Solo Connection: A Parameter Efficient Fine-Tuning Technique for Transformers

Anonymous Authors[1]

Abstract

Parameter-efficient fine-tuning (PEFT) is a versatile and extensible approach for adapting a Large Language Model (LLM) for newer tasks. One of the most prominent PEFT approaches, Low-Rank Adaptation (LoRA), primarily focuses on adjusting the attention weight matrices within individual decoder blocks of a Generative Pre-trained Transformer (GPT-2). In contrast, we introduce Solo Connection—a novel method that adapts the representation at the decoder-block level rather than modifying individual weight matrices. Not only does Solo Connection outperform LoRA on E2E natural language generation benchmarks, but it also reduces the number of trainable parameters by 59% relative to LoRA and by more than 99% compared to full fine-tuning of GPT-2 one of the earliest versions of large language models (LLMs). Another key motivation for Solo Connection comes from homotopy theory, where we introduce a trainable linear transformation that gradually interpolates between a zero vector and the task-specific representation, enabling smooth and stable adaptation over time.

While skip-connections in the original 12-layer GPT-2 are typically confined to individual decoder blocks, subsequent GPT-2 variants scale up to 48 layers, and even larger language models can include 128 or more decoder blocks. These expanded architectures underscore the need to revisit how skip connections are employed during fine-tuning. This paper focuses on "long skip connections" that link outputs of different decoder blocks, potentially enhancing the model's ability to adapt to new tasks while leveraging pre-trained knowledge.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

1. Introduction

Pre-trained Language Models (PLMs) like GPT-2 (Radford et al., 2019), GPT-3, GPT-4, LLAMA-2 (Touvron et al., 2023), and Transformer-XL (Dai et al., 2019) have transformed NLP by leveraging self-supervised objectives such as language modeling. These models predict the next token given a sequence, enabling them to generate coherent text. Despite their success, adapting large PLMs to new domains remains resource-intensive, limiting accessibility for research groups with constrained compute and memory.

Parameter-efficient fine-tuning (PEFT) tackles this challenge by adapting PLMs using a small subset of parameters (Xu et al., 2023). Techniques like LoRA (Hu et al., 2021), BitFit (Ben Zaken et al., 2022), and Adapters (Houlsby et al., 2019a) have shown that models can retain strong performance without full fine-tuning. These methods are effective in low-resource settings and have been used in NLP, vision, and multi-modal tasks (LeCun et al., 2015; Houlsby et al., 2019b; Pathak & Paffenroth, 2021).

We propose *Solo Connection*, a sparse and low-rank fine-tuning strategy inspired by LoRA. Instead of modifying weights directly, it leverages long skip connections within decoder blocks to adapt model representations efficiently. Our approach emphasizes parameter sharing and sparsity, reducing the number of trainable parameters while preserving task performance. The core motivation behind *Solo Connection* is to fundamentally shift the paradigm of parameter-efficient fine-tuning (PEFT) from *intra-layer adaptation* to *inter-layer adaptation*. While it may be loosely described using the language of adapters, such a view oversimplifies its theoretical foundation and architectural design. Unlike most adapter-based approaches—including LoRA and its variants—which focus on inserting adaptation modules within specific subcomponents like attention or feedforward layers, Solo Connection operates across layers. It introduces weighted skip connections that span multiple decoder blocks, allowing information to flow more effectively and enabling parameter sharing across a broader context. This cross-layer mechanism supports richer representation learning while significantly reducing the number of trainable parameters. Another key motivation for Solo Connection stems from homotopy theory, rooted in continuation meth-

Model	#Params	BLEU	NIST	E2E MET	Rouge	CIDEr
FT GPT-2 M	354.92M	68.2	8.62	46.2	71.0	2.47
Baseline LoRA GPT-2 Medium	0.35M	67.45	8.58	45.93	68.8	2.36
Solo Connection GPT-2 Medium	0.26M	67.7	8.64	45.95	69.13	2.36
Baseline LoRA GPT-2 Small	0.29M	65.79	8.49	45.21	67.46	2.27
Solo Connection GPT-2 S	0.12M	67.64	8.64	45.70	68.32	2.31

Table 1. Performance comparison of GPT-2 Medium (GPT-2 M) and GPT-2 Small (GPT-2 S) models using three approaches: fully fine-tuned (FT), LoRA, and our proposed Solo Connection. While the full fine-tuning of GPT-2 M requires 354.92M parameters, Solo Connection uses only 0.26M but still achieves comparable or superior results across NLG metrics compared to other methods. Solo Connection has 99% less trainable parameters, demonstrating its efficiency and effectiveness in reducing parameter counts without sacrificing performance.

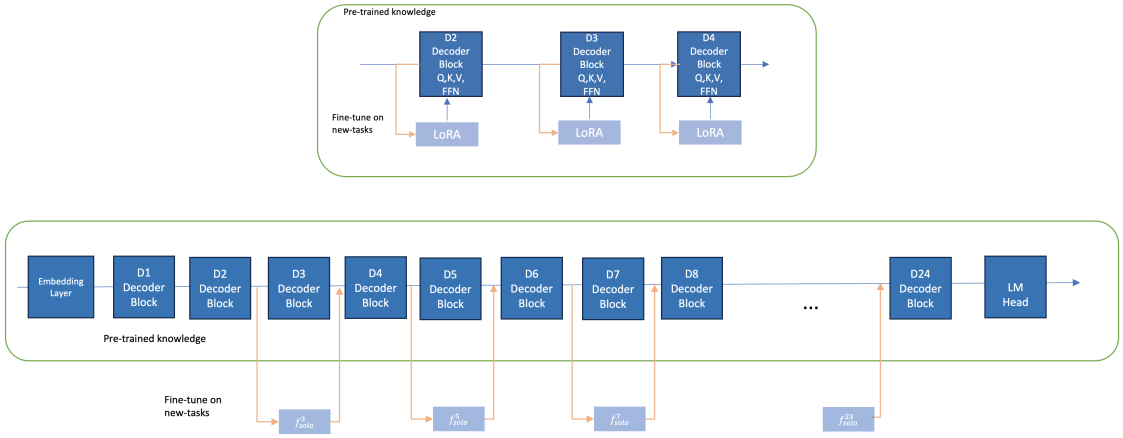

Figure 1. Solo Connection compared with LoRA setup

LoRA (top) is Intra-decoder connection: Fine-tuning large language models with LoRA involves adding trainable modules within each decoder layer, creating an intra-decoder connection. Solo Connection (bottom), on the other hand, introduces an inter-decoder connection where the shared encoder and decoder are connected across different decoder layers. This approach directly adapts the representation of the decoder blocks and we explore its potential in learning newer tasks.

ods (Hershey et al., 2024; Allgower & Georg, 2003; Pathak & Paffenroth, 2021; **?**) and dynamical systems (Allgower & Georg, 2003; Strogatz). To enable gradual and stable adaptation, we introduce a homotopy-inspired (Pathak, 2018; Pathak & Paffenroth, 2019) linear transformation that interpolates between a zero vector and the task-specific representation. This mechanism is governed by trainable parameters that control the extent of adaptation over time. Unlike abrupt modifications to network weights, this smooth transition facilitates progressive learning and inherently stabilizes training. It also acts as an implicit gating mechanism, scaling the Solo Connection output dynamically without manual tuning, offering a principled alternative to heuristic scaling used in methods like LoRA. The key contributions of the paper are as follows:

1. We introduce a novel PEFT method using weighted long skip connections across decoder blocks to reduce redundancy.
2. Solo Connection adapts representations more effectively, achieving better performance with fewer parameters.
3. On the E2E benchmark, our method outperforms LoRA and full fine-tuning while using 59% fewer parameters.
4. We analyze individual design components of Solo Connection in Appendix-C to explain its performance gains.

We evaluate Solo Connection on GPT-2 across five natural language generation (NLG) tasks (Novikova et al., 2017), showing that it consistently outperforms LoRA with up to 59% fewer parameters. While our experiments use transformer-based models, Solo Connection is architecture-agnostic and applicable across domains.

2. Methodology

Fine-tuning in (LeCun et al., 2015) is an essential technique in Deep learning to utilize the pre-trained knowledge for a downstream task. Solo Connection provides a unique per-

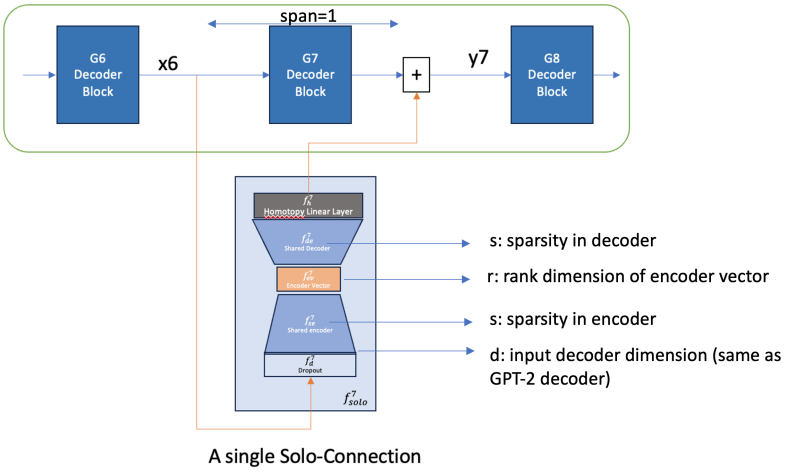

Figure 2. Solo Connection that transforms the representation from a previous decoder and adapts during fine-tuning for subsequent decoders. Here representation embedding of D6 is fine-tuned for D_8. Here we also show the building block of the solo connections, which has encoder-decoder learnable weights along with a homotopy layer.

spective over well-adopted PEFT methods such as Adapters (Houlsby et al., 2019a), LoRA (Hu et al., 2021) and its derivatives (Xu et al., 2023; Zhang et al., 2023; Kopiczko et al., 2024). Our method aims to adapt one or more predetermined decoder blocks. Intuitively, Solo Connection provides task-specific decoder representations for downstream tasks. The principles outlined here apply to any dense layers in deep learning models, though we only focus on GPT-2 language models in our experiments as the motivating use case. Specifically, for the case of GPT-2 (Radford et al., 2019), we observed that most LoRA-based techniques were applied to a weight-matrix; for example, it is used in the Query Q, Key K, or Value V weight matrix. *On the other hand, our method attempts to adapt the decoder block representation directly.*

2.1. The Solo Connection: Sparse and Low-rank Skip Connections

We introduce Sparse and Low-rank Skip Connection (Solo Connection) as a trainable block that can be applied to fine-tune Pre-trained Language Models (PLMs). In GPT-2, a Solo Connection is added to select decoder blocks (e.g., D_7 in Figure-2). It transforms the input (x_6) from a previous decoder block (D_6) before passing it to a subsequent block (D_8). Equation-1 defines the Solo Connection:

$$y_i = D_i(x_{i-1}, \theta_i) + f^i_{\text{solo}}(x_{i-1}, \phi_i) \tag{1}$$

Here, $D_i(x_{i-1}, \theta_i)$ represents the *fixed* pre-trained knowledge, while $f^i_{\text{solo}}(x_{i-1}, \phi_i)$ is the *adaptable* component for downstream tasks. θ_i and ϕ_i are the pre-trained and Solo Connection parameters, respectively. $x_{i-1} \in \mathbb{R}^d$ is the input from the previous decoder block, and $y_i \in \mathbb{R}^d$ is the output for the next block, where d is the representation dimension (1024 for GPT-2 M, 768 for GPT-2 S). Figure-2 details a single Solo Connection. By applying Solo Connections to alternate decoder blocks (Figure-1), we aim to optimize downstream task performance. We test this hypothesis in Section-3.

We apply Solo Connections to alternate decoder blocks in GPT-2, starting from D_2 to the final block. GPT-2 Medium (24 decoders) and GPT-2 Small (12 decoders) results in 11 and 5 Solo Connections, respectively. Figure-1 illustrates this configuration, while Figure-2 details a single Solo Connection as implemented in our experiments. This approach aims to balance adaptability and efficiency in fine-tuning.

2.2. The Building Block of Solo Connection

This section will describe the components of (f_{solo}) Solo Connection.

Equation-2 defines the composition of f_{solo}:

$$f_{\text{solo}} = f_h \circ f_{\text{sd}} \circ f_{\text{ev}} \circ f_{\text{se}} \circ f_d \tag{2}$$

where, f_h is the homotopy linear layer, f_{sd} is the shared decoder, f_{ev} is the encoding-vector, f_{se} is the shared encoder and f_d is the dropout layer.

The shared encoder (f_{se}) transforms the input vector $\mathbf{x}$ into a lower-dimensional representation of size r. This encoder is shared across all Solo Connection modules in the model (Figure 1). The same f_{se} function is applied to the inputs of each Solo Connection, regardless of its position in the network. For example, there are 5 Solo Connections in GPT-2 (S), then all 5 have a single trainable encoder f_{se}. We also add a non-trainable dropout layer f_d before the encoder to help generalize and improve metric performance.

The shared Encoder incorporates two key hyperparameters. The first is dimensionality reduction by rank (r), which transforms the input to a lower-dimensional space of rank (r). Second, sparsity (s) - A hyperparameter s randomly masks a fraction of parameters, setting $(s\%)$ to zero during both forward and backward passes.

The shared Decoder (f_{sd}) transforms the lower-dimensional representation back to the original input dimension. Like the shared encoder, the shared decoder is available across all Solo Connection modules. The dimensionality reduction factor r and sparsity hyperparameter s are crucial for balancing efficiency and performance. Their impact and selection strategies are discussed in Section 3. We use Kaiming initialization (He et al., 2015) for both the encoder and decoder functions in our method. This ensures the values are scaled based on the matrix dimensions, resulting in a consistent variance for all ranks when multiplying respective matrices. As a result, there is no need to fine-tune the learning rate for each rank Kopiczko et al. (2024); Hu et al. (2021). The shared encoder (f_{se}) and decoder (f_{sd}) are trainable, allowing for optimal encoding and decoding transformations. The encoding vector (f_{ev}) is a trainable task-specific bias vector, while the Homotopy Linear layer (f_h) is a trainable linear transformation for output refinement. The dropout layer (f_d), though not trainable itself, aids in preventing overfitting during the training of other components.

Finally, we employ a Homotopy linear layer to learn from projection ($\mathbf{z}$: output of f_{sd}) and gradually adapt to the new task. This homotopy linear layer (f_h) is a topological transformation between zero vector and the adapted representation $z \in \mathbb{R}$, as shown in Equation-3.

$$f_h(\mathbf{z}) = \lambda \mathbf{v} \odot \mathbf{z} + (1 - \lambda)\mathbf{0} \tag{3}$$

Here, λ (scalar) and $\mathbf{v}$ (vector) are trainable parameters, while $\mathbf{z}$ is the output of the shared decoder. Also, value of λ is bounded $[0, 1]$. The homotopy layer is the Solo connection's dynamic scaling and gating mechanism. As λ increases from 0 to 1, the Solo connection gradually adapts to new tasks by incorporating more of $\mathbf{z}$. Simultaneously, λ acts as an automatic scalar for the Solo connection's output, eliminating the need for manual scaling as required in methods like LoRA. The homotopy layer is initially set to 0.001, allowing for gentle, gradual adaptation during fine-tuning.

3. Experiments

We evaluate our fine-tuning approach by replicating the experimental setup of LoRA (Hu et al., 2021). All code and configurations are available on GitHub [Anonymous Link]. We use the E2E NLG Challenge dataset (Novikova et al., 2017), a benchmark with diverse NLG tasks—making it ideal for testing generalization. This diversity ensures models trained here are well-suited for transfer across domains. Also, we fine-tune GPT-2 Small and Medium (Radford et al., 2019) using our proposed *Solo Connection* and compare against full fine-tuning and LoRA baselines. GPT-2 models are widely used in both NLP and vision tasks, supporting their role as versatile backbones. For baselines, we use LoRA's original hyperparameters. In Solo Connection, we modify the rank and tune the learning rate. Our setup is consistent: one GPU, AdamW optimizer, batch size 4, and weight decay 0.1.

Table 1 reports BLEU, NIST, METEOR, ROUGE, and CIDEr scores (Hu et al., 2021; Zhang et al., 2023). Higher scores indicate better performance. FT GPT-2 M[1] is the fully fine-tuned GPT-2 Medium with 354.92M parameters. Due to resource limits, we use previously reported results from (Hu et al., 2021). We compare Baseline LoRA GPT-2 M (0.35M), Solo Connection GPT-2 M (0.26M), Baseline LoRA GPT-2 S (0.29M), and Solo Connection GPT-2 S (0.12M) in terms of trainable parameters.

Solo Connection consistently outperforms LoRA with fewer parameters. For GPT-2 Medium, it achieves a 99.93% reduction over full fine-tuning, and 25.71% over LoRA, with improved scores (+0.37% BLEU, +0.70% NIST, etc.). For GPT-2 Small, it cuts 58.62% of parameters versus LoRA, while improving all metrics (+2.82% BLEU, +1.76% CIDEr, etc.). These results highlight Solo Connection's efficiency in fine-tuning large language models for general-purpose tasks.

4. Conclusion

Large language models have demonstrated strong performance across NLP tasks, but adapting them to new domains remains computationally expensive—often inaccessible to labs with limited resources. To address this, we introduced *Solo Connection*, a parameter-efficient fine-tuning method using long skip connections and decoder-block-level parameter sharing. Solo Connection achieves up to 59% fewer trainable parameters than LoRA while consistently outperforming it on E2E generation benchmarks. Like LoRA, Solo Connection retains a compact set of adaptation weights, separate from the core model. These standalone modules can be independently swapped in and out, enabling multiple domain-specific adapters to run on a single GPU without duplicating the full backbone. This design significantly reduces resource overhead and enables faster, more cost-effective fine-tuning across diverse tasks.

Future Work: We aim to extend Solo Connection to broader datasets and model architectures—including computer vision—pending additional compute resources. This will help assess its scalability and impact across domains.

[1]Model variants are highlighted in blue for readability.

References

Allgower, E. and Georg, K. *Introduction to Numerical Continuation Methods*. Society for Industrial and Applied Mathematics, 2003. doi: 10.1137/1.9780898719154. URL https://epubs.siam.org/doi/abs/10.1137/1.9780898719154.

Anonymous. The equivalence of finite and infinite impulse iterative neural networks. *in preparation for submission as an arXiv.org preprint*, 2023.

Ben Zaken, E., Goldberg, Y., and Ravfogel, S. BitFit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 1–9, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-short.1. URL https://aclanthology.org/2022.acl-short.1.

Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q. V., and Salakhutdinov, R. Transformer-xl: Attentive language models beyond a fixed-length context, 2019.

Doedel, E., Champneys, A., Dercole, F., Fairgrieve, T., Kuznetsov, Y. A., Oldeman, B., Paffenroth, R., Sandstede, B., Wang, X., Zhang, C., et al. Auto-07p: Continuation and bifurcation software for ordinary differential equations. 2007.

Gauthier, D. J., Bollt, E. M., Griffith, A., and Barbosa, W. A. S. Next generation reservoir computing. *CoRR*, abs/2106.07688, 2021. URL https://arxiv.org/abs/2106.07688.

He, K., Zhang, X., Ren, S., and Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *CoRR*, abs/1502.01852, 2015. URL http://arxiv.org/abs/1502.01852.

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 (CVPR)*, June 2016.

Hershey, Q. Exploring neural network structure through iterative neural networks: Connections to dynamical systems. Master's thesis, Worcester Polytechnic Institute, 2022.

Hershey, Q., Paffenroth, R., Pathak, H., and Tavener, S. Rethinking the relationship between recurrent and non-recurrent neural networks: A study in sparsity, 2024. URL https://arxiv.org/abs/2404.00880.

Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., Attariyan, M., and Gelly, S. Parameter-efficient transfer learning for NLP. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2790–2799. PMLR, 09–15 Jun 2019a. URL https://proceedings.mlr.press/v97/houlsby19a.html.

Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., de Laroussilhe, Q., Gesmundo, A., Attariyan, M., and Gelly, S. Parameter-efficient transfer learning for NLP. *CoRR*, abs/1902.00751, 2019b. URL http://arxiv.org/abs/1902.00751.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. Lora: Low-rank adaptation of large language models, 2021.

Kopiczko, D. J., Blankevoort, T., and Asano, Y. M. VeRA: Vector-based random matrix adaptation. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=NjNfLdxr3A.

LeCun, Y., Bengio, Y., and Hinton, G. Deep learning. *nature*, 521(7553):436–444, 2015.

Nilesh Pathak, H. and Paffenroth, R. Parameter continuation methods for the optimization of deep neural networks. pp. 1637–1643, 2019. doi: 10.1109/ICMLA.2019.00268.

Novikova, J., Dušek, O., and Rieser, V. The E2E dataset: New challenges for end-to-end generation. In Jokinen, K., Stede, M., DeVault, D., and Louis, A. (eds.), *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pp. 201–206, Saarbrücken, Germany, August 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-5525. URL https://aclanthology.org/W17-5525.

Pathak, H. N. *Parameter continuation with secant approximation for deep neural networks*. PhD thesis, Master's Thesis at Worcester Polytechnic Institute, 2018.

Pathak, H. N. and Paffenroth, R. Parameter continuation methods for the optimization of deep neural networks. In *2019 18th IEEE International Conference on Machine Learning And Applications (ICMLA)*, pp. 1637–1643. IEEE, 2019.

Pathak, H. N. and Paffenroth, R. Principled curriculum learning using parameter continuation methods. 2021.

Pathak, H. N., Paffenroth, R., and Hershey, Q. Sequentia12d: Organizing center of skip connections for transformers. In *2023 International Conference on Machine Learning and Applications (ICMLA)*, pp. 362–368, 2023. doi: 10.1109/ICMLA58977.2023.00057.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. Language models are unsupervised multitask learners. 2019.

Renduchintala, A., Konuk, T., and Kuchaiev, O. Tied-lora: Enhancing parameter efficiency of lora with weight tying, 2024.

Ronneberger, O., Fischer, P., and Brox, T. U-net: Convolutional networks for biomedical image segmentation, 2015.

Rumelhart, D. E. and McClelland, J. L. *Learning Internal Representations by Error Propagation*, pp. 318–362. The MIT Press, 1987.

Sherstinsky, A. Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. *Physica D: Nonlinear Phenomena*, 404:132306, mar 2020. doi: 10.1016/j.physd.2019.132306.

Strogatz, S. H. *Nonlinear dynamics and chaos: with applications to physics, biology, chemistry, and engineering (studies in nonlinearity)*, volume 1.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., Bikel, D., Blecher, L., Ferrer, C. C., Chen, M., Cucurull, G., Esiobu, D., Fernandes, J., Fu, J., Fu, W., Fuller, B., Gao, C., Goswami, V., Goyal, N., Hartshorn, A., Hosseini, S., Hou, R., Inan, H., Kardas, M., Kerkez, V., Khabsa, M., Kloumann, I., Korenev, A., Koura, P. S., Lachaux, M.-A., Lavril, T., Lee, J., Liskovich, D., Lu, Y., Mao, Y., Martinet, X., Mihaylov, T., Mishra, P., Molybog, I., Nie, Y., Poulton, A., Reizenstein, J., Rungta, R., Saladi, K., Schelten, A., Silva, R., Smith, E. M., Subramanian, R., Tan, X. E., Tang, B., Taylor, R., Williams, A., Kuan, J. X., Xu, P., Yan, Z., Zarov, I., Zhang, Y., Fan, A., Kambadur, M., Narang, S., Rodriguez, A., Stojnic, R., Edunov, S., and Scialom, T. Llama 2: Open foundation and fine-tuned chat models, 2023. URL https://arxiv.org/abs/2307.09288.

Xu, L., Xie, H., Qin, S.-Z. J., Tao, X., and Wang, F. L. Parameter-efficient fine-tuning methods for pretrained language models: A critical review and assessment, 2023.

Zhang, Q., Chen, M., Bukharin, A., Karampatziakis, N., He, P., Cheng, Y., Chen, W., and Zhao, T. Adalora: Adaptive budget allocation for parameter-efficient fine-tuning, 2023.

A. Discussion and Related Work

Recent advances in Parameter-Efficient Fine-Tuning (PEFT) have led to various techniques that effectively adapt pre-trained language models to specific tasks while minimizing additional parameters. Adapters Houlsby et al. (2019a) are a prominent PEFT technique that has gained significant attention in recent years. The authors introduce additional learnable modules, called adapters, which are inserted between the layers of the pre-trained model. These adapters enable task-specific tuning while preserving the pre-trained knowledge. Adapters have demonstrated impressive performance gains in various NLP tasks, including language translation, sentiment analysis, and text classification. Adapters offer high flexibility and modularity, allowing easy integration with existing pre-trained models.

The core motivation behind Solo Connection is to fundamentally shift the paradigm of parameter-efficient fine-tuning (PEFT) from intra-layer adaptation to inter-layer adaptation. While Solo Connection can be loosely described using the language of adapters, doing so overlooks its theoretical foundation and architectural intent. Most adapter-based methods, including LoRA and its variants, operate within layers—modifying specific submodules like attention or feedforward blocks.

LoRA (Hu et al., 2021) is another PEFT technique that adds low-rank matrices to the decoder layer's attention and feed-forward network layers. LoRA has demonstrated impressive performance gains in various NLP tasks and is widely used Xu et al. (2023); Hu et al. (2021). Many methods have further advanced this technique to make this method more efficient and select appropriate rank (Houlsby et al., 2019b; Kopiczko et al., 2024; Zhang et al., 2023; Xu et al., 2023). While Adapter Houlsby et al. (2019a) and LoRA-based methods have impressive results, they all modify the internal workings of the decoder block. We refer to such PEFT techniques as intra-connections in GPT-2. In contrast, our proposed method, Solo Connection, operates outside the decoder layers, adapting the representation from one decoder block to another, as shown in Figure-1. The Solo Connection approach offers a unique perspective on PEFT and has the potential further to improve the efficiency and effectiveness of PEFT techniques.

Skip connections, introduced in ResNet (He et al., 2016), allow deep neural networks to preserve original input information by bypassing certain layers. While widely used in various applications, their implementation in large language models like GPT-2 presents two significant challenges:

Limited Scope in Current Models: In GPT-2, skip connections are typically confined within individual decoder blocks. However, with models ranging from 12 to 128 or more decoder blocks, there's a pressing need to reevaluate the role and potential of skip connections during fine-tuning. This paper focuses on "long skip connections" that link outputs of different decoder blocks, potentially enhancing the model's ability to adapt to new tasks while leveraging pre-trained knowledge.

Lack of Systematic Implementation: Usually, skip connections have been applied ad-hoc, with architectures primarily driven by empirical results. This approach lacks a systematic framework for organizing and optimizing skip connections, especially in complex models. Recent work like Sequential2D Pathak et al. (2023); Hershey (2022); Anonymous (2023) has begun to address this by providing insights into organizing inter- and intra-connections in feedforward and decoder networks.

Our research builds upon these insights, particularly exploring the potential of "weighted long skip connections" for efficient GPT-2 fine-tuning. We draw inspiration from techniques like U-Net Ronneberger et al. (2015), which successfully employed long skip connections between convolutional blocks to achieve superior segmentation results. By addressing these challenges, we aim to develop a more systematic and effective approach to implementing skip connections in large language models, potentially improving their adaptability and performance across various tasks.

Also, the idea of using shared blocks of training across neural networks is used in many works, for example, RNNs Rumelhart & McClelland (1987); Sherstinsky (2020), Tied-Lora Renduchintala et al. (2024) and VeRA Kopiczko et al. (2024) have used such technique to improve parameter efficiency. However, in this paper, we uniquely apply it for the decoder's representation learning and test this technique in Section-3.

Parameter continuation methods (Doedel et al., 2007; Pathak & Paffenroth, 2019; Nilesh Pathak & Paffenroth, 2019; Allgower & Georg, 2003) are a way to adapt slowly from one continuous function to another. Parameter continuation methods and related numerical analysis techniques are widely used in Dynamical Systems, Bifurcations, and Chaos theory but have limited exposure to deep learning Pathak & Paffenroth (2019). Model continuation methods Pathak & Paffenroth (2019; 2021), where a simple neural network model is trained first, and gradually model is made complex, have demonstrated their effectiveness in achieving better training and generalization performance on specific unsupervised tasks. In this paper, we mainly use the homotopy methods Allgower & Georg (2003); Doedel et al. (2007); Nilesh Pathak & Paffenroth (2019) to adapt the learning from the pre-trained phase to the fine-tuning phase gradually. While the homotopy method is easy to understand, the real challenge is where to apply it. In this paper, we discuss these details in Section-2

We extend GPT-2 to adapt to newer tasks given the pre-trained knowledge continually. GPT-2 model is a transformer-based language model Radford et al. (2019); Pathak et al. (2023) that consists of three main parts: the embedding layers, decoder blocks, and language model head. The Decoder block is the main computational block that we denote by D_i that performs multi-headed attention and projection transformations to the input token vectors. We devise and utilize Solo Connection to enhance the decoder-block (D_i) representation for new tasks in a parameter-efficient way. Our proposed method, Solo Connection, uniquely adapts a different approach by connecting the output of one decoder block to the input of another and has components inspired and devised from roots of many well-established research works such as Continuation methods, Skip Connection, and Fine-tuning of Deep learning models. Our approach offers a lightweight and efficient solution for pre-trained language model adaptation, making it an attractive alternative to LoRA and other PEFT techniques.

B. Method: Calculation of Parameters for Fine-tuning

This calculation determines the total number of parameters required for fine-tuning a pre-trained language model. Let us define some variables:

- d: the dimensionality of the decoder vector (1024 in this case)
- r: the dimensionality of the encoder vector (64 in this case)
- s: the sparsity factor (0.6 in this case, since $1-s = 0.4$)
- n: the number of encoding and decoding units (2 in this case 1 for the encoder and 1 for the decoder)
- T: the number of decoder layers.

Here, we show step-by-step parameter count calculation. First, $d \cdot r \cdot n \cdot (1 - s)$, which calculates the number of parameters required for the attention mechanism. It is the product of the decoder dimension, encoder dimension, number of attention heads, and the sparsity factor $(1-s)$. Second term, $r \cdot T$, which calculates the number of parameters required for the encoder layers. It's the product of the encoder dimension and the number of decoder layers. Final term, $d \cdot T$ calculates the number of parameters required for the decoder layers. It's the product of the decoder dimension and the number of decoder layers.

The total number of parameters is the sum of these three components: $(d \cdot r \cdot n \cdot (1 - s)) + r \cdot T + d \cdot T$. For example, with $d = 1024$, $r = 32$, $s = 0.7$, $n = 2$, $T = 11$, $1 - s = 0.3$, the total number of parameters is 31,276. So, the total number of parameters required for fine-tuning in this case it would be 31,276. Note that these calculations are specific to the paper's architecture and may vary depending on the model and fine-tuning setup.

Model	#Params	BLEU	NIST	E2E MET	Rouge	CIDEr
Baseline LoRA GPT-S	0.29M	65.79	8.49	45.21	67.46	2.27
Solo Connection (r=512)	0.8M	67.28	8.60	45.93	68.09	2.33
Solo Connection (r=128)	0.2M	66.39	8.52	44.95	68.07	2.27
Solo Connection (r=64)	0.14M	65.57	8.58	43.66	65.99	2.20
Solo Connection (r=32)	0.078M	65.30	8.53	43.35	65.30	2.17
Solo Connection (r=8)	0.02M	63.93	8.04	40.15	65.27	1.97
Solo Connection (r=512, s=0.6)	0.47M	67.64	8.65	45.70	68.28	2.33
Solo Connection (r=128, s=0.6)	0.12M	67.64	8.64	45.70	68.32	2.31
Solo Connection (r=64, s=0.6)	0.06M	67.72	8.68	45.09	67.18	2.30
Solo Connection (r=32, s=0.6)	0.03M	67.50	8.6	45.46	68.38	2.31
Solo Connection (r=8, s=0.6)	0.011M	65.84	8.46	43.06	66.40	2.18

Table 2. Comparison of performance metrics for various GPT-2 Small fine-tuned using different methods. The table displays the number of parameters (Params) for each model. Additionally, BLEU, NIST, E2E MET, Rouge, and CIDEr scores are provided for each model, facilitating comparison of the performance across different fine-tuning methods.

C. More Experiments

C.1. What is the impact of rank and sparsity?

In Table-2 and Table-3, we show comparison between the various values of the two most important hyperparameters of the Solo Connection i.e. Low-rank dimension and sparsity.

To achieve this, we ran a set of experiments with variable ranks such as 8, 32, 64, 128, and 512 and similarly for the sparsity parameter 0.6, and 0.7. For comparison purposes, we made two groups with and without sparsity to see the clear difference in performance and listed results for GPT-2 M in Table-3 and GPT-2 S in Table-2. In Table-2, we saw most Solo Connection with sparsity performed better than the Solo Connection without sparsity across all hyperparameter settings.

C.2. Impact of Number of Skip Connections and Solo Connections on Performance

We now evaluate the impact of varying both the number of Solo Connections and their span across the decoder. In the previous section, we applied a single Solo Connection to adapt one decoder block at a time. Here, we explore configurations where multiple Solo Connections are used, each spanning a longer range of decoder blocks. Specifically, we experiment with setups where three and five consecutive decoder blocks share a single Solo Connection. These experiments remain highly efficient, as they require minimal trainable parameters. Table 4 presents the performance metrics for different configurations, analyzing both the number of Solo Connections and their span across the decoder. Our results indicate that increasing the Solo Connection span to three blocks maintains performance close to our best results in Table-1. However, when the span is extended to five blocks, we observe a significant performance drop, especially when the overall number of Solo Connections is reduced. These trends remain consistent across different Solo Encoder dimension settings (i.e., 64 and 128).

C.3. Should the matrices, Encoder and Decoder be trained?

Next, after doing an extensive literature survey, we found that the efficiency of LoRA can be further improved using methods such as sparsity, adaptive rank, and parameter sharing Hu et al. (2021); Zhang et al. (2023); Kopiczko et al. (2024). We test the hypothesis of whether the individual components encoder and decoder should be trained or the random and non-trainable transformations can yield similar results as observed in recent literature Gauthier et al. (2021); Kopiczko et al. (2024). In Table-5, we show results with two ranks (r=512 and r=1024), and with both, the generalization results on all the metrics were poor for the case of random and non-trainable functions.

C.4. Affect of Homotopy Linear Layer

In this section, we examine the importance of the homotopy parameter when fine-tuning GPT-2 Small with Solo Connection. We conduct two experiments - Case-1: Solo Connection includes a homotopy layer, as defined in Equation 3. Case-2: The homotopy layer is replaced with a simple trainable vector $g(\mathbf{z}) = \mathbf{v} \odot \mathbf{z}$ where $\mathbf{z}$ is the input vector and $\mathbf{v}$ is the output.

Table 6 highlights the critical role of the homotopy layer in training Solo Connection. Specifically, we find that removing the homotopy parameter λ leads to a training collapse from which the model does not recover, even after multiple epochs. A closer investigation reveals that, in Case-1, the final value of λ post-fine-tuning typically falls between 0 and 0.1, thereby normalizing the contribution from the Solo Connection. In contrast, in Case-2, the random initialization of the trainable vector hinders convergence relative to Case-

Model	#Params	BLEU	NIST	E2E MET	Rouge	CIDEr
GPT-2 M (FT)*	354.92M	68.2	8.62	46.2	71.0	2.47
(LoRA) Baseline	0.35M	67.45	8.58	45.93	68.8	2.36
Solo Connection (r=512)	1.06M	68.10	8.65	45.32	68.13	2.33
Solo Connection (r=128)	0.27M	67.89	8.66	45.29	68.56	2.34
Solo Connection (r=64)	0.14M	65.30	8.58	42.96	63.52	2.18
Solo Connection (r=32)	0.077M	64.93	8.54	43.15	63.27	2.17
Solo Connection (r=8)	0.027M	64.93	8.54	43.15	63.27	2.17
Solo Connection (r=512, s=0.7)	0.26M	67.7	8.64	45.95	69.13	2.36
Solo Connection (r=128, s=0.7)	0.09M	67.04	8.58	45.85	68.55	2.31
Solo Connection (r=64, s=0.6)	0.06M	65.72	8.45	45.27	69.0	2.30
Solo Connection (r=32, s=0.6)	0.037M	66.36	8.5	44.71	67.6	2.32
Solo Connection (r=8, s=0.7)	0.014M	63.06	8.22	42.56	64.6	2.11

Table 3. Performance metrics for GPT2-M (medium) models. Comparison of performance metrics for various GPT-based models fine-tuned using different methods. The table displays the number of parameters (Params) for each model. Additionally, BLEU, NIST, E2E MET, Rouge, and CIDEr scores are provided for each model, facilitating comparison of the performance across different fine-tuning methods.

Model	#Params	BLEU	NIST	E2E MET	Rouge	CIDEr
Solo Connection (128, span=3, s=0.6)	78k	65.73	8.50	44.52	67.14	2.23
Solo Connection (64, span=3, s=0.6)	41k	66.67	8.54	44.35	66.45	2.21
Solo Connection (128, span=5, s=0.6)	76k	28.75	2.54	18.44	30.95	0.31
Solo Connection (64, span=5, s=0.6)	38k	27.28	2.90	17.93	30.06	0.33

Table 4. Performance metrics evaluating the impact of increasing the Solo Connection span to cover multiple decoder blocks. The table highlights how varying the span of Solo Connections affects model performance. Results for span=1 are in the table-2 and is the top performer followed by 3 and 5. Span=3 also shows promising results with fewer trainable parameters than span=1

Model	#Params	BLEU	NIST	E2E MET	Rouge	CIDEr
Solo Connection (r=512)	0.8M	67.28	8.60	45.93	68.09	2.33
Solo Connection (r=512)	9k	53.77	4.8069	34.89	61.25	1.38
Solo Connection (r=1024)	10k	61.56	7.1842	38.57	64.56	1.7116

Table 5. Performance metrics where Solo connection's encoder and decoder are random and not trainable.

1. In future work, we plan to conduct additional experiments to further explore this configuration.

Limitations

We acknowledge that recent PEFT variants offer promising advancements, such as SVF, SVFT, MiLoRA, PiSSA, LoRA-XS, and ProLoRA. However, many of these methods have not yet reported evaluation results on the E2E benchmark, which is central to our study. Moreover, these approaches predominantly focus on intra-layer adaptations (modifying weight matrices within individual transformer blocks), whereas Solo Connection introduces trainable inter-layer skip connections. This distinct architectural choice and a novel homotopy-based adaptation mechanism enable efficient cross-layer representation sharing, setting our approach apart. We commit to incorporating broader baseline comparisons in future work once resources permit.

D. Impact Statement

This paper seeks to advance the field of Machine Learning by introducing a more efficient approach to training Large Language Models (LLMs). By reducing the computational and resource demands associated with fine-tuning, our method has the potential to make advanced language modeling accessible to a broader range of researchers, industries, and organizations. Many people can benefit from this work, for instance by incorporating sophisticated natural language generation or understanding features into applications without incurring prohibitive costs.

Our work has many potential societal implications—both beneficial and unintended. While we do not identify any particular risks that warrant specific emphasis here, we acknowledge that making LLMs easier to develop and deploy could have downstream effects on misinformation, privacy, and bias. We encourage practitioners and researchers to consider these broader impacts when applying our method

Model	#Params	BLEU	NIST	E2E MET	Rouge	CIDEr
Homotopy Layer (Case-1)	0.12M	67.64	8.64	45.70	68.32	2.31
Linear Vector (Case-2)	0.12M	0.0	0.89	0.02	0.13	0.002

Table 6. Comparing performance metrics for Solo connection (r=128, s=0.6) with trainable Homotopy Layer (Case-1) to the performance with a simple trainable vector (Case-2).

thoughtfully and to adopt responsible deployment practices to ensure equitable and ethical use of large-scale language models.