldn't be great when you can
</s>I want the person in the photo to tell me there are children under 6 years old in the
```

Figure 7. An example of the genearated OPT sentence.

A.2. Real-world Datasets

We use OpenWebText `https://huggingface.co/datasets/openwebtext` and BookCorpus `https://huggingface.co/datasets/bookcorpus` from huggingface. The sequence length is 128, and the tokenizer is the original one of `bert-large-uncased`. We always use 15% mask rate for real datasets.

A.3. Compute the True Conditional Probabilities from Generative Models

We can compute the true MLM conditional probability $\Pr(x_t\,|\,x_{-t})$ from the joint probability $\Pr(x_t,x_{-t})$,

$$\Pr(x_t=c\,|\,x_{-t})=\frac{\Pr(x_t=c,x_{-t})}{\sum_{c\in W}\Pr(x_t=c,x_{-t})}.$$

Since we already know the generative model, we can compute the joint probability efficiently. For PCFG, we can compute the joint probability with the inside algorithm, which decomposes the joint probability into lower layers in the parse tree. For HMM, we can compute the joint probability with the Viterbi algorithm. For OPT, we have $\Pr(x_1,...,x_T)=\Pr(x_1)\prod_{t=1}^{T-1}\Pr(x_{t+1}\,|\,x_1,...,x_t)$.

Table 3. Shape of the transformers in Section 2.

d-model	n-nead	#layers	d-inter
64	1	1	256
128	2	4	512
192	3	5	768
256	4	6	1024
512	8	8	2048
768	12	12	3072
1024	16	16	4096
1280	20	20	5120
1536	24	24	6144

A.4. Models

We use transformers on PCFG and OPT-generated datasets. We use learning rate 1e-3 and warmup proportion 0.06. All the models are trained based on the implementation of Izsak et al. (2021). We list the sizes of the transformers in Table 3. d-model is the size of the hidden layers. d-inter is the size of the intermediate layers in MLP. n-head is the number of heads per layer. #layers is the number of layers. For LSTMs, we use the implementation of PyTorch. We consider d-hidden in [128,256,512,768,1024], and #layers in [4,6,8,12,16].

A.5. Algorithms

Standard Pre-training. We use AdamW with constant learning rate 0.001. $\beta_1 = 0.9$ and $\beta_1 = 0.98$. We linearly increase the learning rate to do the warmup for 1000 steps on synthetic datasets and 5000 steps on real datasets. The reason why we use constant learning rate is that we wish to compare checkpoints at different number of steps and want to continue pre-training from checkpoints. Besides, The goal of the paper is to understand the relationship between pre-training loss, downstream performance and flatness, instead of achieving state-of-the-art. We include standard regularization, which is 0.1 dropout and 0.01 weight decay. We always use batchsize = 4096.

The lookup table. The lookup table can be thought of as a modeling whose input is the masked sentence and always output the true conditional probabilities. Since we already know the true conditional probabilities from the generative models as in Section A.3, and the pre-training loss of the lookup table is exactly the entropy of true conditional probability, we do not need to actually implement the lookup table. To evaluate the downstream performance of the lookup table, we can concatenate the true conditional probability vector at each token as the contextual embeddings and linear probe on top of it.

The adversarial algorithm. The adversarial algorithm we use to mess up the downstream performance is maximizing a meta-learning objective in pre-training. Suppose the linear head of the downstream task is g_ϕ and the feature representation is h_ψ. The meta-learning algorithm first trains the head g_ϕ to minimize the training loss of the downstream task, and then update h_ψ to maximize the validation loss on the downstream tasks. Concretely, we randomly sample two disjoint subsets D_1 and D_2 from the downstream training dataset D. We train g_ϕ to minimize the loss of downstream tasks on D_1, $\hat{\phi}(\psi) \in \operatorname{argmin} \frac{1}{|D_1|} \sum_{(x,y)\in D_1} \ell(g_\phi(h_\psi(x)),y)$. Then we train h_ψ to maximize the validation loss on D_2 during pre-training, $\operatorname{minimize}_\psi L(\psi) - \lambda \frac{1}{|D_2|} \sum_{(x,y)\in D_2} \ell(g_{\hat{\phi}(\psi)}(h_\psi(x)),y)$. The optimization can be efficiently carried out with closed form solution of ϕ as shown in Liu et al. (2020).

SAM (Foret et al., 2020). We implement SAM on BookCorpus without dropout and weight decay. The SAM radius ρ is set to 0.05. We split the 4096 batchsize in two halves and compute the adversarial point with the first half and the final update with the second half.

Fine-tuning. Following the standard protocol of Devlin et al. (2018), we use the contextual embeddings of the CLS token for fine-tuning. We use AdamW with learning rate 1e-4. We perform 200 warmup steps and train on the downstream tasks for 10 epochs.

Linear probe. Since the CLS token is not trained in pre-training, we concatenate embeddings of all the tokens in the sentence as the representations. We use AdamW with learning rate 1e-3 to train the linear head. We train on the downstream tasks for 100 epochs. Note that to make the capacity of the linear probe itself controlled, we adopt a random Gaussian projection to dimension 512 on the concatenation of the embeddings and find out that this does not affect the final performance. Therefore

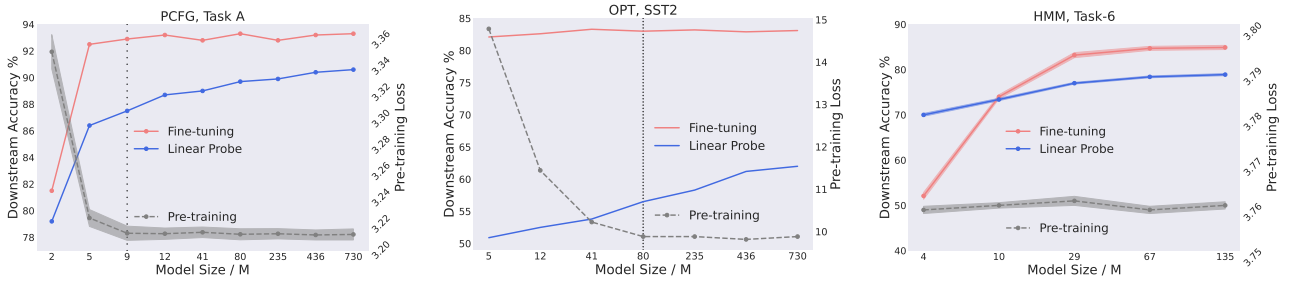
we report the results of standard linear probe by default. The error bar in the figures shows the standard deviation of 5 random seeds in linear probe.

We report the standard deviation of linear probe and fine-tuning from 5 random seeds.

Evaluation of pre-training loss. Since we have access calculate the true conditional probability, we can calculate the cross entropy loss as the sum of the entropy of the true conditional probability and the KL divergence between the predicted and true conditional probabilities. This is more accurate than evaluating on the validation datasets in the standard ways. We report the number of pre-training loss with 10^6 sentences, and calculate the standard deviation on 5 subsets, each of which has size 2×10^5.

A.6. Results on Other Downstream Tasks.

We also provide results on other downstream tasks in this subsection. On PCFG Task A, OPT SST-2 and the Task-6 of HMM, we can also observe the increase in downstream performance as we scale up the models in the saturation regime.

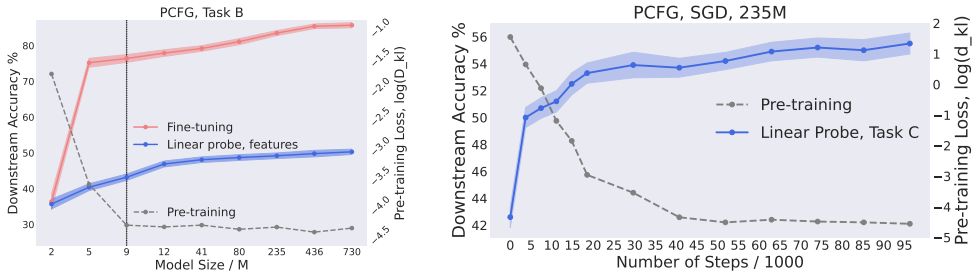


(a) PCFG→Task A (b) OPT→SST-2 (c) HMM→Task-6

Figure 8. The downstream accuracy continues to rise as we increase the model size, althought the pre-training loss remains unchanged.

A.7. Evaluation of Pre-training Loss with KL Divergence.

Since we have access to the true conditional probability from the generative models, we can decompose the cross entropy into the KL divergence between prediction and true conditional probability plus the entropy of the true conditional probability. When comparing different models in the saturation regime, we can use the KL divergence to reduce the variance of the loss evaluation. We provide the results with KL divergence evaluation in Figure 9. Even with the log of the KL divergence as the pre-training loss, we can still observe the models are saturating as training proceeds or as we scale up the models.



(a) PCFG→Task B (b) PCFG→Task C

Figure 9. Results with KL divergence evaluation.

B. Details in Section 4

B.1. Unbiased Estimate of the Trace of Hessian.

Evaluating the trace of Hessian requires the norm of the Jacobian $\nabla_\theta \log[f_\theta(x_{-t})]$. Since the output dimension c and the number of parameters are all very large, computing the Jacobian $\nabla_\theta \log[f_\theta(x_{-t})]$ will be very inefficient. Instead, we can estimate the trace of Hessian unbiasedly with random samples as follows. Suppose $f_\theta(x_{-t})$ is the predicted probability of the conditional probability. In the saturation regime, as $f_\theta(x_{-t})$ approaches the true conditional probability, the Hessian of the pre-training loss w.r.t. the parameters can be expressed as

$$\nabla^2 L(\theta) = \mathbb{E}_{t,x_{-t}} \mathbb{E}_{x_t|t,x_{-t}} \left[\nabla_\theta \log[f_\theta(x_{-t})]_{x_t} (\nabla_\theta \log[f_\theta(x_{-t})]_{x_t})^\top \right].$$

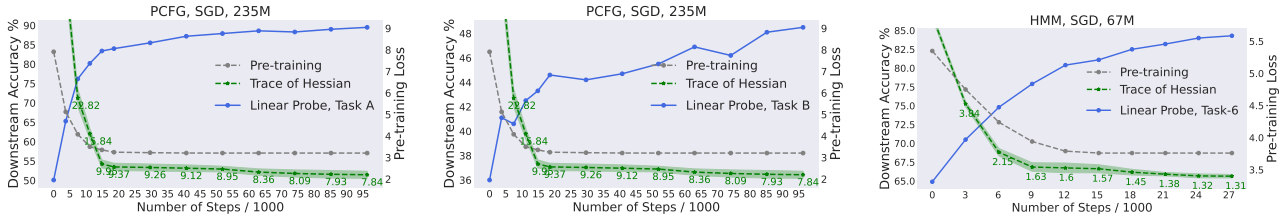
Therefore we have

$$\mathrm{Tr}(\nabla^2 L(\theta)) = \mathbb{E}_{t,x_{-t}} \mathbb{E}_{x_t|t,x_{-t}} \| \nabla_\theta \log[f_\theta(x_{-t})]_{x_t} \|_2^2.$$

To approximate this expectation, we can first sample t,x_{-t} from the language, then draw i.i.d. samples x_t from $(f_\theta(x_{-t}))$, and use the average as the unbiased estimate. For all experiments, we sample 10000 x_{-t} and sample 50 x_t for each x_{-t}.

B.2. Details in Figure 4.

To verify Theorem 3.3 that SGD biases the model towards flatter minima, we conduct MLM on PCFG and HMM-generated datasets with SGD. We set the proportion of warmup stage to 12% total number of steps, and fix the learning rate to 1e-3 after the warmup. We evaluate the downstream performance and the trace of Hessian of different checkpoints along pre-training. The standard deviation of trace of Hessian is calculated based on 5 times of sampling 50 examples as mentioned above. Apart from the PCFG task C and HMM task-10, we also provide results on PCFG tasks A, B and HMM task-6 in Figure 10.



(a) PCFG, SGD, Task A, 235M (b) PCFG, SGD, Task B, 235M (c) HMM, SGD, Task-6,67M

Figure 10. The trace of Hessian correlates with downstream performance for model checkpoints with different number of steps after the pre-training loss converges.

B.3. Details in Figure 5.

To compare the trace of Hessian of transformers with different sizes, we need to embed the smaller transformers into larger transformers. However, this requires the width of the larger transformers to be a multiple of the width of the smaller transformers. We set the width of the largest transformer to the least common multiple of the width of the smaller transformers as in Table 4. For evaluation, we still use linear probe, and follow the setting of Section 2.

B.4. Embedding of a Smaller Transformer into a Larger Transformer.

In this subsection, we show that a smaller transformer can be embed into a larger transformer without changing the functionality. We enable the embedding by considering two techniques (1) adding additional layers using residual connections without changing the functionality and (2) increasing feature dimension / adding more attention heads without change the functionality by duplicating the weights.

B.4.1. The Base Case with MLPs.

To gain some insights of how to increase the feature dimension without changing the functionality, we start with vanilla MLPs without layer-norm (Ba et al., 2016) and residual connections (He et al., 2016). Consider a multi-layer MLP $f_{W,a}(x)$.

Table 4. Shape of the transformers in Figure 5.

d-model	n-nead	#layers	d-inter
128	2	4	512
192	3	5	768
320	5	6	1024
384	6	7	2048
640	10	12	3072
960	15	16	4096
1920	30	20	5120

The weight matrices are $W = [W_0, \dots, W_{L-1}]$.The dimensionality is $W_l \in \mathbb{R}^{d_{l+1} \times d_l}$. The representations are defined recursively, $h_{l+1}(x) = \sigma(W_l h_l(x))$. We denote the input by $h_0(x) = x$ and the final output is defined as $f_{W,a}(x) = a^\top h_L(x)$. The activation σ here is relu or leaky relu. We aim to embed $f_{W,a}(x)$ into $f_{\tilde{W},\tilde{a}}(x)$, where $\tilde{W} = [\tilde{W}_0, \dots, \tilde{W}_{L-1}]$ and $\tilde{W}_l \in \mathbb{R}^{2d_{l+1} \times 2d_l}$. We need to make sure $f_{W,a}(x) = f_{\tilde{W},\tilde{a}}(x)$ for all x.

For this case, we can set $\tilde{W}_l = 1/2 \begin{bmatrix} W_l & W_l \\ W_l & W_l \end{bmatrix}$ for $l \in [L-1]$, $\tilde{W}_0 = \frac{1}{\sqrt{2}} \begin{bmatrix} W \\ W \end{bmatrix}$, and $\tilde{a} = \frac{1}{\sqrt{2}} \begin{bmatrix} a \\ a \end{bmatrix}$. We can verify that $\tilde{h}_l(x) = \frac{1}{\sqrt{2}} \begin{bmatrix} h_l(x) \\ h_l(x) \end{bmatrix}$ inductively. Therefore we have

$$
\begin{aligned}
f_{\tilde{W},\tilde{a}}(x) &= \tilde{a}^\top \tilde{h}_L(x) \\
&= \frac{1}{\sqrt{2}} [a^\top, a^\top] \frac{1}{\sqrt{2}} \begin{bmatrix} h_L(x) \\ h_L(x) \end{bmatrix} \\
&= \frac{1}{2} f_{W,a}(x) + \frac{1}{2} f_{W,a}(x) \\
&= f_{W,a}(x).
\end{aligned}
$$

B.4.2. Real Transformers.

Next we turn to transformers with residual connections and layer norm. We first use the same strategy as the MLP case to add additional feature dimension and attention heads by replicating the weights, and then show how to add new layers using residual connections. At a high level, replicating the weight maintains the mean and the variance calculated by the layernorm. Therefore the representations inside the transformer also get replicated, without changing the values in each of the replicated groups.

Setup. A transformer is composed of an input embedding W_E, L blocks of self-attention, and an output layer. Transformers also contain layer-norm and residual connections. Suppose the input $x = [x_1, \dots, x_T]$, where $x_i \in \mathbb{R}^d$. Each block of the self-attention contains of an attention layer and an MLP layer, both equipped with residual connections and layer-norm. Let us denote by $[h_0(x)]_i = \mathrm{LN}(W_E x_i)$ the input embeddings. Suppose the hidden size is d_h, i.e. $[h_l(x)]_i \in \mathbb{R}^{d_h}$. The attention layer is defined as $[v_l(x)]_i = \mathrm{LN}([h_l(x)]_i + [\mathrm{Attn}_l(h_l(x))]_i)$, and the MLP layer is defined as $[h_{l+1}(x)]_i = \mathrm{LN}([v_l(x)]_i + U_l \sigma(W_l [v_l(x)]_i + b_l))$. The activation σ is GeLU. The final output is $[f(x)]_i = W_E^\top [h_L(x)]_i$. Note that W_E is both the input embedding and the weight of the output layer. They are tied in training.

The layer norm is on the feature dimension. $[\mathrm{LN}(x_i)]_j = \gamma_j * \hat{x}_{ij} + \beta_j$. $\hat{x}_i$ is the normalized version of x_i with zero mean and unit variance. γ and β are trainable.

The multi-head attention consists of n_h self-attention heads. The definition of the multi-head attention is $\mathrm{Attn}_l(h_l(x)) = [(A_{l1} h_l(x) V_{l1})^\top, \dots, (A_{l n_h} h_l(x) V_{l n_h})^\top]^\top O_l$. The output matrix $O_l \in \mathbb{R}^{d_h \times d_h}$. The attention heads composes of the attention score times the feature matrix times the value matrix. The attention score $A_{lk} \in \mathbb{R}^{d' \times d'}$ is computed with softmax dot product. For each $k \in [n_h]$, $A_{lk} = \mathrm{softmax}(h_l(x) Q_{lk} K_{lk}^\top h_l(x)^\top)$.

Following the implementation of Devlin et al. (2018), the dimension of the attention head is always $d' = 64$, thus $d_h = 64 n_h$. The dimension of the intermediate layer in the MLP is set to $4d_h$, which means $U_l \in \mathbb{R}^{d_h \times 4d_h}$ and $W_l \in \mathbb{R}^{4d_h \times d_h}$.

We aim to embed the smaller transformer $f(x)$ into $\tilde{f}(x)$, where $\tilde{d}_h = 2d_h$, $\tilde{n}_h = 2n_h$, and $\tilde{L} = L + L'$.

Increasing feature dimension with replication of the parameters. Although the transformers have layer-norm and residual connections, we can still modify the strategy in the base case with MLPs slightly to increase the width of the model and the number of attention heads without changing the functionality. Consider the following weight replication method. For $l \in [0,...,L-1]$,

$$\tilde{W}_E = \frac{1}{2}\begin{bmatrix} W_E \\ W_E \end{bmatrix}, \tilde{\gamma}_l = \begin{bmatrix} \gamma_l \\ \gamma_l \end{bmatrix}, \tilde{b}_l = \begin{bmatrix} b_l \\ b_l \end{bmatrix}, \tilde{\beta}_l = \begin{bmatrix} \beta_l \\ \beta_l \end{bmatrix}, \tilde{W}_l = \frac{1}{2}\begin{bmatrix} W_l & W_l \\ W_l & W_l \end{bmatrix}, \tilde{U}_l = \frac{1}{2}\begin{bmatrix} U_l & U_l \\ U_l & U_l \end{bmatrix}, \tilde{O}_l = \frac{1}{2}\begin{bmatrix} O_l & O_l \\ O_l & O_l \end{bmatrix},$$

$$\tilde{Q}_{lk} = \frac{1}{2}[Q_{lk}, Q_{lk}], \tilde{K}_{lk} = \frac{1}{2}[K_{lk}, K_{lk}], \tilde{V}_{lk} = \frac{1}{2}[V_{lk}, V_{lk}] \text{ for } k \in [1,...,n_h],$$

$$\tilde{Q}_{lk} = \frac{1}{2}[Q_{l(k-n_h)}, Q_{l(k-n_h)}], \tilde{K}_{lk} = \frac{1}{2}[K_{l(k-n_h)}, K_{l(k-n_h)}], \tilde{V}_{lk} = \frac{1}{2}[V_{l(k-n_h)}, V_{l(k-n_h)}] \text{ for } k \in [n_h+1,...,2n_h].$$

We observe that the intermediate layers of the transformers are also replicated for the first L blocks, i.e. $\tilde{h}_l(x) = \begin{bmatrix} h_l(x) \\ h_l(x) \end{bmatrix}$ and $\tilde{v}_l(x) = \begin{bmatrix} v_l(x) \\ v_l(x) \end{bmatrix}$ for $l \in [1,...,L]$. First note that $[h_0(x)]_i = \mathrm{LN}(W_E x_i)$. Since replicating the features will not change the mean and the variance, we have $\tilde{h}_0(x) = \begin{bmatrix} h_0(x) \\ h_0(x) \end{bmatrix}$. Then we can show that replicating the features will not change the attention scores as well. This makes $\tilde{v}_l(x) = \begin{bmatrix} v_l(x) \\ v_l(x) \end{bmatrix}$. Finally note that we can apply the base case of the MLP to reason about the MLP layer, and show $\tilde{h}_{l+1}(x) = \begin{bmatrix} h_{l+1}(x) \\ h_{l+1}(x) \end{bmatrix}$. Therefore we have shown $\tilde{h}_l(x) = \begin{bmatrix} h_l(x) \\ h_l(x) \end{bmatrix}$ inductively.

Adding additional layers using residual connections. We have demonstrated that $\tilde{h}_l(x) = \begin{bmatrix} h_l(x) \\ h_l(x) \end{bmatrix}$ for $l \in [0,...,L]$. Now let's consider the added L' blocks on top of the small model. Since the transformer contains residual connections, we can add new blocks on top of a small model and fill in zeros to the added parameters. We will show that in this way, $\tilde{h}_l(x) = \tilde{h}_L(x)$, for any $l \in [L,...,L+L']$. This will indicate that $\tilde{h}_{L+L'}(x) = \tilde{h}_L(x) = \begin{bmatrix} h_l(x) \\ h_l(x) \end{bmatrix}$. Recall that $\tilde{W}_E = \frac{1}{2}\begin{bmatrix} W_E \\ W_E \end{bmatrix}$. This indicate that

$$\begin{aligned}
[\tilde{f}(x)]_i &= \tilde{W}_E^\top \tilde{h}_{L+L'}(x) \\
&= \frac{1}{2}[W_E^\top, W_E^\top]\begin{bmatrix} h_L(x) \\ h_L(x) \end{bmatrix} \\
&= \frac{1}{2}[f(x)]_i + \frac{1}{2}[f(x)]_i \\
&= [f(x)]_i,
\end{aligned}$$

which means we can add new layers on top of a small transformer without changing the functionality.

Now we show that $U_l = 0$ and $O_l = 0$ for $l \in [L,...L+L'-1]$ will make $\tilde{h}_l(x) = \tilde{h}_L(x)$, for any $l \in [L,...,L+L']$. This holds because $[v_l(x)]_i = \mathrm{LN}([h_l(x)]_i + [\mathrm{Attn}_l(h_l(x))]_i)$ and $[h_{l+1}(x)]_i = \mathrm{LN}([v_l(x)]_i + U_l\sigma(W_l[v_l(x)]_i + b_l))$. If $O_l = 0$, we have $v_l(x) = h_l(x)$ from the first equation. If $U_l = 0$, we have $h_{l+1}(x) = v_l(x)$ from the second equation. Therefore we have $\tilde{h}_l(x) = \tilde{h}_L(x)$, for any $l \in [L,...,L+L']$.

B.4.3. Viewing a Small Transformer as a Special Case of a Large Transformer

As demonstrated above, smaller transformers can be embedded into larger transformers with functionality preserved. The smaller transformer architecture can therefore be viewed as a subset of the larger transformer architecture. In this sense, a set of transformers with different sizes and the same pre-training loss found in Section 2 can be viewed as a set of transformers with the same size after the embedding. Note that the training algorithm only finds out the natural larger models, instead of the larger models which are embedded from the smaller models. This indicates that the implicit bias of the optimizer can interact with the model architecture. The implicit bias drives the model toward flat minima on both larger models and smaller models. The smaller transformer architecture is a subset of the larger transformer architecture, thus the flattest minima found with a larger transformer is flatter than the minima found with a smaller transformer. (See Figure 12).

C. Limitations

The implicit bias theory works with minibatch SGD. It is well known that Adam is better than SGD on transformers, but there is few understanding about why Adam is better than SGD as an optimizer for language models (Zhang et al., 2020). Although we prove minibatch SGD in language modeling in Theorem 3.3, we still need a more systematic understanding of the implicit bias of Adam in language models.

More general theory on the correlation between flatness and transferability. We show in Section 5 that flatness regularization leads to more transferable models in the simplified Dyck language setting. Note that such kind of general results can be very challenging. Results from the supervised setting cannot be readily adapted since they are obtained (partially) via generalization bounds (Wei & Ma, 2019a;b), which do not apply to the language modeling setting where the implicit bias is not related to the gap between the empirical and population loss. A similar result is from Du et al. (2020), which shows that learning representations with regularization leads to transferable features when the ground truth is from the same gaussian distribution. Proving the correlation between flatness and downstream performance in more general settings likely requires highly non-trivial and novel theoretical tools, and we hope to motivate future work on this topic.

D. Practical Implications

Pre-training Algorithms. Understanding the implicit biases needed for downstream performance may lead to better training methods (instead of better evaluation methods) that might encourage the correct biases more strongly. Therefore, a practical direction is to design better pre-training algorithms with more favorable biases which can lead to better downstream performance than AdamW and SGD.

Better Metrics for Language Models. In common practice, the validation pre-training loss is used to monitor the training process (Brown et al., 2020; Zhang et al., 2022a) and compare different models (Hernandez et al., 2021). However, Saunshi et al. (2022); Tay et al. (2021) show that pre-training loss is not necessarily correlated with downstream performance when comparing different architectures. We further show that pre-training loss may not always be a reliable indicator even for the same architecture. While downstream tasks could be used as a proxy metric for evaluation, the main issue is that large language models are trained to be general / multi-purpose models where the space of downstream tasks is large and unknown during the time of pre-training. Thus from a fundamental standpoint, it is beneficial to design a more reliable indicator that is agnostic to downstream tasks.

Explicit regularization. We show that implicit bias, especially the implicit bias of flatness matters for downstream performance in language modeling. Especially, in Section 2 and Section 4 we observe that SAM recovers the performance and flatness loss from removing dropout. Wei et al. (2020) show that dropout has an explicit and implicit regularization effect to minimize the Jacobian norm, which is closely related to the trace of Hessian. This again corroborates the relationship between flatness and downstream performance. Leveraging the implicit bias to design better explicit regularization in language modeling is also an important direction. Bahri et al. (2021) show explicit flatness regularization with SAM (Foret et al., 2020) can boost downstream performance when applying to downstream tasks themselves and the intermediate stages between pre-training and fine-tuning, but they did not study this on pre-training, partly because SAM is not efficient enough for pre-training (SAM requires back prop for 2 times per step, and more steps to reach the same level of pre-training loss (Foret et al., 2020)).

E. Additional Related Work

Understanding large language models. Saunshi et al. (2020) introduce the natural assumption, which states that downstream tasks can be solved linearly with the true conditional probability. Wei et al. (2021b) instantiate MLM on datasets generated by HMMs and show linear probe on top of MLM models solves downstream tasks. In contrast, our empirical evidence indicates that other factors related to the architecture and optimization also contribute to the performance beyond the natural assumption—somewhat surprisingly, we show that linear probe on top of the features of language models is *better* than linear probe on top of true conditional probability. Similar to our findings, Xie et al. (2021) also observe that despite similar perplexity, larger models are better than smaller modes for in-context learning, while in this paper, we focus on the standard fine-tuning and linear probe evaluation of language models, and provide a novel understanding of the mechanism behind the superiority of large models over small models.

Understanding self-supervised learning. Our work is also related to the broader theoretical self-supervised learning literature. This line of works study why a seemingly unrelated self-supervised objective helps improve the performance on downstream tasks. Arora et al. (2019b) prove that contrastive learning representations work on downstream linear classification tasks. Lee et al. (2020) study reconstruction-based self-supervised learning algorithms and show that linear probe on top of the self-supervised representations solves downstream tasks. HaoChen et al. (2021) show that the contrastive learning loss can be viewed as a principled spectral clustering objective. With the spectral contrastive loss, self-supervised representations recover the cluster structure in the augmentation graph. Recently, Saunshi et al. (2022) introduce the disjoint augmentation regime, where the minimizer of the contrastive learning loss can perform poorly on downstream tasks. Empirically, they find out that subtracting the mean of the representations of each class makes self-supervised models perform worse on downstream tasks, and ResNet (He et al., 2016) can have better downstream performance on downstream tasks than ViT (Dosovitskiy et al., 2020) and MLP-Mixer (Tolstikhin et al., 2021) on modified images. This indicates that pre-training loss is not all that matters for good downstream performance in self-supervised learning.

Implicit bias in supervised learning. There are other prior works (Kushner & Yin, 2003; Borkar, 2009; Su et al., 2014; Li et al., 2017a; Mandt et al., 2017; Duchi & Ruan, 2018; Li et al., 2019) on analyzing discrete-time dynamics via continuous-time approaches, earlier than (Blanc et al., 2019; Damian et al., 2021; Li et al., 2021; 2022). Please see Wen et al. (2022) for a more detailed discussion.

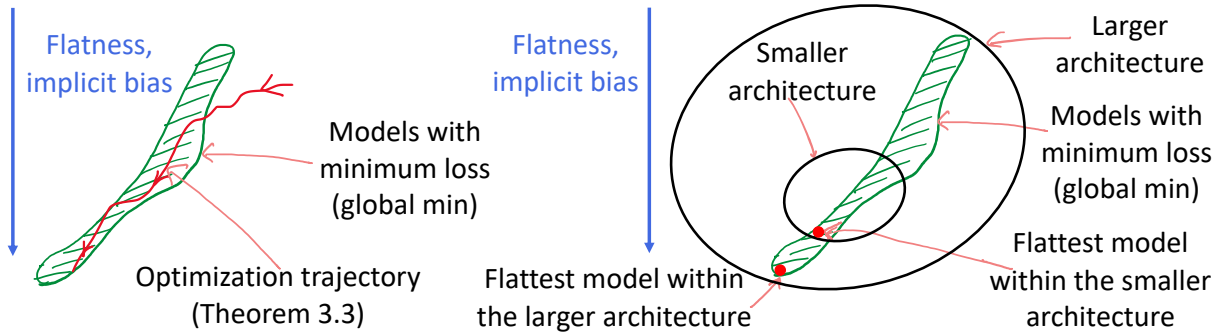


Figure 11. Left: The role of implicit bias. After the pre-training loss converges, implicit bias drives the model toward flat minima, as predicted by Theorem 3.3. Right: The interaction between model size and implicit bias. Implicit bias drives models toward flat minima on both larger models and smaller models. The smaller model architecture can be viewed as a subset of the larger model architecture as justified in Section 4 and Section B.4. Therefore, larger models can achieve flatter minima than smaller models.

F. Omitted Proofs in Section 3

Proof of Lemma 3.2. We first recall loss

$$L(\theta)=\mathbb{E}_{x,t}[-\log\left[f_\theta(x_{-t})\right]_{x_t}]=\mathbb{E}_{t,x_{-t}}\mathbb{E}_{x_t|t,x_{-t}}[-\log\left[f_\theta(x_{-t})\right]_{x_t}].$$

Note that conditioned on any x_{-t},t, it holds that

$$\begin{aligned}
&\mathbb{E}_{x_t|t,x_{-t}}\left[-\nabla_\theta^2\log\left[f_\theta(x_{-t})\right]_{x_t}\right]\\
=&\mathbb{E}_{x_t|t,x_{-t}}\left[-\frac{\nabla_\theta^2[f_\theta(x_{-t})]_{x_t}}{[f_\theta(x_{-t})]_{x_t}}\right]+\mathbb{E}_{x_t|t,x_{-t}}\left[\frac{\nabla_\theta[f_\theta(x_{-t})]_{x_t}(\nabla_\theta[f_\theta(x_{-t})]_{x_t})^\top}{[f_\theta(x_{-t})]_{x_t}^2}\right]\\
=&0+\mathbb{E}_{x_t|t,x_{-t}}\left[\nabla_\theta\log[f_\theta(x_{-t})]_{x_t}(\nabla_\theta\log[f_\theta(x_{-t})]_{x_t})^\top\right],
\end{aligned}$$

where in the last step, we use the assumption that $\theta\in\Gamma$, that is, for all x,t, $f_\theta(x_{-t})=\Pr(\cdot\,|\,x_{-t})$, which implies the following

$$\mathbb{E}_{x_t|t,x_{-t}}\left[-\frac{\nabla_\theta^2[f_\theta(x_{-t})]_{x_t}}{[f_\theta(x_{-t})]_{x_t}}\right]=-\sum_{x_t=1}^{c}\nabla_\theta^2[f_\theta(x_{-t})]_{x_t}=-\nabla_\theta^2\sum_{x_t=1}^{c}[f_\theta(x_{-t})]_{x_t}=-\nabla_\theta^2 1=0.$$

Since θ is a global minimizer of L, we have that $\nabla L(\theta)=\mathbb{E}_{t,x}\nabla_\theta\log[f_\theta(x_{-t})]_{x_t}=0$. Therefore, we have that

$$\begin{aligned}
\Sigma(\theta)=&\mathbb{E}_{t,x}\left[\nabla_\theta\log[f_\theta(x_{-t})]_{x_t}(\nabla_\theta\log[f_\theta(x_{-t})]_{x_t})^\top\right]\\
&-\mathbb{E}_{t,x}\nabla_\theta\log[f_\theta(x_{-t})]_{x_t}(\mathbb{E}_{t,x}\nabla_\theta\log[f_\theta(x_{-t})]_{x_t})^\top\\
=&\mathbb{E}_{t,x_{-t}}\mathbb{E}_{x_t|t,x_{-t}}\left[\nabla_\theta\log[f_\theta(x_{-t})]_{x_t}(\nabla_\theta\log[f_\theta(x_{-t})]_{x_t})^\top\right]\\
=&\mathbb{E}_{t,x_{-t}}\nabla_\theta^2\left(\mathbb{E}_{x_t|t,x_{-t}}[-\log\left[f_\theta(x_{-t})\right]_{x_t}]\right)\\
=&\nabla^2 L(\theta),
\end{aligned}$$

which completes the proof. □

G. Omitted Proofs in Section 5

G.1. Omitted Proofs of Theorem 5.1

Recall that the loss function of MLM is $L(\psi,u)=\mathbb{E}_{x\sim P}[(f_{\psi,u}(x)-g^*(x))^2]$. In downstream adaptation, we have access to a finite dataset $\{x^{(i)}\}_{i=1}^n$ sampled i.i.d. from P_{ds}. The training loss is $\widehat{L}^{P_{\text{ds}}}(\psi,u)=\frac{1}{n}\sum_{i=1}^n[(f_{\psi,u}(x^{(i)})-g^*(x^{(i)}))^2]$, and the population loss for the downstream task is $L^{P_{\text{ds}}}(\psi,u)=\mathbb{E}_{x\sim P_{\text{ds}}}[(f_{\psi,u}(x)-g^*(x))^2]$. An example of the input embedding and the two configurations is provided in Figure 12.

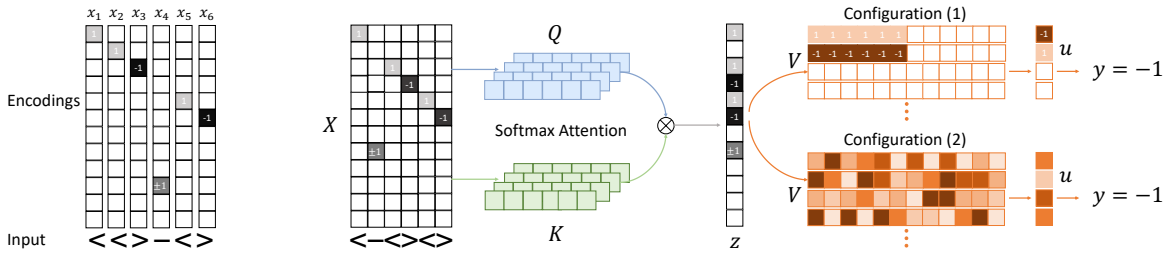


Figure 12. Left: An example of input encodings with $T=6$. Right: Illustration of the two solutions. The attention sums the encodings into z. Solution (1) contains two features transferable to the downstream task. The neurons in solution (2) are sampled randomly, and unrelated to the downstream task. Both solutions output the correct prediction for pre-training, but solution (1) is much flatter.

Proof of Theorem 5.1. We first calculate the trace of Hessian of the pre-training loss and then derive a lower bound for it in Lemma G.1. We then show that the lower bound can be achieved only if the output of the attention are in one direction for all the downstream input in Lemma G.3. This translates to constant sample complexity for the downstream task.

Lemma G.1. *Denote by $h_{Q,K}(x)=\sum_{j=1}^T a_j x_j$ the output of the attention head. In the setting of Theorem 5.1, $I_+=\{i\in[m]\,|\,u_i>0\}$ and $I_-=\{i\in[m]\,|\,u_i<0\}$. The trace of Hessian can be lower bounded,*

$$\text{Tr}[\nabla_\psi^2 L(\psi,u)]+\text{Tr}[\nabla_u^2 L(\psi,u)]\geq\sqrt{\frac{4}{T}},$$

where the lower bound is achieved if and only if the following conditions are satisfied,

$$\forall\, i\in[m],x\in\{x\,|\,V_i^\top h_{Q,K}(x)>0\},\quad V_i^\top h_{Q,K}(x)=|u_i|\|h_{Q,K}(x)\|_2, \tag{3}$$

$$\forall x, i\in I_+, i'\in I_-,\quad V_i^\top x V_{i'}^\top x\leq 0, \tag{4}$$

$$\forall x, j\in[T],\quad a_j=\frac{1}{T}. \tag{5}$$

Denote by $D_+=\{x\mid y=1\}$ and $D_-=\{x\mid y=-1\}$. I_x is the set of index of neurons which is activated on x, $I_x=\{i\in[T]\,|\,V_i^\top x>0\}$. By the condition in equation 5, the attention is taking the average of $[x_t]_{t=1}^T$. We can map D_+ and D_- to the feature space, $H_+=\{h_{Q,K}(x)\,|\,y=1\}$ and $H_-=\{h_{Q,K}(x)\,|\,y=-1\}$. We first show that one neuron cannot be activated on inputs from both D_+ and D_-, and all non zero neuron has to be activated on some input. Also note that a neuron cannot be activated on no input, unless the weight is 0.

Fact G.2. *(1) $\forall x\in D_+$, $I_x\subseteq I_+$. Similarly we have $\forall x\in D_-$, $I_x\subseteq I_-$. (2) Suppose $V_i\neq 0$, then there exists $h\in H_+\cup H_-$, $V_i^\top h>0$.*

Proof of Fact G.2. (1) Otherwise, suppose $j\in I_x\cap I_-$, since $y=\frac{1}{m}\sum_{i=1}^m u_i\sigma(V_i^\top h_{Q,K}(x))>0$, there has to be $j'\in I_x\cap I_+$, which contradicts the condition in equation 4. (2) Suppose $v^\top h\leq 0$ for all $h\in H_+\cup H_-$. Then we have $v^\top h=0$ for all h, since $v^\top h<0$ indicates $v^\top(-h)>0$, and $-h$ belongs to the support of P due to the symmetry of the distribution. However, in Lemma G.5, we show that the matrix stacking all input together has full row rank, thus v has to be 0, leading to a contradiction. □

We have the following lemma characterizing the solutions achieving all the qualities in Lemma G.1. Intuitively, all the neurons can be divided into two sets, and each input can only activate neurons in one of the sets, leading to no cancellation between activated neurons. This holds because of equation 4 and the properties of the input distribution.

Lemma G.3. *Suppose Q,K,V satisfy the equality in Lemma G.1. For all $i \in I_-$, on downstream data x, if $g^*(x)=1$, we have $V_i^\top h_{Q,K}(x)=c_i>0$. c_i is a constant which holds for every x if $g^*(x)=1$. If $g^*(x)=-1$, we have $V_i^\top h_{Q,K}(x)=0$.*

For all $i \in I_+$, on downstream data x, if $g^*(x)=-1$, we have $V_i^\top h_{Q,K}(x)=c_i>0$. c_i is a constant which holds for every x if $g^*(x)=1$. If $g^*(x)=1$, we have $V_i^\top h_{Q,K}(x)=0$.

Now let us consider the downstream task. It suffices to consider the constant vector c. If samples satisfying $g^*(x)=1$ and $g^*(x)=-1$ both show up in the downstream dataset, the minimal norm solution $\tilde{u}$ is $\tilde{u}_{I_-}=\frac{mc_{I_-}}{\|c_{I_-}\|_2^2}$, $\tilde{u}_{I_+}=\frac{-mc_{I_+}}{\|c_{I_+}\|_2^2}$, and $\tilde{u}_{[m]\setminus(I_+\cup I_-)}=0$. Then we can verify that

$$\begin{aligned}
f_{\hat{\psi},\tilde{u}}(x) &= \frac{1}{m}\left[\sum_{i\in I_+}\frac{-mc_i}{\|c_{I_+}\|_2^2}\sigma(V_i^\top h_{Q,K}(x))+\sum_{i\in I_-}\frac{mc_i}{\|c_{I_-}\|_2^2}\sigma(V_i^\top h_{Q,K}(x))\right]\\
&= \frac{1}{m}\left[\sum_{i\in I_+}\frac{-mc_i}{\|c_{I_+}\|_2^2}c_i\mathbb{I}[g^*(x)=-1]+\sum_{i\in I_-}\frac{mc_i}{\|c_{I_-}\|_2^2}c_i\mathbb{I}[g^*(x)=1]\right]\\
&= \mathbb{I}[g^*(x)=1]-\mathbb{I}[g^*(x)=-1]\\
&= g^*(x).
\end{aligned}$$

Therefore, $L^{P_{\text{ds}}}(\hat{\psi},\tilde{u})=\mathbb{E}_{x\sim P_{\text{ds}}}[(f_{\hat{\psi},\tilde{u}}(x)-g^*(x))^2]=0$, which completes the proof. □

We first show that when the pre-training loss equals 0, the trace of Hessian equals the square of the norm of the gradient.

Lemma G.4. *For any parameters θ, if the pre-training loss $L(\theta)=\mathbb{E}_x[(f_\theta(x)-y)^2]=0$, the trace of Hessian equals the square of the norm of the gradient,*

$$\text{Tr}[\nabla_\theta^2 L(\theta)]=\mathbb{E}[\|\nabla_\theta f_\theta(x)\|_2^2].$$

Proof of Lemma G.4. We can express the Hessian as follows.

$$\nabla_\theta^2 L(\theta)=\mathbb{E}_x[\ell'(f_\theta(x),y)\nabla_\theta^2 f_\theta(x)]+\mathbb{E}_x[\frac{1}{2}\ell''(f_\theta(x),y)\nabla_\theta f_\theta(x)\nabla_\theta f_\theta(x)^\top].$$

Since $L(\theta)=\mathbb{E}_x[(f_\theta(x)-y)^2]=0$, we have with probability 1, $\ell'(f_\theta(x),y)=0$ and $\ell''(f_\theta(x),y)=2$ is a constant. □

Proof of Lemma G.1.

$$\begin{aligned}
\mathrm{Tr}[\nabla_\psi^2 L(\psi,u)]+\mathrm{Tr}[\nabla_u^2 L(\psi,u)] &= \sum_{\theta\in[Q,K,V,u]} \mathbb{E}[\|\nabla_\theta f_{\psi,u}(x)\|_2^2] && \text{(By Lemma G.4)}\\
&\geq \mathbb{E}[\|\nabla_V f_{\psi,u}(x)\|_2^2+\|\nabla_u f_{\psi,u}(x)\|_2^2] && (6)\\
&= \frac{1}{m}\mathbb{E}\left[\|\sigma(Vh_{Q,K}(x))\|_2^2+\left\|h_{Q,K}(x)(\mathbb{I}[Vh_{Q,K}(x)>0]\odot u)^\top\right\|_2^2\right]\\
&= \frac{1}{m}\mathbb{E}\left[\sum_{i=1}^m \sigma(V_i^\top h_{Q,K}(x))^2+\|h_{Q,K}(x)\|_2^2\mathbb{I}[V_i^\top h_{Q,K}(x)>0]u_i^2\right]\\
&\geq \frac{2}{m}\mathbb{E}\left[\sum_{i=1}^m \sigma(V_i^\top h_{Q,K}(x))\|h_{Q,K}(x)\|_2|u_i|\right] && (7)\\
&\geq \frac{2}{m}\mathbb{E}\left[\|h_{Q,K}(x)\|_2\left|\sum_{i=1}^m \sigma(V_i^\top h_{Q,K}(x))u_i\right|\right] && (8)\\
&= 2\mathbb{E}\left[\|h_{Q,K}(x)\|_2|f_{\theta,u}(x)|\right]\\
&\geq \sqrt{\frac{4}{T}}. && (9)
\end{aligned}$$

The equality in step 6 is achieved if and only if the gradient of Q and K is 0. Equation (7) is from AM-GM, and the equality is achieved iff

$$V_i^\top h_{Q,K}(x)=|u_i|\|h_{Q,K}(x)\|_2 \quad \forall\, i\in[m], x\in\{x\,|\,V_i^\top h_{Q,K}(x)>0\}.$$

The equality in step 8 is achieved iff on all input, there is no cancellation between activated neurons,

$$\forall x, i\in I_+, i'\in I_-, \quad V_i^\top x V_{i'}^\top x\leq 0.$$

Since the attention score a_j satisfies $a_j>0$ and $\sum_{j=1}^T a_j=1$, and all embeddings x_t in one masked sentence are orthogonal to each other with norm 1, we have $\|h_{Q,K}(x)\|_2\geq\frac{1}{\sqrt{T}}$. The equality is achieved iff $a_j=\frac{1}{T}$ for all x and all $j\in[T]$. □

Proof of Lemma G.3 . Suppose V_i is a neuron with $i\in I_-$. Then there exists $h\in H_-$, $V_i^\top h>0$. Without loss of generality, suppose the masked position in h is 1, i.e. $h_1=0$, $h_2=1$. Now let us consider the components in V_i corresponding to the input positions and the mask positions separately. $V_i^{(c)}=[V_{i,1},V_{i,2},...,V_{i,T}]$ and $V_i^{(p)}=[V_{i,T+1},V_{i,T+2},...,V_{i,2T}]$.

We claim that ${V_i^{(c)}}_{2:T}=c\mathbf{1}$ for some $c>0$ and ${V_i^{(p)}}_1$ is either 0 or c. To prove this, consider $\tilde{h}$, which is only different from h on the mask, $h-\tilde{h}=2e_2$. Also consider $-h$ and $-\tilde{h}$. Due to the symmetry of the distribution, $-h$ and $-\tilde{h}$ are in H_+. By Fact G.2, $V_i^\top(-h)\leq 0$ and $V_i^\top(-\tilde{h})\leq 0$. $V_i^\top(-h)$ cannot be 0, because this will leads to $V_i^\top h=0$.

Case 1. $V_i^\top(-h)<0$ and $V_i^\top(-\tilde{h})<0$. We have $V_i^\top h=V_i^\top\tilde{h}>0$, indicating that $V_i^\top(h-\tilde{h})=2{V_i^{(p)}}_1=0$. Now we show that ${V_i^{(c)}}_{2:T}=c\mathbf{1}$. Due to the condition of equation 3, we know that for any $h\in H_-$ masked on the first token, either $V_i^\top h=0$ or they equal to the same positive value c for all h. We claim that $V_i^\top h=c$ for any $h\in H_-$. Otherwise there exist $H_{-,1}$ and $H_{-,0}$, $H_{-,0}\cap H_{-,1}=\emptyset$ and $H_{-,0}\cup H_{-,1}=H_-\cap\{h\,|\,h_2=\pm 1\}$. $V_i^\top h=c$ for any $h\in H_{-,1}$ and $V_i^\top h=0$ for any $h\in H_{-,0}$. This cannot happen for $T\geq 6$. By Lemma G.5 we know that the matrix stacking all such $h_{2:T}$ together has full row rank, thus ${V_i^{(c)}}_{2:T}=c\mathbf{1}$ for some $c>0$ and ${V_i^{(p)}}_1=0$.

Case 2. $V_i^\top(-h)<0$ and $V_i^\top(-\tilde{h})=0$, which indicates $V_i^\top h={V_i^{(p)}}_1$. Consider another $h'\in H_-$ which is not equal to h and $h'_2=1$. Similarly we can find $\tilde{h}'$, $h'-\tilde{h}'=2e_2$. Still we have $V_i^\top(-h')<0$ and $V_i^\top(-\tilde{h}')=0$, this tells us $V_i^\top h'=V_i^\top h$, due to the condition of equation 3. Applying this to different h's, we have that $V_i^\top h$ equals the same positive value for all $h\in H_-$ and the masked position is 0. By Lemma G.5 we know that the matrix stacking all such $h_{2:T}$ together has full row rank, thus ${V_i^{(c)}}_{2:T}=c\mathbf{1}$ for some $c>0$ and ${V_i^{(p)}}_1=c$.

We have proved that ${V_i^{(c)}}_{2:T}=c\mathbf{1}$ for some $c>0$ and ${V_i^{(p)}}_1$ is either 0 or c. We continue to show that $V_i^{(c)}=c\mathbf{1}$ for the same $c>0$ and either $V_i^{(p)}$ is 0 or its coordinates is $\pm c$.

For case 1, consider $h'' \in H_-$ whose masked position is 2, $h''_4 = 1$. Also suppose that $h''_1 = 1$ By Fact G.2, we know that $V_i^\top(-h'') \le 0$ and $V_i^\top(-\tilde{h}'') \le 0$, this implies that $-V_i^{(c)}{}_1 \le V_i^{(p)}{}_2 \le V_i^{(c)}{}_1$. Applying the same argument above, we know that either $V_i^{(p)}{}_2 = 0$ or $V_i^{(p)}{}_2 = \pm V_i^{(c)}{}_1$, otherwise both $V_i^\top h'' > 0$ and $V_i^\top \tilde{h}'' > 0$ hold, and $V_i^\top h'' \neq V_i^\top \tilde{h}''$, contradicting condition in equation 3. If $V_i^{(p)}{}_2 = V_i^{(c)}{}_1$, from equation 3 we know $V_i^\top h'' = V_i^\top h$, which indicates $V_i^{(p)}{}_2 = V_i^{(c)}{}_1 = \frac{c}{2}$. In this case we can find another h''' whose masked position is 2, $h'''_4 = 1$ but $h'''_1 = -1$. Then $V_i^\top h''' \neq V_i^\top h$, contradicting equation 3. Thus $V_i^{(p)}{}_2 \neq V_i^{(c)}{}_1$. Similarly $V_i^{(p)}{}_2 \neq -V_i^{(c)}{}_1$. The only possible situation is $V_i^{(p)}{}_2 = 0$. Applying the argument in this paragraph to other masked position, we have $V_i^{(p)} = 0$. Since H_- is invariant under permutation, $V_i^{(c)}{}_1 = c$, and $V_i^{(c)} = c\mathbf{1}$.

For case 2, exactly the same argument as the above paragraph with the same h'' and h''' shows that the coordinates of $V_i^{(p)}$ is $\pm c$.

Therefore, we have shown that for any $i \in I_-$, $V_i^{(c)} = c\mathbf{1}$ for same $c > 0$. The symmetry of distribution immediately tells us for any $i \in I_+$, $V_i^{(c)} = c\mathbf{1}$ for $c < 0$.

On the downstream distribution P_*, since there is no masked token, only $V_i^{(c)}$ is working. Since $V_i^{(c)} = c\mathbf{1}$ always holds, we complete the proof. □

Lemma G.5. *Suppose $M \in \mathbb{R}^{(2k+1)\times(2k+1)}$ is a matrix composed of ± 1. The first row of M is $[\underbrace{1,...,1}_{\text{k+1 1's}},\underbrace{-1,...,-1}_{\text{k -1's}}]$. Define a permutation $\rho(1) = 2$, $\rho(2) = 3 ... \rho(2k+1) = 1$. For all $i \ge 2$, $M_{i,\rho(j)} = M_{i-1,j}$. Then the rank of M is $2k+1$.*

Proof of Lemma G.5. Note that $M_i + M_{\rho^{k+1}(i)} = 2e_i$ for all $i \in [2k+1]$, which means we can express the orthonormal basis as linear combination of the rows in M. Therefore, the rank of M is $2k+1$. □

G.2. The existence of random feature solutions G.6

Theorem G.6. *Suppose $m \geq \tilde{O}(2^T T^3 \epsilon^{-2})$, and $V_i \sim \mathcal{N}(0, TI_{2T})$ for all $i \in [m]$. With probability at least $1-\delta$ over V, there exists ψ', u', satisfying $L(\psi', u') \leq \epsilon$ and $\|u'\|_2^2 \leq O(T^2 \delta^{-1})$.*

Proof of Theorem G.6. Since the number of possible input in pre-training is finite, we can invoke Lemma 9 in Bai & Lee (2020) to show that random Gaussian features can fit the pre-training task.

Lemma G.7 (Lemma 9 in Bai & Lee (2020))**.** *Suppose $\|h\|_2 = \sqrt{\frac{1}{T}}$, $v \sim \mathcal{N}(0, TI_{2T})$. There exists a random variable $a(v)$ such that*

$$\mathbb{E}[\sigma(v^\top h)a] = -\sqrt{T}\mathbf{1}^\top h$$

and a satisfies $\mathbb{E}_v[a^2] = O(T^2)$.

Consider $Q = K = 0$. In this case we have $h_{Q,K}(x) = \frac{1}{T}\sum_{j=1}^T x_j$. Also note that for the pre-training task, $y = -\sqrt{T}\mathbf{1}^\top h_{Q,K}(x)$ for all x. Since x_j are norm 1 orthogonal to each other, we have $\|h_{Q,K}(x)\|_2 = \frac{1}{T}$ for all x. Now we can show that $V_i \sim \mathcal{N}(0, TI_{2T})$, $i \in [m]$ independently is the random feature solution which can solve the pre-training task.

Suppose $g(h) = \frac{1}{m}\sum_{r=1}^m \sigma(V_r^\top h)a(V_r)$, and $g_{(R)}(h) := \frac{1}{m}\sum_{r=1}^m \sigma(V_r^\top h)a(V_r)\mathbb{I}(\|V_r\|_2 \leq \sqrt{T}R)$. R is large enough such that $\Pr\left(\sup_{r\in[m]}\|V_r\|_2 \geq \sqrt{T}R\right) \geq 1 - \delta/2$. We have $g(h) = g_{(R)}(h)$ on this event. Let $g^*_{(R)}(h)$ be the truncated version of $g^*(h) = -\sqrt{T}\mathbf{1}^\top h$, $g^*(h)_{(R)} = \mathbb{E}_v[\sigma(v^\top h)a(v)\mathbb{I}(\|v\|_2 \leq \sqrt{T}R)]$. We have

$$\mathbb{E}_V[(g(h) - g^*(h)_{(R)})] \leq \frac{1}{m}\mathbb{E}_v[\sigma(v^\top h)^2 a(v)^2 \mathbb{I}(\|v\|_2 \geq \sqrt{T}R)] \leq C\frac{R^2T^2}{m}.$$

By Chebyshev and a union bound, we have

$$\Pr\left(\max_h |g(h) - g^*(h)_{(R)}| \geq t\right) \leq C\frac{n(h)R^2T^2}{mt^2}.$$

For $t = \frac{\epsilon}{2}$, we have $m \geq n(h)R^2T^2\epsilon^{-2}$.

$$\begin{aligned}
|g^*_{(R)}(h) - g^*(h)| &= \mathbb{E}_v[\sigma(v^\top h)a(v)\mathbb{I}(\|v\|_2 \geq \sqrt{T}R)] \\
&\leq \mathbb{E}_v[a(v)^2]^{\frac{1}{2}}\mathbb{E}_v[\sigma(v^\top h)^4]^{\frac{1}{4}}\Pr\left(\|v\|_2 > \sqrt{T}R\right)^{\frac{1}{4}} \\
&\leq CT\Pr\left(\|v\| > \sqrt{T}R\right)^{\frac{1}{4}}.
\end{aligned}$$

Choosing $R = \tilde{O}(\sqrt{(T)})$ will make $\Pr\left(\|v\| > \sqrt{T}R\right)^{\frac{1}{4}} \leq \frac{c\epsilon}{T}$. Also note that $n(h) = (T/2-1)\binom{T}{T/2+1}$. Thus $m \geq \tilde{O}(2^T T^3 \epsilon^{-2})$ suffices. □