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

Hong Liu **Sang Michael Xie** **Zhiyuan Li** **Tengyu Ma**
Stanford University
{hliu99, sxie, zhiyuanli, tengyuma}@stanford.edu

Abstract

Language modeling on large-scale datasets leads to impressive performance gains on various downstream tasks. The validation pre-training loss (or perplexity) is often used as the evaluation metric when developing language models since the pre-training loss is correlated with downstream performance. 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. On simplified datasets, we identify three ways to produce models with the same pre-training loss but different downstream performance: continuing pre-training after convergence, increasing the model size, and changing the training algorithm. These experiments demonstrate the existence of implicit bias of pre-training algorithms—among models with the same pre-training loss, they implicitly prefer more transferable ones. We prove that SGD prefers flatter minima in language models, and empirically observe a strong correlation between flatness and downstream performance among models with the same pre-training loss.

1 Introduction

Large language models (LLMs) trained on internet-scale data have improved performance on a wide array of downstream tasks [12, 53, 37, 38, 7]. These models are trained with a language modeling pre-training loss to "fill in the blanks"—either predicting the next token/word (autoregressive lanugage modeling loss, or perplexity) or masked tokens (masked language modeling (MLM) loss). The *validation* pre-training loss is used to monitor the training process [7, 55] and compare different models since the pre-training loss is generally strongly correlated with downstream performance [19]. Theoretical works on understanding LLMs also focus on how the pre-training loss affects downstream performance. Saunshi et al. [40], Wei et al. [49], Xie et al. [52] show that good pre-training loss is a main reason for downstream success of LLMs.

In this paper, we question the correlation between the validation pre-training loss and downstream performance for language modeling. We find that different parameter configurations with the same pre-training loss can have different downstream performance, especially when the pre-training loss reaches a near-optimal level. Concretely, we find three situations that demonstrate such a phenomenon: (1) Even after the pre-training loss converges, models at a later time step still tend to perform better. (2) Models trained by standard algorithms have better performance than adversarially trained models with the same pre-training loss. (3) Larger models tend to perform better downstream than smaller models even if they have the same pre-training loss. These situations are most prominent in the *saturation regime*, where the models are close to the minimal possible pre-training loss (aka the entropy of the conditional probability). In the saturation regime, the pre-training loss of all models are almost the same, but the transferability to downstream tasks varies.

In each of the first two cases above, we find two models with the same pre-training loss and the same architecture; but one has a better performance than the other. They only differ by the training algorithms. This suggests the training algorithms have an *implicit bias* toward one of these 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

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

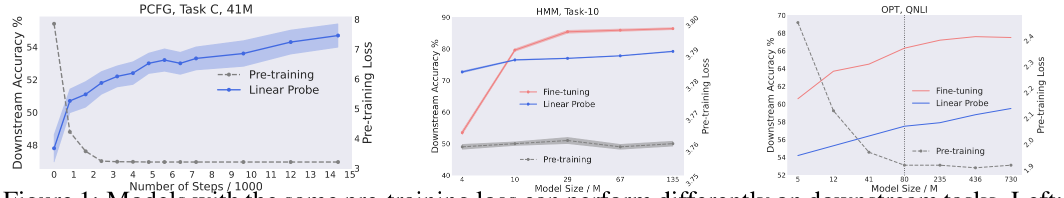


Figure 1: Models with the same pre-training loss can perform differently on downstream tasks. Left: Models at a later time step performs better, even after the pre-training loss converges. Middle and Right: Larger models perform better than smaller models, even with almost the same pre-training loss.

hypothetical large model that represents the smaller model with worse performance (by padding zeros to 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.

In supervised settings, the implicit bias selects generalizable models among all models with small *empirical* loss [11, 29]. However, the role of implicit bias in language modeling has not been studied. The gap between empirical and population losses is typically small in language modeling. The implicit bias selects local minima of the *population* self-supervised loss that transfer better to downstream tasks.

In Section 3, we provide a first-cut theoretical analysis of the implicit bias in 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 in [11, 29], 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 cases above, flatness of the model (measured by the trace of Hessian of the loss) strongly correlates with the downstream performance.

2 The Existence of Implicit Bias in Language Modeling

We investigate the relationship between pre-training loss and downstream performance, and find out that models with the same pre-training loss but different pre-training procedures can have different downstream performance.

2.1 Formulations

Masked language modeling. Consider a vocabulary $W=\{0,1,\ldots,c\}$, where 0 is a special token for the mask. Let $x=[x_1,\ldots,x_T]$ denote the input sequence, and $x_{-t}=[x_1,\ldots,x_{t-1},0,x_{t+1},\ldots,x_T]$ denote the masked sentence, where t is sampled uniformly randomly and independently from $[T]$.[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 $\Pr(\cdot\,|\,x_{-t}) := [\Pr(x_t=1\,|\,x_{-t}),\ldots,\Pr(x_t=c\,|\,x_{-t})]\in\mathbb{R}^c$. The model $f_\theta(\cdot)$ 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, $L(\theta)=\mathbb{E}_{x,t}[\ell(f_\theta(x_{-t}),x_t)]=\mathbb{E}_{x,t}[-\log([f_\theta(x_{-t})]_{x_t})]$. On downstream tasks, we use a randomly initialized linear classfier on top of the contextual representations. In fine-tuning, both of them are trained, while in linear probe, only the classifier is updated.

Saturation regime. We introduce the saturation regime, where the model output equals the true conditional probability, $f_\theta(x_{-t})=\Pr(\cdot\,|\,x_{-t})$. The MLM loss is equal to the entropy of the true conditional probability $L(\theta)=\mathbb{E}_{x,t}[-\log(\Pr(x_t\,|\,x_{-t}))]=\frac{1}{T}\sum_{t=1}^{T}H(x_t\mid x_{-t})$. *All* models in the saturation regime have the same, optimal pre-training loss, and we will show that they behave differently on downstream tasks. Our experiments use expressive enough architectures such that we can enter saturation regime for our simplified datasets. For real large-scale data, it is currently computationally challenging to arrive at the saturation regime. We hope that our experiments can provide insights for even larger models in the future and for other regimes where pre-training loss does not explain downstream performance.

2.2 Experimental Setup

We design controlled experiments to study the correlation between pre-training loss and downstream performance. In particular, we will 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 training algorithms. More details and additional results are provided in Section A.

[1] For simplicity, we only consider masking out one one token in each sentence.

Datasets. We introduce three generative models PCFG [8], HMM and OPT [55] to produce simplified datasets, with which we can study various factors systematically. With the knowledge of the true generative models that generate the data, 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. For PCFG and HMM, the downstream task is to classify the hidden variables. For OPT, we use QNLI and SST-2 from GLUE [46] as downstream tasks. The true conditional probability can be computed efficiently for the three datasets given the knowledge of the generated models, as detailed in Section A.

Models and algorithms. For PCFG and OPT generated datasets, we use transformers. For the HMM generated dataset, we use LSTM. In pre-training, all the models are pre-trained with AdamW. For comparison, we also consider other training algorithms. The first adversarial algorithm is inspired by Liu et al. [31], Raghu et al. [39], where the models are pre-trained with an additional meta-learning objective which messes up downstream performance. The second algorithm is manually setting the weight of the model to represent a lookup table which memorizes all the masked sentences and the corresponding true conditional probability. When the transformer is sufficiently large, the lookup table can be encoded as shown in Yun et al. [54].

2.3 Results

We compare the downstream performance of models with the same pre-training loss in the following situations: (2) training for different number of steps after the pre-training loss converges, (2) using different model sizes, and (3) training with normal training algorithms vs. adversarial training algorithms.

In Figure 1 Left, we plot the validation pre-training loss and the downstream performance of different models checkpoints along pre-training. After the pre-training loss converges, although the pre-training loss does not improve, the downstream accuracy continues increasing.

In Figure 1 Middle and Right, 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, which is 3.758 and 1.865 for HMM and OPT respectively. For OPT-generated dataset, we use the vertical dashed line to indicate the place where the pre-training loss saturates as we scale up the model. For the much simpler HMM, the smallest model can fit pre-training close to the entropy of the true conditional probability. With the same pre-training loss, scaling up the models improves linear probe performance by 4.5% and 2.0%, on HMM, and OPT generated data.

Table 1: Different pre-training algorithms on PCFG.

Method	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

Naturally trained transformers are better than adversarially trained ones. 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 normally trained 235M transformer, it is more than 6% worse than the normal 235M model, and even worse than a normal 9M model on the downstream task B. The lookup table has perfect pre-training loss, but it performs the worst.

The experiments above 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 of the optimizer affect the downstream performance. This indicates the *implicit bias* of the training algorithms toward choosing more transferable models among those with the same pre-training loss and architecture.

3 Implicit Bias Leads to Flat Solutions in Language Modeling

We show that mini-batch SGD can find models in the flatter areas of pre-training loss landscape. The flatness is measured by the trace of Hessian of the pre-training loss $\mathrm{Tr}[\nabla^2 L(\theta)]$.

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. Let η be the learning rate and let θ_k^η denote the parameter at step k. We analyze the process starting from a global minimizer $\overline{\theta}$. At each iteration, we get a fresh sample (x,t), where x is a sentence and t is the position of the mask, and update the parameter θ by $\theta_{k+1} = \theta_k - \eta \nabla_\theta \ell(f_{\theta_k}(x_{-t}), x_t)$. As a regularity condition following prior works [14, 29, 3], 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 $\mathcal{C}^3$-smooth function, and 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 $\theta \in \Gamma, \mathrm{rank}(\nabla^2 L(\theta)) = M$.*

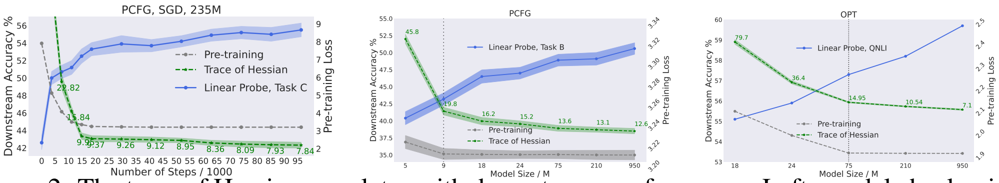


Figure 2: The trace of Hessian correlates with downstream performance. Left: model checkpoints with different number of steps after the pre-training loss converges. Middle and Right: models with different sizes and almost the same pre-training loss.

Even if the language 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 *non-zero* at the global minimizers and the stochastic gradient variance is also non-zero. Therefore, the dynamics of SGD do not completely stop; instead, the iterate oscillates around the manifold Γ. The following lemma shows the covariance of stochastic gradient for language models equals the Hessian of pre-training loss.

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

With Lemma 3.2, we can invoke Corollary 5.2 of Li et al. [29] to derive the following theorem: SGD will locally decrease the trace of Hessian along the solution of ordinary differential equation (1).

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

where $\nabla_\Gamma = P_\Gamma^\perp \nabla$ is the Riemannian gradient on manifold Γ. In other words, the ODE (1) is essentially a projected gradient descent algorithm with loss function $\mathrm{Tr}[\nabla^2 L(\theta)]$, the constraint set Γ, and infinitesimal learning rate. We show 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 $\eta \to 0$.*

4 Flatter Models Have Better Downstream Performance

Evaluation of flatness. As in Theorem 3.3, we measure the flatness of different models by the trace of Hessian of the pre-training loss. For computational feasibility, we adopt a technique inspired by Wei et al. [48] to unbiasedly estimate the trace of Hessian. Details are provided in Section B.

Results. In Figure 2 Left, we compare the downstream accuracy and the trace of Hessian of checkpoints at different times in pre-training. The trace of Hessian demonstrates a decreasing trend after the validation pre-training loss converges. Furthermore, as the trace of Hessian decreases, the downstream performance improves by 1.6% on the PCFG 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, corresponding to a drop of 5.5% in downstream performance.

Table 2: A 235M transformer pre-trained with different algorithms evaluated on PCFG Task C.

Method	Pre-training	Acc %	Tr(H)
AdamW	3.20	55.7±0.6	8.01±0.73
Adversarial	3.20	50.2±1.0	19.34±0.92

In Figure 2 Middle and Right, we compare the downstream accuracy and the trace of Hessian of models with different sizes. On the datasets generated by OPT, the pre-training loss is almost the same for models larger than 75M. As we increase the model size, the trace of Hessian of the pre-training loss decreases from 14.9 to 7.1, correlating with the increase of linear probe accuracy from 57.3% to 59.7%. The same observation hols on the PCFG-generated datasets.

Interaction between implicit bias and model size. Intuitively, 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 family (as justified in Section B). Thus the flattest minimum found within a larger transformer is flatter than the flattest minimum found within a smaller transformer, and performs better downstream.

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

[1] Sanjeev Arora, Nadav Cohen, Wei Hu, and Yuping Luo. Implicit regularization in deep matrix factorization. In *Advances in Neural Information Processing Systems*, pp. 7411–7422, 2019.

[2] Sanjeev Arora, Hrishikesh Khandeparkar, Mikhail Khodak, Orestis Plevrakis, and Nikunj Saunshi. A theoretical analysis of contrastive unsupervised representation learning. In *International Conference on Machine Learning*, 2019.

[3] Sanjeev Arora, Zhiyuan Li, and Abhishek Panigrahi. Understanding gradient descent on the edge of stability in deep learning. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (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.

[4] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

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

[6] Guy Blanc, Neha Gupta, Gregory Valiant, and Paul Valiant. Implicit regularization for deep neural networks driven by an ornstein-uhlenbeck like process. *arXiv preprint arXiv:1904.09080*, 2019.

[7] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

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

[9] Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*, 2020.

[10] Zihang Dai, Guokun Lai, Yiming Yang, and Quoc Le. Funnel-transformer: Filtering out sequential redundancy for efficient language processing. *Advances in neural information processing systems*, 33:4271–4282, 2020.

[11] Alex Damian, Tengyu Ma, and Jason D Lee. Label noise sgd provably prefers flat global minimizers. *Advances in Neural Information Processing Systems*, 34:27449–27461, 2021.

[12] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[13] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

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

[15] Suriya Gunasekar, Jason D Lee, Daniel Soudry, and Nati Srebro. Implicit bias of gradient descent on linear convolutional networks. In *Advances in Neural Information Processing Systems*, pp. 9461–9471, 2018.

[16] Jeff Z HaoChen, Colin Wei, Jason D Lee, and Tengyu Ma. Shape matters: Understanding the implicit bias of the noise covariance. *arXiv preprint arXiv:2006.08680*, 2020.

[17] Jeff Z HaoChen, Colin Wei, Adrien Gaidon, and Tengyu Ma. Provable guarantees for self-supervised deep learning with spectral contrastive loss. *arXiv preprint arXiv:2106.04156*, 2021.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[19] Danny Hernandez, Jared Kaplan, Tom Henighan, and Sam McCandlish. Scaling laws for transfer. *arXiv preprint arXiv:2102.01293*, 2021.

[20] John Hewitt and Christopher D Manning. 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.

[21] Phu Mon Htut, Jason Phang, Shikha Bordia, and Samuel R Bowman. Do attention heads in bert track syntactic dependencies? *arXiv preprint arXiv:1911.12246*, 2019.

[22] Peter Izsak, Moshe Berchansky, and Omer Levy. How to train bert with an academic budget. *arXiv preprint arXiv:2104.07705*, 2021.

[23] Ziwei Ji and Matus Telgarsky. Gradient descent aligns the layers of deep linear networks. *arXiv preprint arXiv:1810.02032*, 2018.

[24] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[25] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2019.

[26] Hae Beom Lee, Hayeon Lee, Donghyun Na, Saehoon Kim, Minseop Park, Eunho Yang, and Sung Ju Hwang. Learning to balance: Bayesian meta-learning for imbalanced and out-of-distribution tasks. In *International Conference on Learning Representations*, 2020.

[27] Yuanzhi Li, Tengyu Ma, and Hongyang Zhang. Algorithmic regularization in over-parameterized matrix sensing and neural networks with quadratic activations. *arXiv preprint arXiv:1712.09203*, pp. 2–47, 2017.

[28] Zhiyuan Li, Yuping Luo, and Kaifeng Lyu. Towards resolving the implicit bias of gradient descent for matrix factorization: Greedy low-rank learning. *arXiv preprint arXiv:2012.09839*, 2020.

[29] Zhiyuan Li, Tianhao Wang, and Sanjeev Arora. What happens after sgd reaches zero loss?–a mathematical framework. *arXiv preprint arXiv:2110.06914*, 2021.

[30] Hanxiao Liu, Zihang Dai, David So, and Quoc V Le. Pay attention to mlps. *Advances in Neural Information Processing Systems*, 34:9204–9215, 2021.

[31] Hong Liu, Jeff Z HaoChen, Colin Wei, and Tengyu Ma. Meta-learning transferable representations with a single target domain. *arXiv preprint arXiv:2011.01418*, 2020.

[32] Kaifeng Lyu and Jian Li. Gradient descent maximizes the margin of homogeneous neural networks. *arXiv preprint arXiv:1906.05890*, 2019.

[33] Jonathan Mamou, Hang Le, Miguel Del Rio, Cory Stephenson, Hanlin Tang, Yoon Kim, and SueYeon Chung. Emergence of separable manifolds in deep language representations. *arXiv preprint arXiv:2006.01095*, 2020.

[34] Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. Rethinking the role of demonstrations: What makes in-context learning work? *arXiv preprint arXiv:2202.12837*, 2022.

[35] Maurice Nivat. 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.

[36] Matthew E Peters, Mark Neumann, Luke Zettlemoyer, and Wen-tau Yih. Dissecting contextual word embeddings: Architecture and representation. *arXiv preprint arXiv:1808.08949*, 2018.

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

[38] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:1–67, 2020.

[39] Aniruddh Raghu, Jonathan Lorraine, Simon Kornblith, Matthew McDermott, and David K Duvenaud. Meta-learning to improve pre-training. *Advances in Neural Information Processing Systems*, 34:23231–23244, 2021.

[40] Nikunj Saunshi, Sadhika Malladi, and Sanjeev Arora. A mathematical exploration of why language models help solve downstream tasks. *arXiv preprint arXiv:2010.03648*, 2020.

[41] Nikunj Saunshi, Jordan Ash, Surbhi Goel, Dipendra Misra, Cyril Zhang, Sanjeev Arora, Sham Kakade, and Akshay Krishnamurthy. Understanding contrastive learning requires incorporating inductive biases. *arXiv preprint arXiv:2202.14037*, 2022.

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

[43] Yi Tay, Dara Bahri, Donald Metzler, Da-Cheng Juan, Zhe Zhao, and Che Zheng. Synthesizer: Rethinking self-attention for transformer models. In *International conference on machine learning*, pp. 10183–10192. PMLR, 2021.

[44] Yi Tay, Mostafa Dehghani, Jinfeng Rao, William Fedus, Samira Abnar, Hyung Won Chung, Sharan Narang, Dani Yogatama, Ashish Vaswani, and Donald Metzler. Scale efficiently: Insights from pre-training and fine-tuning transformers. *arXiv preprint arXiv:2109.10686*, 2021.

[45] Ilya O Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, et al. Mlp-mixer: An all-mlp architecture for vision. *Advances in Neural Information Processing Systems*, 34: 24261–24272, 2021.

[46] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.

[47] Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

[48] Colin Wei, Sham Kakade, and Tengyu Ma. The implicit and explicit regularization effects of dropout. *arXiv preprint arXiv:2002.12915*, 2020.

[49] Colin Wei, Sang Michael Xie, and Tengyu Ma. Why do pretrained language models help in downstream tasks? an analysis of head and prompt tuning. *arXiv preprint arXiv:2106.09226*, 2021.

[50] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*, 2022.

[51] Blake Woodworth, Suriya Gunasekar, Jason D Lee, Edward Moroshko, Pedro Savarese, Itay Golan, Daniel Soudry, and Nathan Srebro. Kernel and rich regimes in overparametrized models. *arXiv preprint arXiv:2002.09277*, 2020.

[52] Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. An explanation of in-context learning as implicit bayesian inference. *arXiv preprint arXiv:2111.02080*, 2021.

[53] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32, 2019.

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

[55] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

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

[57] Yi Zhang, Arturs Backurs, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, and Tal Wagner. Unveiling transformers with lego: a synthetic reasoning task. *arXiv preprint arXiv:2206.04301*, 2022.

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 $y = \mathrm{argmax}_{s \in S} \Pr(s \mid x_1, x_2, \ldots, x_{1+L})$. 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 3.

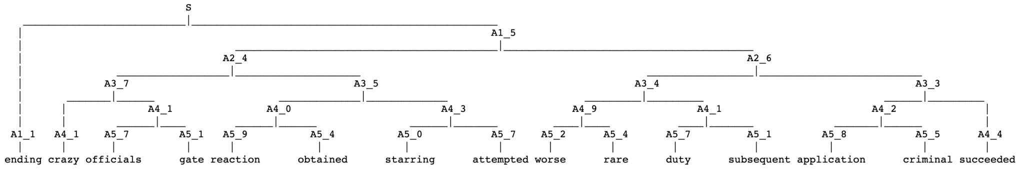

S
A1_5
A2_4
A2_6
A3_7
A3_5
A3_4
A3_3
A4_1
A4_0
A4_3
A4_9
A4_1
A4_2
A1_1
A4_1
A5_7
A5_1
A5_9
A5_4
A5_0
A5_7
A5_2
A5_4
A5_7
A5_1
A5_8
A5_5
A4_4
ending crazy officials gate reaction obtained starring attempted worse rare duty subsequent application criminal succeeded

Figure 3: 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 4.

```
</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 4: An example of the genearated OPT sentence.

A.2 Compute the True Conditional Probabilities

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

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

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 $\Pr(x_1, \ldots, x_T) = \Pr(x_1) \prod_{t=1}^{T-1} \Pr(x_{t+1} \mid x_1, \ldots, x_t)$.

A.3 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. [22]. 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].

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 Algorithms

The lookup table. To evaluate the downstream performance of the lookup table, we first create the lookup table with the data of the downstream task. With the method mentioned in Section A.2, we can generate the true conditional probability of each token and use it as the contextual embeddings.

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, $\text{minimize}_\psi L(\psi) - \lambda \frac{1}{|D_2|}\sum_{(x,y)\in D_2} \ell(g_{\hat{\phi}(\psi)}(h_\psi(x)),y)$. The optimization can be efficiently carried out with closed form solution of ϕ as shown in Liu et al. [31].

Fine-tuning. Following the standard protocol of Devlin et al. [12], 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.

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

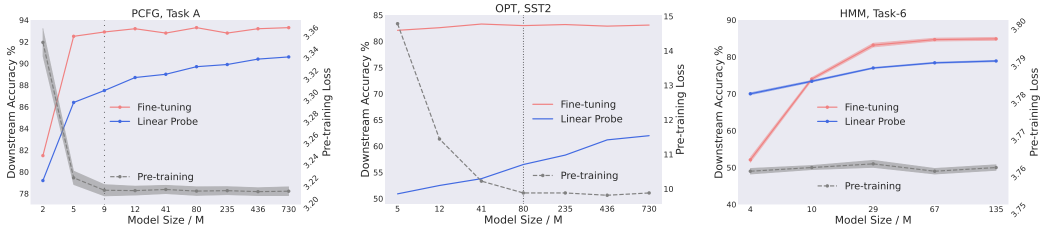
Evaluation of pre-training loss. Since we have access calculate the true conditional probability, we can calculate the cross entropy loss as the sum of the entropy of the true conditional probability and the KL divergence between the predicted and true conditional probabilities. This is more accurate 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.5 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.6 Evaluation of Pre-training Loss with KL Divergence.

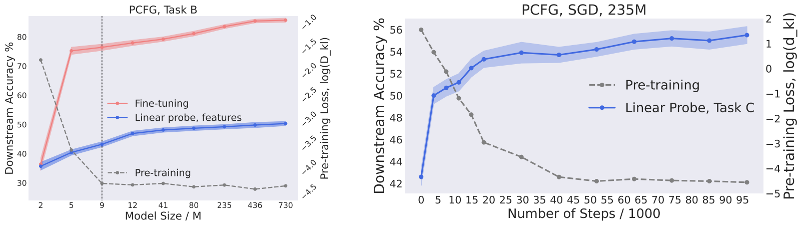
Since we have access to the true conditional probability from the generative models, we can decompose the cross entropy into the KL divergence between prediction and true conditional probability plus the entropy of the true conditional probability. When comparing different models in the saturation regime, we can use the KL divergence to reduce the variance of the loss evaluation. We provide the results with



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

Figure 5: The downstream accuracy continues to rise as we increase the model size, althought the pre-training loss remains unchanged.

KL divergence evaluation in Figure 6. 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 6: 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

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

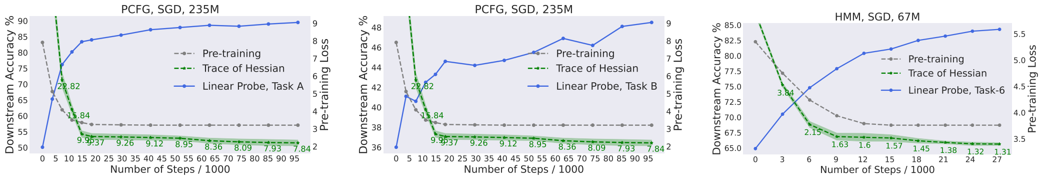
Therefore we have

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

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 2.

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 7.



(a) PCFG, SGD, Task A, 235M

(b) PCFG, SGD, Task B, 235M

(c) HMM, SGD, Task-6,67M

Figure 7: 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 ??.

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.

Table 4: Shape of the transformers in Figure **??**.

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

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 [4] and residual connections [18]. Consider a multi-layer MLP $f_{W,a}(x)$. The weight matrices are $W=[W_0,...,W_{L-1}]$.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 $\tilde{W}=[\tilde{W}_0,...,\tilde{W}_{L-1}]$ 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 $x=[x_1,...,x_T]$, 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 $\mathrm{Attn}_l(h_l(x))=[(A_{l1}h_l(x)V_{l1})^\top,...,(A_{ln_h}h_l(x)V_{ln_h})^\top]^\top O_l$. 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 $k \in [n_h]$, $A_{lk} =$ softmax$(h_l(x) Q_{lk} K_{lk}^\top h_l(x)^\top)$.

Following the implementation of Devlin et al. [12], 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 $[h_0(x)]_i = \text{LN}(W_E x_i)$. 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 $[v_l(x)]_i = \text{LN}([h_l(x)]_i + [\text{Attn}_l(h_l(x))]_i)$ and $[h_{l+1}(x)]_i = \text{LN}([v_l(x)]_i + U_l \sigma(W_l [v_l(x)]_i + b_l))$. 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 8).

C Additional Related Work

Language modeling and downstream adaptation. Large language modeling has revolutionized the NLP field. Starting from Devlin et al. [12], a line of works improve the downstream performance on a wide range of tasks with increasing model size and data amount [53, 37, 38, 7]. LLMs even exhibit unexpected emergent behaviors, such as in-context learning [52, 34], step-by-step reasoning [50], and zero-shot learning [7]. Kaplan et al. [24], Hernandez et al. [19] 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. In practice, the pre-training loss is used as an evaluation metric for language models. A notable example is the efficient transformer line of works, which benchmark the pre-training loss given the same computation constraint [10, 47, 9, 43, 30].

Understanding large language models. Empirical works on understanding MLM find out that the representations of language models encode rich semantic and syntactic information [36, 21, 20, 33]. Theoretical works show that good LM loss, or the ability to fit the LM conditional probability, is a sufficient condition for good performance on downstream tasks. Zhang & Hashimoto [56] show MLM representations recover latent variables in graphical models. Saunshi et al. [40] introduce the natural assumption, which states that downstream tasks can be solved linearly with the true conditional probability. Wei et al. [49] 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.

Recent works also provide empirical evidence that good pre-training loss cannot fully explain the downstream success. Tay et al. [44] find out that a narrow but deep transformer is better than a wide but shallow transformer with the same pre-training loss. Zhang et al. [57] demonstrate that Albert [25] generalizes better to OOD tasks than Bert on a synthetic reasoning task. These works indicate that the architecture is an important factor for good downstream performance beyond pre-training loss. This paper discovers the implicit bias in language modeling on models with the same architecture and the same pre-training loss. Similar to our findings, Xie et al. [52] 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. [2] prove that contrastive learning representations work on downstream linear classification tasks. Lee et al. [26] 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. [17] 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. [41] 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 [18] can have better downstream performance on downstream tasks than ViT [13] and MLP-Mixer [45] 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. The training algorithm chooses solutions with certain properties, and usually leads to better generalization [15, 42, 27, 23, 1, 32, 28, 51, 16]. Recently, Blanc et al. [6], Damian et al. [11], Li et al. [29] 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.

D Omitted Proofs in Section 3

Proof of Lemma 3.2. We first recall loss

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

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

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

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

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

which completes the proof. □

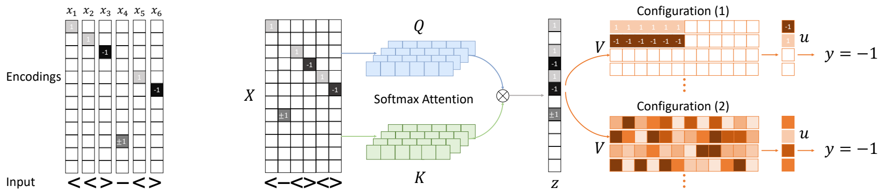


Figure 8: **The synthetic language setting.** Left: An example of input encodings with sentence length $T=6$. Right: Illustration of the two solutions. The softmax attention can sum the token encodings into z. Solution (1) contains two features transferable to the downstream task. The neurons in solution (2) are sampled from Gaussian distribution, and not related to the downstream task. Both solutions can output the correct prediction for MLM pre-training, but solution (1) has much smaller trace of Hessian.

E Flatness Regularization Provably Identifies Transferable Models on Synthetic Language

Toward formally proving the connection between the flatness regularization (introduced by the stochastic gradient as argued in Section 3) and the 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 essentially memorize the conditional probability using random features. We will prove that the flattest model is the desired model that transfers to downstream tasks.

Pre-training Distribution. Consider a variant of the Dyck language [35] 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. Then, we randomly select one position in $[T]$ and replace the bracket with a mask token.

Downstream Task. The most intrinsic property about the synthetic language is the difference in the number of left and right brackets in a prefix, 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 of sentences which do not necessarily belong to the the language. 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 $v \sim \text{Unif}(\{\pm e_{t+T}\})$. Examples of the encodings with $T=6$ are provided in Figure 8 (Left). 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 (See Figure 8 (Right)). 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 F.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 E.1.

Consider minimizing the trace of Hessian among all the solutions to the MLM pre-training task: $\text{minimize}_{\psi,u} \operatorname{Tr}[\nabla_\psi^2 L(\psi,u)] + \operatorname{Tr}[\nabla_u^2 L(\psi,u)]$, subject to $L(\psi,u) = 0$.

Theorem E.1. *Suppose $m \geq 2$ and $T \geq 6$. The flattest solution $\hat{\psi}, \hat{u}$ are defined as the solution of the optimization problem above. $\tilde{u}$ is the minimizer of downstream training loss on top of $\hat{\psi}$, $\tilde{u} \in \operatorname{argmin}_u \|u\|_2$ subject to $\widehat{L}^{P_{\text{ds}}}(\hat{\psi},u) = 0$. Then with probability at least $1 - 2^{-n}$, $L^{P_{\text{ds}}}(\hat{\psi},\tilde{u}) = 0$.*

F Omitted Proofs in Section E

F.1 Omitted Proofs of Theorem E.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 $L^{P_{\text{ds}}}(\psi,u) = \mathbb{E}_{x\sim P_{\text{ds}}}[(f_{\psi,u}(x) - g^*(x))^2]$.

Proof of Theorem E.1. We first calculate the trace of Hessian of the pre-training loss and then derive a lower bound for it in Lemma F.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 F.3. This translates to constant sample complexity for the downstream task.

Lemma F.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 E.1, $I_+ = \{i \in [m] \mid u_i > 0\}$ and $I_- = \{i \in [m] \mid u_i < 0\}$. The trace of Hessian can be lower bounded,*

$$\mathrm{Tr}[\nabla^2_\psi L(\psi,u)] + \mathrm{Tr}[\nabla^2_u L(\psi,u)] \geq \sqrt{\frac{4}{T}},$$

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

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

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

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

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, $I_x = \{i \in [T] \mid V_i^\top x > 0\}$. 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, $H_+ = \{h_{Q,K}(x) \mid y = 1\}$ and $H_- = \{h_{Q,K}(x) \mid y = -1\}$. 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 F.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 F.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 F.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 F.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 F.3. *Suppose Q,K,V satisfy the equality in Lemma F.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 F.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,*

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

Proof of Lemma F.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 F.1.

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

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 F.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 F.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 F.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 F.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 F.2, we know that ${V_i}^\top(-h'') \leq 0$ and ${V_i}^\top(-\tilde{h}'') \leq 0$, this implies that $-{V_i^{(c)}}_1 \leq {V_i^{(p)}}_2 \leq {V_i^{(c)}}_1$. 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 F.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 $i \geq 2$, $M_{i,\rho(j)} = M_{i-1,j}$. Then the rank of M is $2k+1$.*

Proof of Lemma F.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 $2k+1$. □

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

Theorem F.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 F.6. Since the number of possible input in pre-training is finite, we can invoke Lemma 9 in Bai & Lee [5] to show that random Gaussian features can fit the pre-training task.

Lemma F.7 (Lemma 9 in Bai & Lee [5])**.** *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. □