The Expressive Power of Low-Rank Adaptation

Yuchen Zeng YZENG58@WISC.EDU
Kangwook Lee KANGWOOK.LEE@WISC.EDU
University of Wisconsin-Madison

Abstract

Low-Rank Adaptation (LoRA), a parameter-efficient fine-tuning method that leverages low-rank adaptation of weight matrices, has emerged as a prevalent technique for fine-tuning pre-trained models such as large language models and diffusion models. Despite its huge success in practice, the theoretical underpinnings of LoRA have largely remained unexplored. This paper takes the first step to bridge this gap by theoretically analyzing the expressive power of LoRA. We prove that, for fully connected neural networks, LoRA can adapt any model f to accurately represent any smaller target model $\bar{f}$ if LoRA-rank $\geq$ (width of f) $\times \frac{\text{depth of } \bar{f}}{\text{depth of } f}$. We also quantify the approximation error when LoRA-rank is lower than the threshold. For Transformer networks, we show any model can be adapted to a target model of the same size with rank-$(\frac{\text{embedding size}}{2})$ LoRA adapters.

1. Introduction

Recent foundation models, such as large language models [21, 23, 31], have achieved remarkable success in a wide range of applications, owing to their significant complexity and size. This has led to the growing popularity of parameter-efficient fine-tuning approaches [2, 15, 16, 20], designed to adapt models to target tasks more efficiently. Among these, one of the most prevalent parameter-efficient fine-tuning methods is *Low-Rank Adaptation* (LoRA) [15], which employs lightweight low-rank adapters to pre-trained weight matrices. To date, LoRA has been widely used and achieved considerable success in adapting large language models [6, 12, 15] and image generation models [27] for various downstream tasks. Despite the empirical success of LoRA, little is known in theory about how it works.

Our Contributions. In this paper, we present the first set of theoretical results that characterize the expressive power of Low-Rank Adaptation (LoRA) for different model architectures: Fully Connected Neural Networks (FNN) and Transformer Networks (TFN). The core of our main theoretical findings on FNN is encapsulated in the following informal statement.

Theorem 1 (Informal) *Let $\bar{f}$ be a target FNN and f_0 be an arbitrary frozen FNN. Under mild conditions on ranks and network architectures, there exist low-rank adapters such that a low-rank adapted version of f_0 is exactly equal to $\bar{f}$.*

We present the detailed formulations of Theorem 1 in Theorem 3 and the specialized instance Corollary 4 for random models. To the best of our knowledge, this is the first known results on the expressive power of LoRA. While this informal theorem is for exact approximation, we also derive the approximation bounds as well, i.e., we characterize the approximation error between the finetuned model and the target model as a function of the LoRA-rank increases, as provided in Theorem 5.

We also summarize our main findings on TFN in the following informal theorem.

Theorem 2 (Informal) *Let $\bar{f}$ be the target TFN and f_0 be the frozen TFN. Under mild conditions on ranks and network architectures, there exist low-rank adapters for attention weight matrices such that a low-rank adapted version of f_0 is exactly equal to $\bar{f}$.*

The formal statement of Theorem 2 is provided in Theorem 6, with a specialized version in Corollary 20 tailored for random models.

Related Works. In stark contrast to the flourishing research on the expressive power of neural networks [1, 3, 4, 8, 11, 13, 14, 17–19, 24–26, 28–30, 32**?** –34], there exists a limited number of works investigating the expressive power of adaptation methods. A notable exception is Giannou et al. [10], investigating the expressive power of normalization parameter fine-tuning. They demonstrate that fine-tuning the normalization layers alone can adapt a randomly initialized ReLU network to match any target network that is $O(\text{width})$ times smaller. We borrow some proof techniques from this work, including techniques for extending results from linear neural networks to ReLU neural networks. In another recent work [9], the authors show that input processing alone (while freezing the pretrained network) can adapt any random two-layer ReLU network to achieve arbitrarily high accuracy on a Bernoulli data model over hypercube vertices. Despite these early attempts, no existing study has yet explored the expressive power of LoRA, the current leading adaptation method.

Notations Define $[N] := \{1, 2, \ldots, N\}$. We use $\boldsymbol{I}$ to represent the identity matrix. For a sequence of L matrices $(\boldsymbol{W}_l)_{l=1}^L$, we simplify the product of these matrices $\boldsymbol{W}_L \boldsymbol{W}_{L-1} \cdots \boldsymbol{W}_1$ as $\prod_{l=1}^L \boldsymbol{W}_l$. When $m > n$, we define $\sum_{i=m}^n a_i = 0$ and $\prod_{i=m}^n a_i = 1$ for scalars $(a_i)_{i=m}^n$, and $\sum_{i=m}^n \boldsymbol{W}_i = \boldsymbol{O}$ and $\prod_{i=m}^n \boldsymbol{W}_i = \boldsymbol{I}$ for square matrices $(\boldsymbol{W}_i)_{i=m}^n$.

Singular Value Decomposition (SVD) of the matrix $\boldsymbol{W}$ can be expressed as $\boldsymbol{W} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^\top$, where $\boldsymbol{U}, \boldsymbol{V} \in \mathbb{R}^{D\times D}$ are orthonormal matrices and $\boldsymbol{D} \in \mathbb{R}^{D\times D}$ is a diagonal matrix. The singular values, sorted in descending sequence, are represented on the diagonal of $\boldsymbol{D}$, denoted as $\sigma_1(\boldsymbol{W}) \geq \sigma_2(\boldsymbol{W}) \geq \cdots \geq \sigma_D(\boldsymbol{W}) \geq 0$, where $\sigma_d(\boldsymbol{W})$ denotes the d-th largest singular value for all $d \in [D]$. When $d \geq D$, $\sigma_d(\boldsymbol{W})$ is defined as zero. The best rank-r approximation (in the Frobenius norm or the 2-norm) of $\boldsymbol{W}$ is $\sum_{i=1}^r \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^T$, where $\boldsymbol{u}_i$ and $\boldsymbol{v}_i$ are the i-th column of U and V, respectively [7, 22]. We denote this best rank-r approximation by $\mathrm{LR}_r(\boldsymbol{W})$. When $r \geq \mathrm{rank}(\boldsymbol{W})$, we define $\mathrm{LR}_r(\boldsymbol{W}) = \boldsymbol{W}$.

2. Expressive Power of FNNs with LoRA

2.1. Problem Setting

We use $\mathtt{FNN}_{L,D}(\cdot; (\boldsymbol{W}_l)_{l=1}^L, (\boldsymbol{b}_l)_{l=1}^L)$ to denote a L-layer width-D fully connected ReLU neural network with weight matrices $\boldsymbol{W}_l \in \mathbb{R}^{D\times D}$ and biases $\boldsymbol{b}_l \in \mathbb{R}^D$, where $l \in [L]$. The target FNN $\bar{f}$ and frozen FNN f_0 can be represented as follows:

$$\text{Target FNN } \bar{f} := \mathtt{FNN}_{\bar{L},D}(\cdot; (\overline{\boldsymbol{W}}_l)_{l=1}^{\bar{L}}, (\bar{\boldsymbol{b}}_l)_{l=1}^{\bar{L}}), \text{ Frozen FNN } f_0 := \mathtt{FNN}_{L,D}(\cdot; (\boldsymbol{W}_l)_{l=1}^L, (\boldsymbol{b}_l)_{l=1}^L),$$

where $\overline{\boldsymbol{W}}_l \in \mathbb{R}^{D\times D}$ and $\bar{\boldsymbol{b}}_l \in \mathbb{R}^D$ represent the weight matrix and bias vector for the l-th layer of the target model $\bar{f}$, respectively. Likewise, $\boldsymbol{W}_l \in \mathbb{R}^{D\times D}$, $\boldsymbol{b}_l \in \mathbb{R}^D$ are those for f_0, for layer $l \in [L]$. Given a specified LoRA-rank $R \in [D]$, we adapt the frozen FNN f_0 into a new model f via LoRA. The adapted model f is defined as

$$\text{Adapted FNN} \quad f := \mathtt{FNN}_{L,D}(\cdot; (\boldsymbol{W}_l + \Delta\boldsymbol{W}_l)_{l=1}^L, (\widehat{\boldsymbol{b}}_l)_{l=1}^L),$$

where the weight matrix for the low-rank adapter $\Delta \boldsymbol{W}_l \in \mathbb{R}^{D\times D}$ satisfies specified rank constraints, updated bias vector $\widehat{\boldsymbol{b}}_l \in \mathbb{R}^D$ for $l \in [L]$[1].

It is common for the pretrained model to be larger than necessary. Therefore, we focus on a setting where the frozen model is deeper than the target model, i.e., $L \geq \overline{L}$. Furthermore, in this section, we let the input space $\mathcal{X} \in \mathbb{R}^{D\times D}$ be bounded.

2.2. Main Results on Fully Connected Neural Networks

Our goal here is to identify whether there exist rank-R or lower weight matrices $(\Delta \boldsymbol{W}_l)_{l=1}^L$ for the adapter and bias vectors $(\widehat{\boldsymbol{b}}_l)_{l=1}^L$ such that f can approximate $\overline{f}$ well when both frozen model f_0 and the target model $\overline{f}$ are FNNs. The key idea here involves approximating each layer of the target model using every $M = \lfloor L/\overline{L} \rfloor$ layers of the adapted model. To be more specific, we select the low-rank adapters to construct the $l_{i,1}$-th to $l_{i,2}$-th layers of f for approximating the i-th layer of target model $\overline{f}$, where

$$l_{i,1} = (i-1)M + 1, \quad l_{i,2} = \begin{cases} iM, & i \in [\overline{L}-1] \\ L, & i = \overline{L} \end{cases}, \text{ for } i \in [\overline{L}].$$

Our first theorem identifies the minimal LoRA-rank which guarantees that the frozen model f_0 can be adapted to match the target model $\overline{f}$. The assumption here is mild, as it can be met by randomly generated models (Lemma 13), with details deferred to Sec. E.2.

Theorem 3 *Under Assumption 1, there exists rank-R or lower matrices $(\Delta \boldsymbol{W}_l)_{l=1}^L$ with $\Delta \boldsymbol{W}_l \in \mathbb{R}^{D\times D}$ and bias vectors $(\widehat{\boldsymbol{b}}_l)_{l=1}^L$ with $\widehat{\boldsymbol{b}}_l \in \mathbb{R}^D$ such that when the rank of the low-rank adapter $R \geq \max_{i\in[\overline{L}]} \operatorname{rank}(\overline{\boldsymbol{W}}_i - \prod_{l\in P_i^u} \boldsymbol{W}_l)/M$, the low-rank adapted model f can exactly represent the target model $\overline{f}$, i.e., $f(\boldsymbol{x}) = \overline{f}(\boldsymbol{x})$, $\forall \boldsymbol{x} \in \mathcal{X}$.*

The following corollary shows that with LoRA, even a randomly generated model can be adapted into a target model.

Corollary 4 *Assume that the elements of matrices $(\overline{\boldsymbol{W}}_l)_{l=1}^{\overline{L}}, (\boldsymbol{W}_l)_{l=1}^L$ are independently drawn from arbitrary continuous distributions. When $R \geq D/M$, with probability 1, there exists rank-R or lower matrices $\Delta \boldsymbol{W}_1, \ldots, \Delta \boldsymbol{W}_L \in \mathbb{R}^{D\times D}$ and bias vectors $\widehat{\boldsymbol{b}}_1, \ldots, \widehat{\boldsymbol{b}}_L \in \mathbb{R}^D$ such that low-rank adapted model f can exactly represent the target model $\overline{f}$ on $\mathcal{X}$, i.e., $f(\boldsymbol{x}) = \overline{f}(\boldsymbol{x})$, $\forall \boldsymbol{x} \in \mathcal{X}$.*

This corollary suggests that with $\geq 2RD \cdot L \geq 2D^2 L/\overline{L} \approx 2D^2\overline{L}$ learnable parameters, even a random FNN can be adapted into the target model $\overline{f}$. It is noteworthy that the total number of parameters of the target model is $D^2\overline{L}$. This indicates that even though the learnable parameters under LoRA finetuning appear to be highly constrained (low-rank constrained learnable parameters distributed across many layers), the effective expressive power of LoRA is nearly optimal up to a constant factor of 2. Furthermore, Theorem 3 indicates that if the model f is 'close' to $\overline{f}$ such that $\max_{i\in[\overline{L}]} \operatorname{rank}(\overline{\boldsymbol{W}}_i - \prod_{l\in P_i^u} \boldsymbol{W}_l)$ is small, the number of learnable parameters used by LoRA can be lower than $D^2\overline{L}$.

Meanwhile, when the employed LoRA-rank is lower than the critical threshold, the following theorem provides an upper bound for the approximation error.

1. We consider the case where the bias parameters can also be updated.

Theorem 5 *Define the approximation error of i-th layer as $E_i = \sigma_{RM+1}(\overline{\boldsymbol{W}}_i - \prod_{l\in P_i^u} \boldsymbol{W}_l)$, and the magnitude of the parameters and the input as $\beta := \max_{i\in[\bar{L}]} \left(\sqrt{\|\Sigma\|_F}\prod_{j=1}^{i}\left\|\overline{\boldsymbol{W}}_j\right\|_F + \sum_{j=1}^{i}\prod_{k=j+1}^{i-1}\left\|\overline{\boldsymbol{W}}_k\right\|_F \left\|\bar{\boldsymbol{b}}_j\right\|_2\right) \bigvee \sqrt{\|\Sigma\|_F}$.*

Under Assumption 1, there exists rank-R or lower matrices $(\Delta\boldsymbol{W}_l)_{l=1}^{L}$ with $\Delta\boldsymbol{W}_l \in \mathbb{R}^{D\times D}$ and bias vectors $(\widehat{\boldsymbol{b}}_l)_{l=1}^{L}$ with $\widehat{\boldsymbol{b}}_l \in \mathbb{R}^D$ such that for input $\mathbf{x} \in \mathcal{X}$ with $\mathbb{E}\mathbf{x}\mathbf{x}^\top = \Sigma$,

$$\mathbb{E}\left\|f(\mathbf{x}) - \bar{f}(\mathbf{x})\right\|_2 \le \beta \sum_{i=1}^{\bar{L}} \max_{k\in[\bar{L}]} \left(\left\|\overline{\boldsymbol{W}}_k\right\|_F + E_k\right)^{\bar{L}-i} E_i.$$

Theorem 5 provides an upper bound on the approximation error for the adapted model. This bound is influenced by several factors: (i) magnitude of the target model's parameters and the input, which is captured by β and $\left\|\overline{\boldsymbol{W}}_k\right\|_{\mathrm{F}}$, (ii) the rank of the adapter R and the discrepancy between the frozen model and the target model $(\overline{\boldsymbol{W}}_i - \prod_{l\in P_i^u} \boldsymbol{W}_l)_{i=1}^{\bar{L}}$, both of which contribute to the term E_i, (iii) the depth of the frozen model L, reflected in M and consequently E_i.

All the proofs of the results derived here are provided in Sec. E.2. Moreover, we further optimize our results in Sec. E.3.

3. Expressive Power of Transformer Networks with LoRA

3.1. Problem Setting

A Transformer Network (TFN) comprises multiple transformer blocks and an output layer. Each block includes a self-attention layer followed by a token-wise feedforward layer. We focus on TFNs with multi-head attention layers, deferring single-head attention layer discussions to Sec. F.1. We use the same notation as Yun et al. [33], excluding skip connections for analytical feasibility.

Consider an input matrix $\boldsymbol{X} \in \mathbb{R}^{D\times N}$, where D is the dimension of the token embeddings and N is the number of tokens. The output of the l-th transformer block is denoted as $\boldsymbol{Z}_l$, which can be computed as follows:

$$\texttt{Attn}_l(\boldsymbol{Z}_{l-1}) := \sum_{h=1}^{H} \boldsymbol{W}_{Ol}^h \boldsymbol{W}_{Vl}^h \boldsymbol{Z}_{l-1} \cdot \mathrm{softmax}\left((\boldsymbol{W}_{Kl}^h \boldsymbol{Z}_{l-1})^\top \boldsymbol{W}_{Ql}^h \boldsymbol{Z}_{l-1}\right),$$
$$\boldsymbol{Z}_l := \boldsymbol{W}_{2l} \cdot \texttt{ReLU}(\boldsymbol{W}_{1l} \cdot \texttt{Attn}_l(\boldsymbol{Z}_{l-1}) + \boldsymbol{b}_{1l}\mathbf{1}_N^\top) + \boldsymbol{b}_{2l}\mathbf{1}_N^\top,$$

where we define $\boldsymbol{Z}_0 = \boldsymbol{X}$. Here, H is the number of attention heads. The weight matrices for each head $h \in [H]$ in the l-th transformer block are $\boldsymbol{W}_{Ol}^h, \boldsymbol{W}_{Vl}^h, \boldsymbol{W}_{Kl}^h, \boldsymbol{W}_{Ql}^h \in \mathbb{R}^{D\times D}$. The softmax operator $\mathrm{softmax}(\cdot)$ is applied column-wise to the matrix. Further, $\boldsymbol{W}_{2l}, \boldsymbol{W}_{1l} \in \mathbb{R}^{D\times D}$ are the weight matrices and $\boldsymbol{b}_{1l}, \boldsymbol{b}_{2l} \in \mathbb{R}^D$ are the bias vectors in the feedforward layers.

A Transformer network, denoted as $\texttt{TFN}_{L,D}$, is a composition of L Transformer blocks, followed by an softmax output layer $\mathrm{softmax}(\boldsymbol{W}_o\,\cdot)$, where $\boldsymbol{W}_o \in \mathbb{R}^{D\times D}$. The final output of the TFN is given by $\mathrm{softmax}(\boldsymbol{W}_o\boldsymbol{Z}_L)$. To study the expressive power of LoRA within TFNs featuring multi-head attention layers, we next specify the parameters of the target model $\bar{f}$, frozen model f_0, and the adapted model f, each with L transformer blocks and a dimension D for the simplicity of analysis. For ease of presentation, we drop the subscript in $\texttt{TFN}_{L,D}$, referring to it simply as $\texttt{TFN}$. Given a specified rank $R \in [D]$ for LoRA, these models are defined as follows:

$$\text{Target TFN } g = \texttt{TFN}\left(\cdot; \left(((\overline{\boldsymbol{W}}^h_{Ol}, \overline{\boldsymbol{W}}^h_{Vl}, \overline{\boldsymbol{W}}^h_{Kl}, \overline{\boldsymbol{W}}^h_{Ql})_{h=1}^H, \overline{\boldsymbol{W}}_{2l}, \overline{\boldsymbol{W}}_{1l})_{l=1}^L, \overline{\boldsymbol{W}}_o\right), (\bar{\boldsymbol{b}}_{1l}, \bar{\boldsymbol{b}}_{2l})_{l=1}^L\right),$$

$$\text{Frozen TFN } f_0 = \texttt{TFN}\left(\cdot; \left(((\boldsymbol{W}^h_{Ol}, \boldsymbol{W}^h_{Vl}, \boldsymbol{W}^h_{Kl}, \boldsymbol{W}^h_{Ql})_{h=1}^H, \boldsymbol{W}_{2l}, \boldsymbol{W}_{1l})_{l=1}^L, \boldsymbol{W}_o\right), (\boldsymbol{b}_{1l}, \boldsymbol{b}_{2l})_{l=1}^L\right),$$

$$\text{Adapted TFN } f = \texttt{TFN}\Big(\cdot; \big(((\boldsymbol{W}^h_{Ol} + \Delta\boldsymbol{W}^h_{Ol}, \boldsymbol{W}^h_{Vl} + \Delta\boldsymbol{W}^h_{Vl}, \boldsymbol{W}^h_{Kl} + \Delta\boldsymbol{W}^h_{Kl}, \boldsymbol{W}^h_{Ql} + \Delta\boldsymbol{W}^h_{Ql})_{h=1}^H,$$
$$\boldsymbol{W}_{2l} + \Delta\boldsymbol{W}_{2l}, \boldsymbol{W}_{1l} + \Delta\boldsymbol{W}_{1l})_{l=1}^L, \boldsymbol{W}_o + \Delta\boldsymbol{W}_o), (\widehat{\boldsymbol{b}}_{1l}, \widehat{\boldsymbol{b}}_{2l})_{l=1}^L\Big),$$

where all the weight matrices $\in \mathbb{R}^{D\times D}$, and the bias vectors $\in \mathbb{R}^D$. Moreover, the weight matrices of the low-rank adapters $\Delta\boldsymbol{W}^h_{Ol}, \Delta\boldsymbol{W}^h_{Vl}, \Delta\boldsymbol{W}^h_{Kl}, \Delta\boldsymbol{W}^h_{Ql}, \Delta\boldsymbol{W}_{2l}, \Delta\boldsymbol{W}_{1l}$ for all $h \in [H]$ and $l \in [L]$ are of rank R or lower.

3.2. Main Results on Transformer Networks

We now present our main findings on TFNs. The first result relies on a non-singularity assumption (Assumption 4) tailored for TFN. This assumption is mild, and models with randomly generated weights can satisfy its criteria (Lemma 19). Further details are deferred to Sec. F.2.

The following theorem shows that adding LoRA adapters primarily to the self-attention layers enables the adapted model f to exactly represent the target model $\bar{f}$. This finding aligns with Hu et al. [15], which advocates for adapting only the attention weights when applying LoRA to TFNs.

Theorem 6 *Consider a given LoRA-rank $R \in [D]$. Let Assumption 4 hold. Define the rank-based functionality gap G_i to i-th transformer block ($i \in [L]$) or output layer ($i = L+1$) as*

$$G_i = \begin{cases} \max_h \left(\operatorname{rank}(\overline{\boldsymbol{W}}^{h\top}_{Ki}\overline{\boldsymbol{W}}^h_{Qi} - \boldsymbol{W}^{h\top}_{Ki}\boldsymbol{W}^h_{Qi})\right) \vee \max_h \left(\operatorname{rank}(\overline{\boldsymbol{W}}_{1i}\overline{\boldsymbol{W}}^h_{Oi}\overline{\boldsymbol{W}}^h_{Vi} - \boldsymbol{W}_{1i}\boldsymbol{W}^h_{Oi}\boldsymbol{W}^h_{Vi})\right), & i = 1, \\ \max_h \Big(\operatorname{rank}(\overline{\boldsymbol{W}}^\top_{2,i-1}\overline{\boldsymbol{W}}^{h\top}_{Ki}\overline{\boldsymbol{W}}^h_{Qi}\overline{\boldsymbol{W}}_{2,i-1} - \boldsymbol{W}^\top_{2,i-1}\boldsymbol{W}^{h\top}_{Ki}\boldsymbol{W}^h_{Qi}\boldsymbol{W}_{2,i-1})\Big) & \\ \qquad \vee \max_h \left(\operatorname{rank}(\overline{\boldsymbol{W}}_{1i}\overline{\boldsymbol{W}}^h_{Oi}\overline{\boldsymbol{W}}^h_{Vi}\overline{\boldsymbol{W}}_{2,i-1} - \boldsymbol{W}_{1i}\boldsymbol{W}^h_{Oi}\boldsymbol{W}^h_{Vi}\boldsymbol{W}_{2,i-1})\right), & 2 \le i \le L, \\ \operatorname{rank}(\overline{\boldsymbol{W}}_o\overline{\boldsymbol{W}}_{2L} - \boldsymbol{W}_o\boldsymbol{W}_{2L}), & i = L+1. \end{cases}$$

If $R \ge \max_{i\in[L+1]}\lceil\frac{G_i}{2}\rceil$, then there exists low-rank adapters with rank lower than $R \in [D]$ $((\Delta\boldsymbol{W}^h_{Kl}, \Delta\boldsymbol{W}^h_{Ql}, \Delta\boldsymbol{W}^h_{Vl}, \Delta\boldsymbol{W}^h_{Ol})_{h=1}^H)_{l=1}^L, \Delta\boldsymbol{W}_{2L}, \Delta\boldsymbol{W}_o$ with other low-rank adapters set to $\boldsymbol{O}$, and updated bias vectors $(\widehat{\boldsymbol{b}}_{1l}, \widehat{\boldsymbol{b}}_{2l})_{l=1}^L$, such that for any $\boldsymbol{X} \in \mathbb{R}^{D\times N}$, the adapted model f exactly represents target model $\bar{f}$, i.e., $f(\boldsymbol{X}) = \bar{f}(\boldsymbol{X})$.

The complete proof of Theorem 6 and the similar results for random models can be found in Sec. F.2.

4. Discussion and Future Work

To the best of our knowledge, this paper is the first to offer a theoretical understanding of LoRA fine-tuning on both FNN and TFN. Despite these advancements our work achieves, several intriguing questions still remain open. First, for TFN, we have only identified the conditions under which the LoRA-adapted model exactly matches the target model, due to the analytical complexity of TFN. It would be interesting to quantify the approximation error when the rank is lower than required. Furthermore, for TFN, we constrain the target model and the frozen model to have identical embedding size and depth, and we omit the skip connections and layer norms for simplicity. Another intriguing direction would be to study the expressive power of LoRA under TFN cases with more general settings on TFN architectures.

References

[1] Peter L Bartlett and Shahar Mendelson. Rademacher and Gaussian complexities: Risk bounds and structural results. In *Computational Learning Theory (COLT)*, volume 2111, pages 224–240, 2001.

[2] Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. BitFit: Simple parameter-efficient fine-tuning for Transformer-based masked language-models. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1–9, 2022.

[3] Yoshua Bengio and Olivier Delalleau. On the expressive power of deep architectures. In *Algorithmic Learning Theory*, pages 18–36, 2011.

[4] Alberto Bietti and Francis Bach. Deep equals shallow for ReLU networks in kernel regimes. 2021.

[5] Richard Caron and Tim Traynor. The zero set of a polynomial. *WSMR Report*, 2005.

[6] Tuan Dinh, Yuchen Zeng, Ruisu Zhang, Ziqian Lin, Michael Gira, Shashank Rajput, Jy-yong Sohn, Dimitris Papailiopoulos, and Kangwook Lee. LIFT: Language-interfaced fine-tuning for non-language machine learning tasks. *Advances in Neural Information Processing Systems (NeurIPS)*, 35:11763–11784, 2022.

[7] Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1936.

[8] Ronen Eldan and Ohad Shamir. The power of depth for feedforward neural networks. In *Annual Conference on Learning Theory*, volume 49, pages 907–940, 2016.

[9] Matthias Englert and Ranko Lazic. Adversarial Reprogramming revisited. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pages 28588–28600, 2022.

[10] Angeliki Giannou, Shashank Rajput, and Dimitris Papailiopoulos. The expressive power of tuning only the Norm layers. *arXiv preprint arXiv:2302.07937*, 2023.

[11] Angeliki Giannou, Shashank Rajput, Jy-Yong Sohn, Kangwook Lee, Jason D. Lee, and Dimitris Papailiopoulos. Looped Transformers as programmable computers. In *International Conference on Machine Learning (ICML)*, volume 202, pages 11398–11442, 2023.

[12] Michael Gira, Ruisu Zhang, and Kangwook Lee. Debiasing pre-trained language models via efficient fine-tuning. In *Workshop on Language Technology for Equality, Diversity and Inclusion*, pages 59–69, 2022.

[13] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366, 1989.

[14] Daniel Hsu, Clayton H Sanford, Rocco Servedio, and Emmanouil Vasileios Vlatakis-Gkaragkounis. On the approximation power of two-layer networks of random ReLUs. In *Conference on Learning Theory*, volume 134, pages 2423–2461, 2021.

[15] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2022.

[16] Shengding Hu, Zhen Zhang, Ning Ding, Yadao Wang, Yasheng Wang, Zhiyuan Liu, and Maosong Sun. Sparse structure search for delta tuning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pages 9853–9865, 2022.

[17] Holden Lee, Rong Ge, Tengyu Ma, Andrej Risteski, and Sanjeev Arora. On the ability of neural nets to express distributions. In *Conference on Learning Theory*, pages 1271–1296, 2017.

[18] Shiyu Liang and R. Srikant. Why deep neural networks for function approximation? In *International Conference on Learning Representations (ICLR)*, 2017.

[19] Valerii Likhosherstov, Krzysztof Choromanski, and Adrian Weller. On the expressive power of self-attention matrices. 2021.

[20] Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin A Raffel. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pages 1950–1965, 2022.

[21] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[22] Leon Mirsky. Symmetric gauge functions and unitarily invariant norms. *The quarterly journal of mathematics*, 1960.

[23] OpenAI. GPT-4 technical report, 2023.

[24] Samet Oymak, Ankit Singh Rawat, Mahdi Soltanolkotabi, and Christos Thrampoulidis. On the role of attention in prompt-tuning. In *International Conference on Machine Learning (ICML)*, 2023.

[25] Sejun Park, Chulhee Yun, Jaeho Lee, and Jinwoo Shin. Minimum width for universal approximation. In *International Conference on Learning Representations (ICLR)*, 2021.

[26] Maithra Raghu, Ben Poole, Jon Kleinberg, Surya Ganguli, and Jascha Sohl-Dickstein. On the expressive power of deep neural networks. In *International Conference on Machine Learning (ICML)*, pages 2847–2854, 2017.

[27] Simo Ryu. Low-rank adaptation for fast text-to-image diffusion fine-tuning. https://github.com/cloneofsimo/lora, 2023.

[28] Haizhao Shen, ZuoweiYang and Shijun Zhang. Deep network approximation characterized by number of neurons. *Communications in Computational Physics*, (5):1768–1811, 2020.

[29] Matus Telgarsky. Representation benefits of deep feedforward networks. *arXiv preprint arXiv:1509.08101*, 2015.

[30] Matus Telgarsky. Benefits of depth in neural networks. In *Conference on Learning Theory*, pages 1517–1539, 2016.

[31] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. LLaMA: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[32] Vladimir N Vapnik and A Ya Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. In *Measures of Complexity: Festschrift for Alexey Chervonenkis*, pages 11–30. 2015.

[33] Chulhee Yun, Srinadh Bhojanapalli, Ankit Singh Rawat, Sashank Reddi, and Sanjiv Kumar. Are Transformers universal approximators of sequence-to-sequence functions? In *International Conference on Learning Representations (ICLR)*, 2020.

[34] Chulhee Yun, Yin-Wen Chang, Srinadh Bhojanapalli, Ankit Singh Rawat, Sashank Reddi, and Sanjiv Kumar. O(n) connections are expressive enough: Universal approximability of sparse transformers. 33:13783–13794, 2020.

Appendix

Appendix A. List of Common Notations

We first give a list of common notations that are used in the main body and appendix for reference.

- f: LoRA-adapted model.
- $\bar{f}$: target model.
- f_0: frozen/pretrained model.
- R: rank of LoRA adapters.
- D: dimensionality of the model, representing the number of neurons in each layer for FNNs and the embedding size for TFNs.
- L: depth of the (frozen) model, representing the number of layers for FNNs and the number of transformer blocks for TFNs.
- N: sequence length of the input for TFNs.
- $\boldsymbol{x}$: input.
- $\mathbf{x}$: random input.
- $\boldsymbol{X}$: matrix input.
- $\mathcal{X}$: input space.
- Σ: $\mathbb{E}\mathbf{x}\mathbf{x}^\top$.
- $\boldsymbol{W}$: a weight matrix associated with (frozen) model. Subscripts and superscripts may be added for specificity.
- $\boldsymbol{b}$: a bias vector associated with the (frozen) model. Subscripts may be added for specificity.
- $\boldsymbol{z}_l$: the output of the first l layers in the (frozen) FNN.
- $\boldsymbol{Z}_l$: the output of the first l transformer blocks in a (frozen) TFN.
- $\overline{\boldsymbol{W}}$: a weight matrix associated with the target model. Subscripts and superscripts may be added for specificity.
- $\bar{\boldsymbol{b}}$: a bias vector associated with the target model. Subscripts may be added for specificity.
- $\bar{\mathbf{z}}_l$: the intermediate output of the first l layers in target FNN given the random input $\mathbf{x}$.
- $\overline{\boldsymbol{Z}}_l$: the output of the first l transformer blocks in a target TFN.
- $\bar{L}$: depth of the target model, representing the number of layers for FNNs and the number of transformer blocks for TFNs.
- $\Delta\boldsymbol{W}$: the weight matrix of a LoRA adapter.
- $\widehat{\boldsymbol{b}}$: a bias vector associated with the LoRA-adapted model.

- $\widehat{\mathbf{z}}_l$: the output of the first l layers in the LoRA-adapted model given the random input $\mathbf{x}$.
- $\widehat{\boldsymbol{Z}}_l$: the output of the first l transformer blocks in the LoRA-adapted model.
- M: the ratio of the depth of the frozen model to that of the target model, i.e., $L/\bar{L}$.
- $\mathcal{P}$: partition $\mathcal{P} = \left\{P_1, \ldots, P_{\bar{L}}\right\}$, each element P_i specifies that the layers with index $l \in P_i$ in the adapted model will be used to approximate the i-th layer in the target model.
- P_i: the i-th element in partition $\mathcal{P}$.
- $\mathcal{P}^{\mathrm{u}}$: uniform partition $\mathcal{P}^{\mathrm{u}} := \{\{1, \ldots, M\}, \{M+1, \ldots, 2M\}, \ldots, \left\{(\bar{L}-1)M+1, \ldots, L\right\}\}$. The uniform partition indicates that every M layers in the adapted model are employed to approximate each layer in the target model.
- P_i^{u}: the i-th element in uniform partition $\mathcal{P}^{\mathrm{u}}$.
- $\boldsymbol{I}_D$: the $D \times D$ identity matrix. When the context permits, the subscript D of $\boldsymbol{I}_D$ may be omitted, simplifying the notation to $\boldsymbol{I}$.
- $\boldsymbol{I}_{a:b,D}$: a diagonal matrix where the diagonal entries from the ath to bth position are set to 1, while all remaining entries are 0s.
- $\sigma_d(\cdot)$: the d-th largest singular value for the given square matrix. When d is greater than the width of the matrix, $\sigma_d(\cdot) = 0$.
- $\mathrm{LR}_r(\cdot)$: best rank-r approximation of a square matrix in Frobenuis norm and spectral norm. The subscript r may be omitted to indicate a general low-rank approximation without specifying the rank.
- $\prod_{l \in P_i} \boldsymbol{W}_l$: product of the weight matrices from the layers $l \in P_i$, with the later layer positioned to the left and the earlier layer to the right in the matrix product. For example, $\prod_{l \in P_1^{\mathrm{u}}} \boldsymbol{W}_l = \prod_{l=1}^{M} \boldsymbol{W}_l = \boldsymbol{W}_M \cdots \boldsymbol{W}_1$.

Appendix B. Linear Algebra

In this section, we present a collection of commonly used matrix inequalities and the basic properties of randomly generated matrices.

B.1. Common Matrix Inequalities

Here, we present some commonly used basic properties for matrix multiplication including rank computation, norm inequalities, as well as key results involving the trace and Frobenius norm of

matrices for reference:

$$\begin{aligned}
&\operatorname{rank}(\boldsymbol{AB}) \leq \operatorname{rank}(\boldsymbol{A}) \wedge \operatorname{rank}(\boldsymbol{B});\\
&\|\boldsymbol{Ax}\|_2 \leq \|\boldsymbol{A}\|_2 \|\boldsymbol{x}\|_2 ;\\
&\mathbb{E}\boldsymbol{x}^\top \boldsymbol{A}\mathbf{x} = \operatorname{tr}(\boldsymbol{A}\operatorname{Cov}(\mathbf{x})) + (\mathbb{E}\mathbf{x})^\top \boldsymbol{A}(\mathbb{E}\mathbf{x}) = \operatorname{tr}(\boldsymbol{A}\mathbb{E}\mathbf{x}\mathbf{x}^\top);\\
&\operatorname{tr}(\boldsymbol{AB}) = \operatorname{tr}(\boldsymbol{BA});\\
&\operatorname{tr}(\boldsymbol{AB}) \leq \operatorname{tr}(\boldsymbol{A})\operatorname{tr}(\boldsymbol{B});\\
&\|\boldsymbol{A}\|_{\mathrm{F}} = \sqrt{\operatorname{tr}(\boldsymbol{AA}^\top)};\\
&\|\boldsymbol{A}\|_{\mathrm{F}} = \operatorname{tr}(\boldsymbol{A}) \text{ for symmetric } \boldsymbol{A};\\
&\|\boldsymbol{A}\|_{\mathrm{F}} = \sqrt{\sum_i \sigma_i^2(\boldsymbol{A})}.
\end{aligned} \tag{1}$$

B.2. Non-Singularity of Randomly Generated Matrices

Although the non-singularity of randomly generated matrices is already established, we include a proof for completeness.

To facilitate the proof, we introduce a lemma which states that if a polynomial is non-zero, then the set of roots corresponding to a zero value of the polynomial has a Lebesgue measure of zero.

Lemma 7 (Caron and Traynor [5]) *Let $p(\boldsymbol{x})$ be a polynomial of degree d, $\boldsymbol{x} \in \mathbb{R}^n$. If p is not the zero polynomial, then the set $\mathcal{S} := \{\boldsymbol{x} \in \mathbb{R}^n \mid p(\boldsymbol{x}) = 0\}$ is of Lebesgue measure zero.*

We note that the determinant of a matrix can be viewed as a polynomial function of its vectorized version. Based on this insight, we proceed with our proof.

Lemma 8 *Let $\mathbf{X} \in \mathbb{R}^{D\times D}$ be a random matrix that follows arbitrary continuous distribution with support having non-zero Lebesgue measure on $\mathbb{R}^{D\times D}$. Then, $\mathbf{X}$ is non-singular with probability 1.*

Proof [Proof of Lemma 8] The result is a direct consequence of Lemma 7. Let $\mathbf{x} = \operatorname{vec}(\mathbf{X})$. Then, $\mathbf{x}$ is a random vector following arbitrary continuous distribution with a support having non-zero Lebesgue measure on $\mathbb{R}^{D\times D}$.

First, we establish the relationship:

$$\mathbb{P}\left(\det(\mathbf{X}) = 0\right) = \mathbb{P}\left(p(\mathbf{x}) = \mathbf{0}\right)$$

for some polynomial function p. We denote the support of random vector $\mathbf{x}$ by $\mathcal{X} \subset \mathbb{R}^{D^2}$, and the probability density function (PDF) of $\mathbf{x}$ by q. Then,

$$\mathbb{P}\left(p(\mathbf{x}) = \mathbf{0}\right) = \int_{\mathcal{X}} \mathbf{1}\left\{p(\boldsymbol{x}) = \mathbf{0}\right\} q(\boldsymbol{x})\mathrm{d}\boldsymbol{x} = \int_{\mathcal{X}\cap\{\boldsymbol{x}:p(\boldsymbol{x})=\mathbf{0}\}} q(\boldsymbol{x})\mathrm{d}\boldsymbol{x}.$$

By Lemma 7, the Lebesgue measure of $\{\boldsymbol{x} : p(\boldsymbol{x}) = \mathbf{0}\}$ is zero. Hence,

$$\int_{\mathcal{X}\cap\{\boldsymbol{x}:p(\boldsymbol{x})=\mathbf{0}\}} q(\boldsymbol{x})\mathrm{d}\boldsymbol{x} = 0.$$

By combining all the equations above, we conclude that $\mathbb{P}(\det(\mathbf{X}) = 0) = 0$, which implies $\mathbf{X}$ is non-singular with probability 1. ■

Appendix C. Extended Related Works

Expressive Power of Neural Networks Theoretical study of the expressive power of unfrozen neural networks has progressed since the first universal approximation theorem [13], showing that sufficient network width and depth can guarantee function approximation [3, 8, 18]. Many recent studies obtained similar results for deep neural networks with modern twists such as ReLU activations and Transformer networks [4, 11, 14, 17, 19, 24–26, 28–30, 33**?** , 34]. Metrics like Vapnik-Chervonenkis and Rademacher complexities [1, 32] assess classification capacity. However, these theories *cannot* fully explain the performance of frozen neural networks as they generally cannot factor in pre-trained model parameters and adaptation methods.

Expressive Power of Adaptation Methods In stark contrast to the flourishing research on the expressive power of neural networks, there exists a limited number of works investigating the expressive power of adaptation methods. A notable exception is Giannou et al. [10], investigating the expressive power of normalization parameter fine-tuning. They demonstrate that fine-tuning the normalization layers alone can adapt a randomly initialized ReLU network to match any target network that is $O(\text{width})$ times smaller. We borrow some proof techniques from this work, including techniques for extending results from linear neural networks to ReLU neural networks. In another recent work [9], the authors show that input processing alone (while freezing the pretrained network) can adapt any random two-layer ReLU network to achieve arbitrarily high accuracy on a Bernoulli data model over hypercube vertices. Despite these early attempts, no existing study has yet explored the expressive power of LoRA, the current leading adaptation method.

Appendix D. Warm up: Expressive Power of Linear Models with LoRA

Before delving into the expressive power of LoRA for FNN and TFN, we begin by investigating the simplest scenario: both the target model $\overline{f}$ and the frozen model f_0 are linear, i.e.,

$$\text{Target Model} \quad \overline{f}(\boldsymbol{x}) = \overline{\boldsymbol{W}}\boldsymbol{x}, \qquad \text{Frozen Model} \quad f_0(\boldsymbol{x}) = \boldsymbol{W}_L \cdots \boldsymbol{W}_1 \boldsymbol{x} = \left(\textstyle\prod_{l=1}^{L} \boldsymbol{W}_l\right) \boldsymbol{x}.$$

This problem serves as a simplified version of approximating a target FNN, where the target model $\overline{f}$ has a single layer, the frozen model f_0 has L layers, all bias vectors in both two models are zero, and the activation functions are linear. Throughout this paper, for the sake of simplicity, we will assume that both models have the same number of neurons in each layer, i.e., $\overline{\boldsymbol{W}}, \boldsymbol{W}_1, \ldots, \boldsymbol{W}_L \in \mathbb{R}^{D\times D}$. Nevertheless, our results are readily extendable to situations where the frozen model is wider than the target model, which is a more natural setting as the frozen models are often overparameterized to ensure high capacity and good performance across diverse tasks in practice. See the discussion in Sec. I for more details.

The objective here is to incorporate low-rank adapters into the frozen model so that the adapted model can effectively approximate the target model. Unless otherwise specified, we always consider a uniform LoRA-rank for all low-rank adapters throughout this paper. For a given LoRA-rank $R \in [D]$, we apply LoRA adapters $(\Delta\boldsymbol{W}_l)_{l=1}^{L}$ to the frozen model, and the adapted model can be represented as

$$\text{Adapted Model} \quad f(\boldsymbol{x}) = (\boldsymbol{W}_L + \Delta\boldsymbol{W}_L)\cdots(\boldsymbol{W}_1 + \Delta\boldsymbol{W}_1)\boldsymbol{x},$$

where $\operatorname{rank}(\Delta\boldsymbol{W}_l) \le R$ for all $l \in [L]$.

Since the frozen model and adpated model are all linear, we can focus on quantifying the discrepancy between the linear coefficients, i.e., $\prod_{l=1}^{L}(\boldsymbol{W}_l + \Delta\boldsymbol{W}_l) - \overline{\boldsymbol{W}}$. In the subsequent lemma, we establish the minimal achievable norm, and identify the smallest LoRA-rank required for the adapted model to exactly represent the target model, i.e., $f = \overline{f}$, under a non-singularity assumption. We will demonstrate in Sec. 2.2 that this non-singularity assumption is mild, as it can be satisfied even by randomly generated weight matrices.

Lemma 9 *Define error matrix $\boldsymbol{E} := \overline{\boldsymbol{W}} - \prod_{l=1}^{L}\boldsymbol{W}_l$, and denote its rank by $R_{\boldsymbol{E}} = \operatorname{rank}(\boldsymbol{E})$. For a given LoRA-rank $R \in [D]$, assume that all the weight matrices of the frozen model $(\boldsymbol{W}_l)_{l=1}^{L}$, and $\prod_{l=1}^{L}\boldsymbol{W}_l + \mathrm{LR}_r(\boldsymbol{E})$ are non-singular for all $r \le R(L-1)$. Then, the approximation error*

$$\min_{\Delta\boldsymbol{W}_l:\operatorname{rank}(\Delta\boldsymbol{W}_l)\le R}\left\|\prod_{l=1}^{L}(\boldsymbol{W}_l+\Delta\boldsymbol{W}_l)-\overline{\boldsymbol{W}}\right\|_2 = \sigma_{RL+1}\underbrace{\left(\overline{\boldsymbol{W}}-\prod_{l=1}^{L}\boldsymbol{W}_l\right)}_{\textit{Error matrix } \boldsymbol{E}},$$

and the optimal solution to the matrix approximation problem satisfies $\prod_{l=1}^{L}(\boldsymbol{W}_l + \Delta\boldsymbol{W}_l) = \prod_{l=1}^{L}\boldsymbol{W}_l + \mathrm{LR}_{RL\wedge R_{\boldsymbol{E}}}(\boldsymbol{E})$. Therefore, when $R \ge \lceil \frac{R_{\boldsymbol{E}}}{L} \rceil$, we have $\prod_{l=1}^{L}(\boldsymbol{W}_l + \Delta\boldsymbol{W}_l) = \overline{\boldsymbol{W}}$, implying $f \equiv g$.

Proof [Proof of Lemma 9]

Our goal is to find matrices $\Delta\boldsymbol{W}_1, \ldots, \Delta\boldsymbol{W}_L$ of rank R or lower such that the product of the adapted matrices approximates the target matrix well, i.e., we aim to solve the following constrained optimization problem:

$$\min_{\Delta\boldsymbol{W}_l:\operatorname{rank}(\Delta\boldsymbol{W}_l)\le R}\left\|\prod_{l=1}^{L}(\boldsymbol{W}_l+\Delta\boldsymbol{W}_l)-\overline{\boldsymbol{W}}\right\|_2.$$

By subtracting $\prod_{l=1}^{L}\boldsymbol{W}_l$ from both terms, the constrain optimization problem becomes

$$\min_{\Delta\boldsymbol{W}_l:\operatorname{rank}(\Delta\boldsymbol{W}_l)\le R}\left\|\underbrace{\left(\prod_{l=1}^{L}(\boldsymbol{W}_l+\Delta\boldsymbol{W}_l)-\prod_{l=1}^{L}\boldsymbol{W}_l\right)}_{:=\boldsymbol{A}}-\underbrace{\left(\overline{\boldsymbol{W}}-\prod_{l=1}^{L}\boldsymbol{W}_l\right)}_{:=\boldsymbol{E}}\right\|_2. \tag{2}$$

To perform analysis on (2), we start with the analysis of $\boldsymbol{A}$ as follows:

$$\begin{aligned}\boldsymbol{A} &= \prod_{l=1}^{L}(\Delta\boldsymbol{W}_l+\boldsymbol{W}_l)-\prod_{l=1}^{L}\boldsymbol{W}_l\\ &= \Delta\boldsymbol{W}_L\prod_{l=1}^{L-1}(\Delta\boldsymbol{W}_l+\boldsymbol{W}_l)+\boldsymbol{W}_L\prod_{l=1}^{L-1}(\Delta\boldsymbol{W}_l+\boldsymbol{W}_l)-\prod_{l=1}^{L}\boldsymbol{W}_l.\end{aligned}$$

Here, we have separated the first term in the product $\prod_{l=1}^{L}(\Delta \boldsymbol{W}_l + \boldsymbol{W}_l)$, breaking it into two parts: one involving $\Delta \boldsymbol{W}_L$ and the other $\boldsymbol{W}_L$. We can further expand the part involving $\boldsymbol{W}_L$:

$$\begin{aligned}\boldsymbol{A} =& \Delta \boldsymbol{W}_L \prod_{l=1}^{L-1}(\Delta \boldsymbol{W}_l + \boldsymbol{W}_l) \\ &+ \boldsymbol{W}_L \left(\Delta \boldsymbol{W}_{L-1} \prod_{l=1}^{L-2}(\Delta \boldsymbol{W}_l + \boldsymbol{W}_l) + \boldsymbol{W}_{L-1} \prod_{l=1}^{L-2}(\Delta \boldsymbol{W}_l + \boldsymbol{W}_l) \right) - \prod_{l=1}^{L} \boldsymbol{W}_l.\end{aligned}$$

At this point, it becomes clear that this expression can be iteratively decomposed. Following this pattern, we can express $\boldsymbol{A}$ as:

$$\begin{aligned}\boldsymbol{A} =& \Delta \boldsymbol{W}_L \prod_{l=1}^{L-1}(\Delta \boldsymbol{W}_l + \boldsymbol{W}_l) + \boldsymbol{W}_L \Delta \boldsymbol{W}_{L-1} \prod_{l=1}^{L-2}(\Delta \boldsymbol{W}_l + \boldsymbol{W}_l) \\ &+ \ldots + (\prod_{l=2}^{L} \boldsymbol{W}_l)(\Delta \boldsymbol{W}_1 + \boldsymbol{W}_1) - \prod_{l=1}^{L} \boldsymbol{W}_l \\ =& \sum_{l=1}^{L} \underbrace{\left[(\prod_{i=l+1}^{L} \boldsymbol{W}_i) \Delta \boldsymbol{W}_l (\prod_{i=1}^{l-1}(\boldsymbol{W}_i + \Delta \boldsymbol{W}_i)) \right]}_{:=\boldsymbol{A}_l}.\end{aligned} \tag{3}$$

In this final form, $\boldsymbol{A}$ is decomposed as $\boldsymbol{A} = \sum_{l=1}^{L} \boldsymbol{A}_l$. It is important to note that $\operatorname{rank}(\boldsymbol{A}_l) \leq \operatorname{rank}(\Delta \boldsymbol{W}_l) \leq R$. Consequently, $\operatorname{rank}(\boldsymbol{A}) \leq \sum_{l=1}^{L} \operatorname{rank}(\boldsymbol{A}_l) \leq RL$.

Then, the optimization problem (2) can be relaxed into a low-rank approximation problem

$$(2) \geq \min_{\boldsymbol{A}:\operatorname{rank}(\boldsymbol{A}) \leq RL} \|\boldsymbol{A} - \boldsymbol{E}\|_2, \tag{4}$$

where the optimal solution is $\boldsymbol{A} = \operatorname{LR}_{RL \wedge R_{\boldsymbol{E}}}(\boldsymbol{E}) \coloneqq \boldsymbol{E}'$. Therefore, if we can identify rank-R or lower matrices $(\Delta \boldsymbol{W}_l)_{l=1}^{L}$ such that

$$\underbrace{\prod_{l=1}^{L}(\boldsymbol{W}_l + \Delta \boldsymbol{W}_l) - \prod_{l=1}^{L} \boldsymbol{W}_l}_{:=\boldsymbol{A}} = \underbrace{\operatorname{LR}_{RL \wedge R_{\boldsymbol{E}}}(\overline{\boldsymbol{W}} - \prod_{l=1}^{L} \boldsymbol{W}_l)}_{:=\boldsymbol{E}'}, \tag{5}$$

then we effectively solve the matrix approximation problem as defined in (2). Moreover, it is straightforward to verify that (5) directly implies all statements in this lemma. Therefore, our remaining proof focuses on proving (5).

Denote $R_{\boldsymbol{E}'} = RL \wedge R_{\boldsymbol{E}}$. To derive the explicit form of $\boldsymbol{E}'$, we first refer to the SVD of $\boldsymbol{E}$ as

$$\boldsymbol{E} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^{\top},$$

where $\boldsymbol{U}$ and $\boldsymbol{V}$ are orthonormal matrices and the first $R_{\boldsymbol{E}}$ diagonal entries of $\boldsymbol{D}$ are non-zero, with all remaining entries being zero. Based on this, $\boldsymbol{E}'$ is expressed as

$$\boldsymbol{E}' = \boldsymbol{U}\boldsymbol{D}\boldsymbol{I}_{1:RL,D}\boldsymbol{V}^{\top}.$$

Having already derived the decomposition $\boldsymbol{A} = \sum_{l=1}^{L} \boldsymbol{A}_l$, we next aim to decompose $\boldsymbol{E}'$ as $\boldsymbol{E}' = \sum_{l=1}^{L} \boldsymbol{E}'\boldsymbol{Q}_l$, where $\boldsymbol{Q}_1, \ldots, \boldsymbol{Q}_L \in \mathbb{R}^{D\times D}$. The goal now shifts to identifying $\Delta\boldsymbol{W}_l, \boldsymbol{Q}_l$ such that $\boldsymbol{A}_l = \boldsymbol{E}'\boldsymbol{Q}_l$ for each $l \in [L]$. Achieving this would complete the proof of (5).

Therefore, our goal becomes finding $\Delta\boldsymbol{W}_1, \ldots, \Delta\boldsymbol{W}_L$ with $\mathrm{rank}(\Delta\boldsymbol{W}_l) \le R$ for all $l \in [L]$ such that

$$\boldsymbol{A}_l = (\prod_{i=l+1}^{L} \boldsymbol{W}_i)\Delta\boldsymbol{W}_l(\prod_{i=1}^{l-1}(\boldsymbol{W}_i + \Delta\boldsymbol{W}_i)) = \boldsymbol{E}'\boldsymbol{Q}_l, \quad \text{for all } l \in [L]. \tag{6}$$

One sufficient condition for achieving (6) is that the decomposed matrices $\boldsymbol{Q}_1, \boldsymbol{Q}_L$ and low-rank adapters $\Delta\boldsymbol{W}_1, \ldots, \Delta\boldsymbol{W}_L$ meet the following conditions:

$$\sum_{l=1}^{L} \boldsymbol{E}'\boldsymbol{Q}_l = \boldsymbol{E}', \tag{7}$$

$$\Delta\boldsymbol{W}_l = (\prod_{i=l+1}^{L} \boldsymbol{W}_i)^{-1}\boldsymbol{E}'\boldsymbol{Q}_l(\prod_{i=1}^{l-1}(\boldsymbol{W}_i + \Delta\boldsymbol{W}_i))^{-1}, \text{ for all } l \in [L] \tag{8}$$

$$\mathrm{rank}(\Delta\boldsymbol{W}_l) \le R, \text{ for all } l \in [L], \tag{9}$$

$$\mathrm{rank}(\boldsymbol{W}_l + \Delta\boldsymbol{W}_l) = D, \text{ for all } l \in [L-1]. \tag{10}$$

Here (7) describes the decomposition of $\boldsymbol{E}'$, (8) provides one simple solution to (6) when (10) holds, and (9) is the rank constraint on the low-rank adapter. In particular, the (10) is used to ensure the invertibility of $\prod_{i=1}^{l}(\boldsymbol{W}_i + \Delta\boldsymbol{W}_i)$ for $l \in [L-1]$. This condition is not necessary for $l = L$ as the inverse of $\boldsymbol{W}_L + \Delta\boldsymbol{W}_L$ is not required for computing any low-rank adapters.

We will show that the matrices $(\boldsymbol{Q}_l)_{l=1}^{L}$ defined by

$$\boldsymbol{Q}_l = \boldsymbol{V}\boldsymbol{I}_{(R(l-1)+1)\wedge R_{\boldsymbol{E}'}:Rl\wedge R_{\boldsymbol{E}'},D}\boldsymbol{V}^\top, \quad \text{for all } l \in [L], \tag{11}$$

and $\Delta\boldsymbol{W}_l$ defined by (8) for all $l \in [L]$ satisfies the all four conditions (7), (8), (9), and (10). We note that the definition of $(\boldsymbol{Q}_l)_{l=1}^{L}$ clearly satisfies condition (7). For the remaining conditions, namely (8), (9), (10), we proceed the proof by induction.

When $l = 1$. We begin by examining the three conditions (8), (9) and (10) under the base case $l = 1$. We first determine $\boldsymbol{Q}_1$ and $\Delta\boldsymbol{W}_1$ based on (11) and (8):

$$\Delta\boldsymbol{W}_1 = (\prod_{i=2}^{L} \boldsymbol{W}_i)^{-1}\boldsymbol{E}'\boldsymbol{Q}_1, \ \boldsymbol{Q}_1 = \boldsymbol{I}_{1:R,D}. \tag{12}$$

By the choice of $\Delta\boldsymbol{W}_1$, we satisfy the condition (8). Moreover, it directly follows that $\mathrm{rank}(\Delta\boldsymbol{W}_1) \le \mathrm{rank}(\boldsymbol{Q}_1) = R$, thereby fulfilling the rank constraint in (9).

Therefore, we just need to prove that $\boldsymbol{W}_1 + \Delta\boldsymbol{W}_1$ is full-rank, as required by condition (10). To compute $\mathrm{rank}(\boldsymbol{W}_1 + \Delta\boldsymbol{W}_1)$, we proceed as follows:

$$
\begin{aligned}
&\mathrm{rank}(\boldsymbol{W}_1 + \Delta\boldsymbol{W}_1) \\
&\overset{(12)}{=} \mathrm{rank}(\boldsymbol{W}_1 + (\prod_{i=2}^{L} \boldsymbol{W}_i)^{-1}\boldsymbol{E}'\boldsymbol{Q}_1) && \text{(Substituting for } \Delta\boldsymbol{W}_1\text{)} \\
&= \mathrm{rank}((\prod_{i=1}^{L} \boldsymbol{W}_i) + \boldsymbol{E}'\boldsymbol{Q}_1) && \text{(Left multiplying with invertible } (\prod_{i=2}^{L} \boldsymbol{W}_i)^{-1}\text{)} \\
&= \mathrm{rank}((\prod_{i=1}^{L} \boldsymbol{W}_i) + \mathrm{LR}_{R\wedge R_{\boldsymbol{E}'}}(\boldsymbol{E})). && \text{(Simplifying)}
\end{aligned}
$$

Given the assumption that $\prod_{l=1}^{L} \boldsymbol{W}_l + \mathrm{LR}_r(\boldsymbol{E})$ is full rank for all $r \leq R(L-1)$, $\mathrm{rank}(\boldsymbol{W}_1 + \Delta\boldsymbol{W}_1) = \mathrm{rank}((\prod_{i=1}^{L} \boldsymbol{W}_i) + \mathrm{LR}_{R\wedge R_{\boldsymbol{E}'}}(\boldsymbol{E})) = D$, satisfying the last condition (10).

When $l > 1$. Consider $l = 2, \ldots, L$. We assume that for $i \in [l-1]$, we have determined matrices $\boldsymbol{Q}_i$ and $\Delta\boldsymbol{W}_i$ based on (11) and (8), respectively, and we assume that they satisfy the conditions (8), (9), and (10).

First, under the induction assumption that $\boldsymbol{W}_i + \Delta\boldsymbol{W}_i$ is invertible for all $i \in [l-1]$, to achieve $\boldsymbol{A}_l = \boldsymbol{E}'\boldsymbol{Q}_l$, we set $\Delta\boldsymbol{W}_l$ based on (8). This definition ensures $\mathrm{rank}(\Delta\boldsymbol{W}_l) \leq \mathrm{rank}(\boldsymbol{Q}_l) = R$, thereby satisfying the condition (9). To prove that $\boldsymbol{W}_l + \Delta\boldsymbol{W}_l$ is full-rank (condition (10)), we focus on computing $\mathrm{rank}(\boldsymbol{W}_l + \Delta\boldsymbol{W}_l)$. We proceed as follows:

$$
\begin{aligned}
&\mathrm{rank}(\boldsymbol{W}_l + \Delta\boldsymbol{W}_l) \\
&\overset{(8)}{=} \mathrm{rank}(\boldsymbol{W}_l + (\prod_{i=l+1}^{L} \boldsymbol{W}_i)^{-1}\boldsymbol{E}'\boldsymbol{Q}_l(\prod_{i=1}^{l-1}(\boldsymbol{W}_i + \Delta\boldsymbol{W}_i)^{-1})) && \text{(Substituting for } \Delta\boldsymbol{W}_l\text{)} \\
&= \mathrm{rank}(\boldsymbol{I}_D + (\prod_{i=l}^{L} \boldsymbol{W}_i)^{-1}\boldsymbol{E}'\boldsymbol{Q}_l(\prod_{i=1}^{l-1}(\boldsymbol{W}_i + \Delta\boldsymbol{W}_i))^{-1}) && \text{(Left multiplying invertible } \boldsymbol{W}_l^{-1}\text{)} \\
&= \mathrm{rank}\Big(\prod_{i=1}^{l-1}(\boldsymbol{W}_i + \Delta\boldsymbol{W}_i) + (\prod_{i=l}^{L} \boldsymbol{W}_i)^{-1}\boldsymbol{E}'\boldsymbol{Q}_l\Big) && \text{(Right multiplying invertible } \prod_{i=1}^{l-1}(\boldsymbol{W}_i + \Delta\boldsymbol{W}_i)\text{)} \\
&= \mathrm{rank}\Big((\boldsymbol{W}_{l-1} + \Delta\boldsymbol{W}_{l-1})\prod_{i=1}^{l-2}(\boldsymbol{W}_i + \Delta\boldsymbol{W}_i) + (\prod_{i=l}^{L} \boldsymbol{W}_i)^{-1}\boldsymbol{E}'\boldsymbol{Q}_l\Big) && \text{(Rearranging terms)} \\
&\overset{(8)}{=} \mathrm{rank}\Big((\boldsymbol{W}_{l-1} + (\prod_{i=l}^{L} \boldsymbol{W}_i)^{-1}\boldsymbol{E}'\boldsymbol{Q}_{l-1}(\prod_{i=1}^{l-2}(\boldsymbol{W}_i + \Delta\boldsymbol{W}_i))^{-1})\prod_{i=1}^{l-2}(\boldsymbol{W}_i + \Delta\boldsymbol{W}_i) \\
&\qquad + (\prod_{i=l}^{L} \boldsymbol{W}_i)^{-1}\boldsymbol{E}'\boldsymbol{Q}_l\Big) && \text{(Substituting for } \Delta\boldsymbol{W}_{l-1}\text{)} \\
&= \mathrm{rank}\Big((\prod_{i=l-1}^{L} \boldsymbol{W}_i + \boldsymbol{E}'\boldsymbol{Q}_{l-1}(\prod_{i=1}^{l-2}(\boldsymbol{W}_i + \Delta\boldsymbol{W}_i))^{-1})\prod_{i=1}^{l-2}(\boldsymbol{W}_i + \Delta\boldsymbol{W}_i)
\end{aligned}
$$

$$+ \boldsymbol{E}'\boldsymbol{Q}_l\Big) \qquad \text{(Left multiplying } \prod_{i=l}^{L} \boldsymbol{W}_i\text{)}$$

$$= \operatorname{rank}\left((\prod_{i=l-1}^{L} \boldsymbol{W}_i \prod_{i=1}^{l-2} (\boldsymbol{W}_i + \Delta\boldsymbol{W}_i) + \boldsymbol{E}'\boldsymbol{Q}_{l-1} + \boldsymbol{E}'\boldsymbol{Q}_l \right) \qquad \text{(Rearranging terms)}$$

$$= \cdots$$

$$= \operatorname{rank}(\prod_{i=1}^{L} \boldsymbol{W}_i + \boldsymbol{E}'(\sum_{i=1}^{l} \boldsymbol{Q}_i)) \qquad \text{(Taking similar steps)}$$

$$= \operatorname{rank}(\prod_{i=1}^{L} \boldsymbol{W}_i + \operatorname{LR}_{Rl \wedge R_{\boldsymbol{E}'}}(\boldsymbol{E})). \qquad \text{(Simplifying)}$$

By the assumption that $\prod_{l=1}^{L} \boldsymbol{W}_l + \operatorname{LR}_r(\boldsymbol{E})$ is full-rank for $r \leq R(L-1)$ and consequently, $\operatorname{rank}(\boldsymbol{W}_l + \Delta\boldsymbol{W}_l) = \operatorname{rank}(\prod_{i=1}^{L} \boldsymbol{W}_i + \operatorname{LR}_{Rl \wedge R_{\boldsymbol{E}'}}(\boldsymbol{E})) = D$, satisfying the last condition (10).

Conclusion of Inductive Proof. Thus, by induction, we show that the definitions of $(\Delta\boldsymbol{W}_l)_{l=1}^{L}$ in (8) and $(\boldsymbol{Q}_l)_{l=1}^{L}$ in (11) ensure that $\boldsymbol{A}_l = \boldsymbol{E}'\boldsymbol{Q}_l$ for all $l \in [L]$. Summing over l from 1 to L satisfies condition (5), thereby completing the proof. ■

In fact, this lemma delivers a crucial insight. When we consider $L = 1$ and $R = D$, the lemma becomes strikingly similar to the Eckart–Young–Mirsky theorem [7, 22]. However, there is a significant difference from the classical theorem on the optimal low-rank approximation, which involves a single target matrix and a single matrix as an optimization variable. Our lemma demonstrates that a comparable result can be achieved for a "product of matrices," where each matrix is optimized subject to a low-rank constraint. That being said, even though each matrix is constrained by a low rank, the "effective rank" is the sum of these low ranks, i.e., in this scenario, is LR. Consequently, once the low-rank adapters are optimally configured, one can make the product equal to the best rank LR-approximation of the target matrix. This can be viewed as an extension of the matrix approximation theorem to a product of matrices, each subject to low-rank constraints. Our main theoretical results on the expressive power of LoRA, which we will present in the subsequent sections, will build upon this core matrix approximation result.

The following lemma extends the results to a more general setting where each low-rank adapter can be assigned a different LoRA-rank.

Lemma 10 *Define error matrix* $\boldsymbol{E} := \overline{\boldsymbol{W}} - \prod_{l=1}^{L} \boldsymbol{W}_l$*, and denote its rank by* $R_{\boldsymbol{E}} = \operatorname{rank}(\boldsymbol{E})$*. For a sequence of LoRA-ranks for all layers* $(R_l)_{l=1}^{L}$*, assume that all the weight matrices of the frozen model* $(\boldsymbol{W}_l)_{l=1}^{L}$*, and* $\prod_{l=1}^{L} \boldsymbol{W}_l + \operatorname{LR}_r(\boldsymbol{E})$ *are non-singular for all* $r \leq \sum_{l=1}^{L-1} R_l$*. Then, the approximation error*

$$\min_{\Delta\boldsymbol{W}_l : \operatorname{rank}(\Delta\boldsymbol{W}_l) \leq R_l} \left\| \prod_{l=1}^{L} (\boldsymbol{W}_l + \Delta\boldsymbol{W}_l) - \overline{\boldsymbol{W}} \right\|_2 = \sigma_{\sum_{l=1}^{L} R_l + 1} \underbrace{\left(\overline{\boldsymbol{W}} - \prod_{l=1}^{L} \boldsymbol{W}_l \right)}_{\textit{Error matrix } \boldsymbol{E}},$$

and the optimal solution to the matrix approximation problem satisfies $\prod_{l=1}^{L}(\boldsymbol{W}_l + \Delta\boldsymbol{W}_l) = \prod_{l=1}^{L}\boldsymbol{W}_l + \mathrm{LR}_{(\sum_{l=1}^{L} R_l)\wedge R_{\boldsymbol{E}}}(\boldsymbol{E})$. *Therefore, when* $\sum_{l=1}^{L} R_l \geq R_{\boldsymbol{E}}$, *we have* $\prod_{l=1}^{L}(\boldsymbol{W}_l + \Delta\boldsymbol{W}_l) = \overline{\boldsymbol{W}}$, *implying* $f \equiv g$.

Proof [Proof of Lemma 10]

The proof follows the same steps of Lemma 9 with only minor modifications.

In the current setting, we target the following constrained optimization problem:

$$\min_{\Delta\boldsymbol{W}_l:\mathrm{rank}(\Delta\boldsymbol{W}_l)\leq R_l} \left\| \prod_{l=1}^{L}(\boldsymbol{W}_l + \Delta\boldsymbol{W}_l) - \overline{\boldsymbol{W}} \right\|_2,$$

where we allow each LoRA adapter $\Delta\boldsymbol{W}_l$ can possess different LoRA-ranks R_l, i.e., $\mathrm{rank}(\Delta\boldsymbol{W}_l) \leq R_l$, $l \in [L]$. Subtracting $\prod_{l=1}^{L}\boldsymbol{W}_l$ from both terms leads us to a similar constrained optimization problem as (2). The only distinction lies in the rank constraint:

$$\min_{\Delta\boldsymbol{W}_l:\mathrm{rank}(\Delta\boldsymbol{W}_l)\leq R_l} \left\| \underbrace{\left(\prod_{l=1}^{L}(\boldsymbol{W}_l + \Delta\boldsymbol{W}_l) - \prod_{l=1}^{L}\boldsymbol{W}_l\right)}_{:=\boldsymbol{A}} - \underbrace{\left(\overline{\boldsymbol{W}} - \prod_{l=1}^{L}\boldsymbol{W}_l\right)}_{:=\boldsymbol{E}} \right\|_2. \tag{13}$$

Following the same steps, we decompose $\boldsymbol{A}$ into (3). Given that $\mathrm{rank}(\boldsymbol{A}_l) \leq \mathrm{rank}(\Delta\boldsymbol{W}_l) \leq R_l$, we deduce that $\mathrm{rank}(\boldsymbol{A}) \leq \sum_{l=1}^{L}\mathrm{rank}(\boldsymbol{A}_l) \leq \sum_{l=1}^{L} R_l$. Consequently, the optimization problem above can be eased into a low-rank approximation problem analogous to (4):

$$(13) \geq \min_{\boldsymbol{A}:\mathrm{rank}(\boldsymbol{A})\leq\sum_{l=1}^{L} R_l} \|\boldsymbol{A} - \boldsymbol{E}\|_2,$$

where the optimal solution is $\boldsymbol{A} = \mathrm{LR}_{(\sum_{l=1}^{L} R_l)\wedge R_{\boldsymbol{E}}}(\boldsymbol{E}) := \boldsymbol{E}'$. Therefore, if we can identify the LoRA adapters $(\Delta\boldsymbol{W}_l)_{l=1}^{L}$ with $\mathrm{rank}(\Delta\boldsymbol{W}_l) \leq R_l$ such that

$$\underbrace{\prod_{l=1}^{L}(\boldsymbol{W}_l + \Delta\boldsymbol{W}_l) - \prod_{l=1}^{L}\boldsymbol{W}_l}_{:=\boldsymbol{A}} = \underbrace{\mathrm{LR}_{(\sum_{l=1}^{L} R_l)\wedge R_{\boldsymbol{E}}}(\overline{\boldsymbol{W}} - \prod_{l=1}^{L}\boldsymbol{W}_l)}_{:=\boldsymbol{E}'},$$

the proof is completed.

The remaining part of the proof adheres to the steps outlined in the proof of Lemma 9 deriving (5). The only difference is that we consider a different selection of $(\boldsymbol{Q}_l)l = 1^L$ that satisfies (9) here:

$$\boldsymbol{Q}_l = \boldsymbol{V}\boldsymbol{I}_{(\sum_{i=1}^{l-1} R_i)\wedge R_{\boldsymbol{E}'}:(\sum_{i=1}^{l} R_i)\wedge R_{\boldsymbol{E}'},D}\boldsymbol{V}^\top.$$

Applying the same steps with this change yields the desired outcomes. ■

Appendix E. Expressive Power of Fully Connected Neural Networks with LoRA

In this section, we provide the full proof for deriving the main results outlined in Sec. 2, and more general results.

E.1. Approximating One-Layer ReLU FNN

We start with investigating the expressive power of LoRA on one-layer FNN. In this setting, our aim is to identify LoRA adapters $(\Delta \boldsymbol{W}_l)_{l=1}^L$ and bias vectors $(\widehat{\boldsymbol{b}}_l)_{l=1}^L$ such that the adapted model

$$\texttt{ReLU}((\boldsymbol{W}_L + \Delta \boldsymbol{W}_L) \cdot \texttt{ReLU}((\boldsymbol{W}_{L-1} + \Delta \boldsymbol{W}_{L-1}) \cdot \texttt{ReLU}(\cdots) + \widehat{\boldsymbol{b}}_{L-1}) + \widehat{\boldsymbol{b}}_L)$$

closely approximates the target one-layer ReLU FNN model $\texttt{ReLU}(\overline{\boldsymbol{W}}_1 \cdot + \bar{\boldsymbol{b}}_1)$.

This differs from the setting described in Sec. D, where a multi-layer FNN with linear activation functions and zero biases was used to approximate a one-layer FNN with the same properties. In the current setting, we introduce non-linearity through the use of ReLU activation functions in the frozen model and also take biases into account. Consequently, to generalize the findings to this new setting, addressing the introduced non-linearity due to the ReLU activation functions in the frozen model is the main challenge.

We employ the following two steps to extend the results in Sec. D to the current setting.

1. (Linearization) We eliminate the nonlinearity in the first $L-1$ layers of the adapted model, making it equivalent to a one-layer ReLU FNN. This can be readily achieved by choosing sufficiently large bias vectors for the first $L-1$ layers to ensure that all ReLUs in these layers are activated. This technique of eliminating non-linearity is inspired by Giannou et al. [10].

2. (Weight Matrix Alignment) We update the bias vectors of the last layer $\widehat{\boldsymbol{b}}_L$ to align with that of the target model $\bar{f}$, and apply the linear model approximation results (i.e., Lemma 9) to identify the low-rank adapters that match the weight matrix $\bar{f}$.

Following the steps above, we arrive at the subsequent lemma, which demonstrates that any one-layer FNN can be closely approximated by a multi-layer FNN finetuned via LoRA.

Lemma 11 *Define error matrix $\boldsymbol{E} := \overline{\boldsymbol{W}}_1 - \prod_{l=1}^L \boldsymbol{W}_l$, with its rank represented by $R_{\boldsymbol{E}} = \operatorname{rank}(\boldsymbol{E})$. Consider a LoRA-rank $R \in [D]$. Assume that the weight matrices $\boldsymbol{W}_1, \ldots, \boldsymbol{W}_L \in \mathbb{R}^{D\times D}$ and $\prod_{l=1}^L \boldsymbol{W}_l + \mathrm{LR}_r(\boldsymbol{E})$ for all $r \leq R(L-1)$ are non-singular. Let $\mathbf{x}$ be a random input sampled from a distribution with bounded support $\mathcal{X}$ and let $\Sigma = \mathbb{E}\mathbf{x}\mathbf{x}^\top$. Then, there exists rank-R or lower matrices $\Delta \boldsymbol{W}_1, \ldots, \Delta \boldsymbol{W}_L \in \mathbb{R}^{D\times D}$ and bias vectors $\widehat{\boldsymbol{b}}_1, \ldots, \widehat{\boldsymbol{b}}_L \in \mathbb{R}^D$ such that for any input $\boldsymbol{x} \in \mathcal{X}$,*

$$f(\boldsymbol{x}) - \bar{f}(\boldsymbol{x}) = \mathtt{ReLU}\left(\left(\mathrm{LR}_{RL\wedge R_{\boldsymbol{E}}}(\overline{\boldsymbol{W}}_1 - \prod_{l=1}^L \boldsymbol{W}_l) - (\overline{\boldsymbol{W}}_1 - \prod_{l=1}^L \boldsymbol{W}_l)\right)\boldsymbol{x}\right).$$

Therefore, when $R \geq R_{\boldsymbol{E}}/L$, the adapted model exactly represents the target model, i.e., $f(\boldsymbol{x}) = \bar{f}(\boldsymbol{x})$ for all $\boldsymbol{x} \in \mathcal{X}$.

Furthermore, let $\mathbf{x}$ be a random input sampled from a distribution with bounded support $\mathcal{X}$ and let $\Sigma = \mathbb{E}\mathbf{x}\mathbf{x}^\top$. Then, the expected squared error is bounded as

$$\mathbb{E}\left\|f(\mathbf{x}) - \bar{f}(\mathbf{x})\right\|_2^2 \leq \|\Sigma\|_F \sigma_{RL\wedge R_{\boldsymbol{E}}+1}^2(\overline{\boldsymbol{W}}_1 - \prod_{l=1}^L \boldsymbol{W}_l).$$

Proof [Proof of Lemma 11] This proof consists of three main steps: (i) linearize the first $L-1$ layers of the adapted model f to reduce it to a single-layer FNN, (ii) align the weight matrices and bias vectors of this simplified f with those of the target model $\bar{f}$, (iii) derive an upper bound of the error $\mathbb{E}\left\|f(\mathbf{x}) - \bar{f}(\mathbf{x})\right\|_2^2$.

Linearization. The main challenge here stems from the non-linearities introduced by the ReLU activation function. To remove the non-linearities in the first $L-1$ layers of updated model f, since the input space $\mathcal{X}$ is bounded, we can set all the entries of $\widehat{\boldsymbol{b}}_1, \ldots, \widehat{\boldsymbol{b}}_{L-1}$ sufficiently large, thereby activating all ReLUs in the first $L-1$ layers of f. Consequently, we have

$$
\begin{aligned}
f(\boldsymbol{x}) &= \texttt{ReLU}((\boldsymbol{W}_L + \Delta\boldsymbol{W}_L)\boldsymbol{z}_{L-1} + \widehat{\boldsymbol{b}}_L) \\
&= \texttt{ReLU}\left((\boldsymbol{W}_L + \Delta\boldsymbol{W}_L)\texttt{ReLU}((\boldsymbol{W}_{L-1} + \Delta\boldsymbol{W}_{L-1})\boldsymbol{z}_{L-2} + \widehat{\boldsymbol{b}}_{L-1}) + \widehat{\boldsymbol{b}}_L\right) \\
&= \texttt{ReLU}\left((\boldsymbol{W}_L + \Delta\boldsymbol{W}_L)((\boldsymbol{W}_{L-1} + \Delta\boldsymbol{W}_{L-1})\boldsymbol{z}_{L-2} + \widehat{\boldsymbol{b}}_{L-1}) + \widehat{\boldsymbol{b}}_L\right) \\
&= \texttt{ReLU}\left((\boldsymbol{W}_L + \Delta\boldsymbol{W}_L)(\boldsymbol{W}_{L-1} + \Delta\boldsymbol{W}_{L-1})\boldsymbol{z}_{L-2} + (\boldsymbol{W}_L + \Delta\boldsymbol{W}_L)\widehat{\boldsymbol{b}}_{L-1} + \widehat{\boldsymbol{b}}_L\right) \\
&= \cdots \\
&= \texttt{ReLU}\left(\prod_{l=1}^{L}(\boldsymbol{W}_l + \Delta\boldsymbol{W}_l)\boldsymbol{x} + (\sum_{l=1}^{L-1}\prod_{i=l+1}^{L}(\boldsymbol{W}_i + \Delta\boldsymbol{W}_i)\widehat{\boldsymbol{b}}_l) + \widehat{\boldsymbol{b}}_L\right),
\end{aligned}
$$

which is equivalent to a single-layer ReLU neural network with weight matrix $\prod_{l=1}^{L}(\boldsymbol{W}_l + \Delta\boldsymbol{W}_l)$ and bias vector $(\sum_{l=1}^{L-1}\prod_{i=l+1}^{L}(\boldsymbol{W}_i + \Delta\boldsymbol{W}_i)\widehat{\boldsymbol{b}}_l) + \widehat{\boldsymbol{b}}_L$.

Parameter Alignment. To match the updated model $f(\boldsymbol{x})$ and target model $\overline{f}(\boldsymbol{x})$, we proceed as follows. For weight matrix, Lemma 9 guarantees the existence of rank-R or lower matrices $\Delta\boldsymbol{W}_1, \ldots, \Delta\boldsymbol{W}_L \in \mathbb{R}^{D\times D}$ such that

$$
\prod_{l=1}^{L}(\boldsymbol{W}_l + \Delta\boldsymbol{W}_l) = \prod_{l=1}^{L}\boldsymbol{W}_l + \mathrm{LR}_{RL\wedge R_{\boldsymbol{E}}}(\overline{\boldsymbol{W}} - \prod_{l=1}^{L}\boldsymbol{W}_l). \tag{14}
$$

For the bias vector, we set $\widehat{\boldsymbol{b}}_L = \bar{\boldsymbol{b}}_1 - \sum_{l=1}^{L-1}\prod_{i=l+1}^{L}(\boldsymbol{W}_i + \Delta\boldsymbol{W}_i)\widehat{\boldsymbol{b}}_l$ such that $\sum_{l=1}^{L-1}\prod_{i=l+1}^{L}(\boldsymbol{W}_i + \Delta\boldsymbol{W}_i)\widehat{\boldsymbol{b}}_l + \widehat{\boldsymbol{b}}_L = \bar{\boldsymbol{b}}_1$. Therefore, we obtain

$$
f(\boldsymbol{x}) - \overline{f}(\boldsymbol{x}) = \texttt{ReLU}\left(\left(\mathrm{LR}_{RL\wedge R_{\boldsymbol{E}}}(\overline{\boldsymbol{W}}_1 - \prod_{l=1}^{L}\boldsymbol{W}_l) - (\overline{\boldsymbol{W}}_1 - \prod_{l=1}^{L}\boldsymbol{W}_l)\right)\boldsymbol{x}\right).
$$

Error Derivation. We compute the expected squared error as follows:

$$
\begin{aligned}
&\mathbb{E}\left\|f(\mathbf{x}) - \overline{f}(\mathbf{x})\right\|_2^2 \\
&\leq \mathbb{E}\left\|\left(\mathrm{LR}_{RL\wedge R_{\boldsymbol{E}}}(\overline{\boldsymbol{W}}_1 - \prod_{l=1}^{L}\boldsymbol{W}_l) - (\overline{\boldsymbol{W}}_1 - \prod_{l=1}^{L}\boldsymbol{W}_l)\right)\mathbf{x}\right\|_2^2 && \text{(ReLU is 1-Lipschitz)} \\
&\overset{(1)}{\leq} \left\|\mathrm{LR}_{RL\wedge R_{\boldsymbol{E}}}(\overline{\boldsymbol{W}}_1 - \prod_{l=1}^{L}\boldsymbol{W}_l) - (\overline{\boldsymbol{W}}_1 - \prod_{l=1}^{L}\boldsymbol{W}_l)\right\|_2^2 \mathbb{E}\left\|\mathbf{x}\right\|_2^2 \\
&= \left\|\Sigma\right\|_{\mathrm{F}}\sigma_{RL\wedge R_{\boldsymbol{E}}+1}^2(\overline{\boldsymbol{W}}_1 - \prod_{l=1}^{L}\boldsymbol{W}_l). && \text{(By the definition of } \mathrm{LR}_{RL\wedge R_{\boldsymbol{E}}}(\cdot))
\end{aligned}
$$

This completes the proof. ■

Lemma 11 is extended to cases where different LoRA-ranks can be used for different low-rank adapters, as detailed in the following lemma.

Lemma 12 *Define error matrix* $\boldsymbol{E} := \overline{\boldsymbol{W}}_1 - \prod_{l=1}^{L} \boldsymbol{W}_l$*, and denote its rank by* $R_{\boldsymbol{E}} = \operatorname{rank}(\boldsymbol{E})$*. Consider a sequence of LoRA-ranks* $(R_l)_{l=1}^{L}$*. Assume that the weight matrices* $\boldsymbol{W}_1, \ldots, \boldsymbol{W}_L \in \mathbb{R}^{D\times D}$ *and* $\prod_{l=1}^{L} \boldsymbol{W}_l + \mathrm{LR}_r(\boldsymbol{E})$ *for all* $r \leq \sum_{l=1}^{L-1} R_l$ *are non-singular. Then, there LoRA adapters* $(\Delta \boldsymbol{W}_l)_{l=1}^{L}$ *satisfying the rank constraints* $\operatorname{rank}(\Delta \boldsymbol{W}_l) \leq R_l$ *for all* $l \in [L]$ *and bias vectors* $\widehat{\boldsymbol{b}}_1, \ldots, \widehat{\boldsymbol{b}}_L \in \mathbb{R}^D$ *such that for any input* $\boldsymbol{x} \in \mathcal{X}$*,*

$$f(\boldsymbol{x}) - \bar{f}(\boldsymbol{x}) = \texttt{ReLU}\left(\left(\mathrm{LR}_{(\sum_{l=1}^{L} R_l)\wedge R_{\boldsymbol{E}}}(\overline{\boldsymbol{W}}_1 - \prod_{l=1}^{L} \boldsymbol{W}_l) - (\overline{\boldsymbol{W}}_1 - \prod_{l=1}^{L} \boldsymbol{W}_l)\right)\boldsymbol{x}\right).$$

Therefore, when $\sum_{l=1}^{L} R_l \geq R_{\boldsymbol{E}}$*, the adapted model exactly represents the target model, i.e.,* $f(\boldsymbol{x}) = \bar{f}(\boldsymbol{x})$ *for all* $\boldsymbol{x} \in \mathcal{X}$*.*

Furthermore, for a random input $\mathbf{x}$ *drawn from a distribution supported on* $\mathcal{X}$*, and with* $\Sigma = \mathbb{E}\mathbf{x}\mathbf{x}^\top$*, the expected squared error is bounded by:*

$$\mathbb{E}\left\|f(\mathbf{x}) - \bar{f}(\mathbf{x})\right\|_2^2 \leq \|\Sigma\|_F \, \sigma^2_{(\sum_{l=1}^{L} R_l)\wedge R_{\boldsymbol{E}}+1}(\overline{\boldsymbol{W}}_1 - \prod_{l=1}^{L} \boldsymbol{W}_l).$$

Proof [Proof of Lemma 12] This proof closely adheres to the steps detailed in the proof of Lemma 11.

The primary change implemented here is that, when we draw the analogy to (14), we apply Lemma 10 instead of Lemma 9. This results in

$$\prod_{l=1}^{L}(\boldsymbol{W}_l + \Delta \boldsymbol{W}_l) = \prod_{l=1}^{L} \boldsymbol{W}_l + \mathrm{LR}_{(\sum_{l=1}^{L} R_l)\wedge R_{\boldsymbol{E}}}(\overline{\boldsymbol{W}} - \prod_{l=1}^{L} \boldsymbol{W}_l).$$

Utilizing the steps from the proof of Lemma 11 and integrating the modification specified above, we can establish the desired result. ■

E.2. Approximating Multi-Layer ReLU FNN with Uniform Model Parition

We now generalize our discussion to the approximation of multi-layer ReLU FNNs. The key strategy for extending the results to approximating multi-layer ReLU FNNs under LoRA is model partition, inspired from Giannou et al. [10]. To elucidate this, we start with a specific example.

Example 1 *Consider the case where* $\overline{L} = 2$ *and* $L = 4$*. We view a two-layer target model* $\bar{f}$ *as a composition of two one-layer ReLU FNNs. Accordingly, we partition the four-layer adapted model* f *into two submodels, each consisting of two layers. For each layer in the target model, we utilize two corresponding layers in the frozen/adapted model for approximation. This problem then simplifies into a one-layer FNN approximation problem, which has already been addressed in Lemma 11.*

Based on this example, we introduce a ordered partition $\mathcal{P} = \left\{P_1, \ldots, P_{\overline{L}}\right\}$ to partition the layers in the adapted model f, where $\bigcup_{i=1}^{\overline{L}} P_i = [L]$. Each element $P_i \in \mathcal{P}$ consists of consecutive integers. Given a partition $\mathcal{P}$, each element P_i specifies that the layers with index $l \in P_i$ in the

adapted model will be used to approximate the i-th layer in the target model. Example 1, which uses every two layers in the adapted model to approximate each layer in the target model, can be considered as a partition represented as $\{\{1,2\},\{3,4\}\}$. Similarly, we extend this simple uniform partition into general cases for $\overline{L}$-layer target FNN and L-layer frozen FNN:

$$\mathcal{P}^{\mathrm{u}} = \left\{P_1^{\mathrm{u}}, \ldots, P_{\overline{L}}^{\mathrm{u}}\right\} := \left\{\{1,\ldots,M\},\{M+1,\ldots,2M\},\ldots,\left\{(\overline{L}-1)M+1,\ldots,L\right\}\right\},$$

where $M := \lfloor L/\overline{L} \rfloor$. The uniform partition indicates that every M layers in the adapted model are employed to approximate each layer in the target model. We use $\prod_{l\in P_i} \boldsymbol{W}_l$ to denote the product of the weight matrices from the layers $l \in P_i$, with the later layer positioned to the left and the earlier layer to the right in the matrix product. For example, $\prod_{l\in P_1^{\mathrm{u}}} \boldsymbol{W}_l = \prod_{l=1}^{M} \boldsymbol{W}_l = \boldsymbol{W}_M \cdots \boldsymbol{W}_1$.

We first extend Lemma 11 to multi-layer FNN approximation setting using this uniform partition. Note that all the discussion in Sec. 2 is based on the uniform partition, with $P_i^{\mathrm{u}} = \{l_{i,1},\ldots,l_{i,2}\}$, where $i \in [\overline{L}]$. We restate all the results here using the notation of uniform partition instead of $l_{i,1}$ and $l_{i,2}$ here.

Assumption 1 (Non-Singularity) *For a fixed LoRA-rank $R \in [D]$, the weight matrices of the frozen model $(\boldsymbol{W}_l)_{l=1}^{L}$ and matrices $\left(\prod_{l\in P_i^{u}} \boldsymbol{W}_l\right) + \mathrm{LR}_r(\overline{\boldsymbol{W}}_i - \prod_{l\in P_i^{u}} \boldsymbol{W}_l)$ are non-singular for all $r \le R(M-1)$ and $i \in [\overline{L}]$.*

Lemma 13 *Let $(\overline{\boldsymbol{W}}_l)_{l=1}^{\overline{L}}, (\boldsymbol{W}_l)_{l=1}^{L} \in \mathbb{R}^{D\times D}$ be matrices whose elements are drawn independently from arbitrary continuous distributions. Then, with probability 1, Assumption 1 holds $\forall R \in [D]$.*

Proof [Proof of Lemma 13] We first use Lemma 8 to establish that $\overline{\boldsymbol{W}}_1,\ldots,\overline{\boldsymbol{W}}_{\overline{L}},\boldsymbol{W}_1,\ldots,\boldsymbol{W}_L$ are non-singular with probability 1. The goal of the remaining proof is to demonstrate that $\left(\prod_{l\in P_i^{\mathrm{u}}} \boldsymbol{W}_l\right) + \mathrm{LR}_r(\overline{\boldsymbol{W}}_i - \prod_{l\in P_i^{\mathrm{u}}} \boldsymbol{W}_l)$ is full-rank with probability 1. In this proof, we use $p_{\cdot}$ to denote the probability density function, where the subscript indicates the associated random variable.

Fix an arbitrary $i \in [\overline{L}]$ and $r \in [R]$. Then probability of the $\left(\prod_{l\in P_i^{\mathrm{u}}} \boldsymbol{W}_l\right) + \mathrm{LR}_r\left(\overline{\boldsymbol{W}}_i - \prod_{l\in P_i^{\mathrm{u}}} \boldsymbol{W}_l\right)$ being full-rank can be computed as

$$\begin{aligned}
&\mathbb{P}\left\{\det\left(\left(\prod_{l\in P_i^{\mathrm{u}}} \boldsymbol{W}_l\right) + \mathrm{LR}_r\left(\overline{\boldsymbol{W}}_i - \prod_{l\in P_i^{\mathrm{u}}} \boldsymbol{W}_l\right)\right) \neq 0\right\} \\
&= \int_{\mathcal{E}} \mathbb{P}\left\{\det\left(\left(\prod_{l\in P_i^{\mathrm{u}}} \boldsymbol{W}_l\right) + \mathrm{LR}_r(\boldsymbol{E})\right) \neq 0 \,\middle|\, \overline{\boldsymbol{W}}_i - \prod_{l\in P_i^{\mathrm{u}}} \boldsymbol{W}_l = \boldsymbol{E}\right\} p_{\overline{\boldsymbol{W}}_i - \prod_{l\in P_i^{\mathrm{u}}} \boldsymbol{W}_l}(\boldsymbol{E})\mathrm{d}\boldsymbol{E}.
\end{aligned}$$

If the conditional random matrix $\left(\prod_{l\in P_i^{\mathrm{u}}} \mathbf{W}_l\right) + \mathrm{LR}_r(\boldsymbol{E}) \mid \overline{\boldsymbol{W}}_i - \prod_{l\in P_i^{\mathrm{u}}} \boldsymbol{W}_l = \boldsymbol{E}$ has a continuous distribution with support of non-zero Lebesgue measure on $\mathbb{R}^{D\times D}$, then

$$\mathbb{P}\left\{\det\left(\left(\prod_{l\in P_i^{\mathrm{u}}} \boldsymbol{W}_l\right) + \mathrm{LR}_r(\boldsymbol{E})\right) \neq 0 \,\middle|\, \overline{\boldsymbol{W}}_i - \prod_{l\in P_i^{\mathrm{u}}} \boldsymbol{W}_l = \boldsymbol{E}\right\} = 1$$

ensuring $\left(\prod_{l\in P_i^{\mathrm{u}}} \boldsymbol{W}_l\right) + \mathrm{LR}_r\left(\overline{\boldsymbol{W}}_i - \prod_{l\in P_i^{\mathrm{u}}} \boldsymbol{W}_l\right)$ is full-rank with probability 1.

Consequently, the remaining part of the proof aims to show that the conditional random matrix $\left(\prod_{l\in P_i^{\mathrm{u}}} \mathbf{W}_l\right) + \mathrm{LR}_r(\boldsymbol{E}) \mid \overline{\boldsymbol{W}}_i - \prod_{l\in P_i^{\mathrm{u}}} \mathbf{W}_l = \boldsymbol{E}$ follows arbitrary continuous distribution with support having non-zero Lebesgue measure on $\mathbb{R}^{D\times D}$. Denote $\mathbf{W} = \prod_{l\in P_i^{\mathrm{u}}} \boldsymbol{W}_l$. Now, consider the conditional distribution of $\prod_{l\in P_i^{\mathrm{u}}} \mathbf{W}_l \mid \overline{\boldsymbol{W}}_i - \prod_{l\in P_i^{\mathrm{u}}} \mathbf{W}_l = \boldsymbol{E}$, which can be written as

$$p_{\mathbf{W}\mid\overline{\mathbf{W}}_i-\mathbf{W}=\boldsymbol{E}}(\boldsymbol{W}) = p_{\overline{\mathbf{W}}_i}(\boldsymbol{E}+\boldsymbol{W}).$$

Since $p_{\overline{\mathbf{W}}_i}$ is continuous with support of non-zero Lebesgue measure on $\mathbb{R}^{D\times D}$, the same holds for $\prod_{l\in P_i^{\mathrm{u}}} \mathbf{W}_l \mid \overline{\boldsymbol{W}}_i - \prod_{l\in P_i^{\mathrm{u}}} \mathbf{W}_l = \boldsymbol{E}$. Furthermore, adding a constant matrix $\mathrm{LR}_r(\boldsymbol{E})$ to this conditional distribution preserves the desired properties, thus completing the proof. ■

Theorem 3 *Under Assumption 1, there exists rank-R or lower matrices $(\Delta\boldsymbol{W}_l)_{l=1}^L$ with $\Delta\boldsymbol{W}_l \in \mathbb{R}^{D\times D}$ and bias vectors $(\widehat{\boldsymbol{b}}_l)_{l=1}^L$ with $\widehat{\boldsymbol{b}}_l \in \mathbb{R}^D$ when the rank of the low-rank adapter $R \geq \max_{i\in[\overline{L}]} \mathrm{rank}(\overline{\boldsymbol{W}}_i - \prod_{l\in P_i^{\mathrm{u}}} \boldsymbol{W}_l)/M$, the low-rank adapted model f can exactly represent the target model $\overline{f}$, i.e., $f(\boldsymbol{x}) = \overline{f}(\boldsymbol{x})$ for all input $\boldsymbol{x} \in \mathcal{X}$.*

Proof [Proof of Theorem 3] The key to this proof lies in a simple idea: for each layer $i \in [\overline{L}]$ in the target model, we can update M layers (i.e., $(i-1)M+1$-th layer to iM-th layer) in the frozen model to approximate it as guaranteed by Lemma 11. Hence, all layers of the target model can be approximated by the adapted model.

Model Decomposition. We partition the adapted model f into $\overline{L}$ sub-models, each defined as

$$f_i(\cdot) = \mathtt{FNN}_{\overline{L},D}(\cdot; (\boldsymbol{W}_l + \Delta\boldsymbol{W}_l)_{l\in P_i^{\mathrm{u}}}, (\widehat{\boldsymbol{b}}_l)_{l\in P_i^{\mathrm{u}}}), \quad i \in [\overline{L}].$$

In a similar manner, we break down $\overline{f}$ into $\overline{L}$ sub-models, each is a one-layer FNN:

$$\overline{f}_i(\cdot) = \mathtt{FNN}_{1,D}(\cdot; \overline{\boldsymbol{W}}_i, \overline{\boldsymbol{b}}_i),\ i \in [\overline{L}].$$

We can then express $f(\boldsymbol{x})$ and $\overline{f}(\boldsymbol{x})$ as compositions of their respective sub-models:

$$f(\cdot) = f_{\overline{L}} \circ \cdots f_1(\cdot), \quad \overline{f}(\cdot) = \overline{f}_{\overline{L}} \circ \cdots \overline{f}_1(\cdot).$$

To analyze the error $\mathbb{E}\left\|f(\mathbf{x}) - \overline{f}(\mathbf{x})\right\|_2 = \mathbb{E}\left\|f(\mathbf{x}) - \overline{f}(\mathbf{x})\right\|_2$, we consider the error caused by each submodel. Let $\widetilde{R}_i = \mathrm{rank}(\overline{\boldsymbol{W}}_i - \prod_{l\in P_i^{\mathrm{u}}} \boldsymbol{W}_l)$ denote the rank of the discrepancy between the target weight matrix and the frozen weight matrices, where $i \in [\overline{L}]$. By Lemma 11, we can select $\Delta\boldsymbol{W}_1, \ldots, \Delta\boldsymbol{W}_L, \widehat{\boldsymbol{b}}_1, \ldots, \widehat{\boldsymbol{b}}_L$ such that

$$f_i(\boldsymbol{z}) - \overline{f}_i(\boldsymbol{z}) = \mathtt{ReLU}\left(\left(\mathrm{LR}_{RL\wedge\widetilde{R}_i}(\overline{\boldsymbol{W}}_i - \prod_{l\in P_i^{\mathrm{u}}} \boldsymbol{W}_l) - (\overline{\boldsymbol{W}}_i - \prod_{l\in P_i^{\mathrm{u}}} \boldsymbol{W}_l)\right)\boldsymbol{z}\right), \tag{15}$$

$$\mathbb{E}\left\|f_i(\mathbf{z}) - \overline{f}_i(\mathbf{z})\right\|_2^2 \leq \left\|\mathbb{E}\mathbf{z}\mathbf{z}^\top\right\|_{\mathrm{F}} \sigma^2_{RL\wedge\widetilde{R}_i+1}(\overline{\boldsymbol{W}}_i - \prod_{l=1}^{L} \boldsymbol{W}_l). \tag{16}$$

Given these selected parameters, f_i is functionally equivalent to a one-layer FNN:

$$f_i(\boldsymbol{z}) = \texttt{ReLU}\left(\left(\text{LR}_{RL\wedge\widetilde{R}_i}(\overline{\boldsymbol{W}}_i - \prod_{l\in P_i^{\text{u}}} \boldsymbol{W}_l) + \prod_{l\in P_i^{\text{u}}} \boldsymbol{W}_l\right)\boldsymbol{z}\right).$$

Clearly, when $R \geq \max_i \lceil \frac{\widetilde{R}_i}{M} \rceil$, it follows that $f_i = g_i$ for all $i \in [\bar{L}]$, which implies $f = g$. ■

Corollary 4 *Assume that the elements of matrices $(\overline{\boldsymbol{W}}_l)_{l=1}^{\bar{L}}, (\boldsymbol{W}_l)_{l=1}^{L}$ are independently drawn from arbitrary continuous distributions. When $R \geq D/M$, there exists rank-R or lower matrices $\Delta\boldsymbol{W}_1, \ldots, \Delta\boldsymbol{W}_L \in \mathbb{R}^{D\times D}$ and bias vectors $\widehat{\boldsymbol{b}}_1, \ldots, \widehat{\boldsymbol{b}}_L \in \mathbb{R}^D$ such that low-rank adapted model f can functionally cover the target model $\bar{f}$ on $\mathcal{X}$, i.e., $f(\boldsymbol{x}) = \bar{f}(\boldsymbol{x})$ for all input $\boldsymbol{x} \in \mathcal{X}$, with probability 1.*

Proof [Proof of Corollary 4] To prove the statement, we start by noting that combining Lemma 13 and Theorem 3 directly gives us $f(\boldsymbol{x}) = \bar{f}(\boldsymbol{x})$ on $\mathcal{X}$ when $R \geq \max_{i\in[\bar{L}]} \lceil \text{rank}(\overline{\boldsymbol{W}}_i - \prod_{l\in P_i^{\text{u}}} \boldsymbol{W}_l)/M \rceil$. Therefore, the only thing left is to show that $\text{rank}(\overline{\mathbf{W}}_i - \prod_{l\in P_i^{\text{u}}} \mathbf{W}_l) = D$ for $i \in [\bar{L}]$ with probability 1. In this proof, we use $p_{\cdot}$ to denote the probability density function, where the subscript indicates the associated random variable.

To establish this, consider the following probability expression:

$$\begin{aligned}
&\mathbb{P}\left\{\det\left(\overline{\mathbf{W}}_i - \prod_{l\in P_i^{\text{u}}} \mathbf{W}_l\right) \neq 0\right\} \\
&= \int \mathbb{P}\left\{\det\left(\overline{\mathbf{W}}_i - \boldsymbol{W}\right) \neq 0 \middle| \prod_{l\in P_i^{\text{u}}} \mathbf{W}_l = \boldsymbol{W}\right\} p_{\prod_{l\in P_i^{\text{u}}} \mathbf{W}_l}(\boldsymbol{W}) \mathrm{d}\boldsymbol{W}.
\end{aligned}$$

Since $\overline{\mathbf{W}}$ is independent of $\prod_{l\in P_i^{\text{u}}} \mathbf{W}_l$, we have

$$\mathbb{P}\left\{\det\left(\overline{\mathbf{W}}_i - \boldsymbol{W}\right) \neq 0 \middle| \prod_{l\in P_i^{\text{u}}} \mathbf{W}_l = \boldsymbol{W}\right\} = \mathbb{P}\left\{\det\left(\overline{\mathbf{W}}_i - \boldsymbol{W}\right) \neq 0\right\} \overset{\text{Lemma 8}}{=\!=} 1.$$

Therefore, we conclude that $\mathbb{P}\left\{\det\left(\overline{\mathbf{W}}_i - \prod_{l\in P_i^{\text{u}}} \mathbf{W}_l\right) \neq 0\right\} = 1$, which completes the proof. ■

Theorem 5 *Define the approximation error of i-th layer as $E_i = \sigma_{RM+1}(\overline{\boldsymbol{W}}_i - \prod_{l\in P_i^{u}} \boldsymbol{W}_l)$, and the magnitude of the parameters and the input as $\beta := \max_{i\in[\bar{L}]} \left(\sqrt{\|\Sigma\|_F} \prod_{j=1}^{i} \left\|\overline{\boldsymbol{W}}_j\right\|_F + \sum_{j=1}^{i} \prod_{k=j+1}^{i-1} \left\|\overline{\boldsymbol{W}}_k\right\|_F \left\|\bar{\boldsymbol{b}}_j\right\|_2\right) \bigvee \sqrt{\|\Sigma\|_F}$.*

Under Assumption 1, there exists rank-R or lower matrices $(\Delta\boldsymbol{W}_l)_{l=1}^{L}$ with $\Delta\boldsymbol{W}_l \in \mathbb{R}^{D\times D}$ and bias vectors $(\widehat{\boldsymbol{b}}_l)_{l=1}^{L}$ with $\widehat{\boldsymbol{b}}_l \in \mathbb{R}^D$ such that for input $\mathbf{x} \in \mathcal{X}$ with $\mathbb{E}\mathbf{x}\mathbf{x}^\top = \Sigma$,

$$\mathbb{E}\left\|f(\mathbf{x}) - \bar{f}(\mathbf{x})\right\|_2 \leq \beta \sum_{i=1}^{\bar{L}} \max_{k\in[\bar{L}]} \left(\left\|\overline{\boldsymbol{W}}_k\right\|_F + E_k\right)^{\bar{L}-i} E_i.$$

Proof [Proof of Theorem 5] This proof is a continuation of the proof of Theorem 3. In this proof, we will consider a more general case, without enforcing any constraints on the rank of the adapters R. We use $\widehat{\boldsymbol{W}}_i$ to denote the corresponding weight matrix, i.e., $\widehat{\boldsymbol{W}}_i = \mathrm{LR}_{RL\wedge\widetilde{R}_i}(\overline{\boldsymbol{W}}_1 - \prod_{l\in P_i^{\mathrm{u}}} \boldsymbol{W}_l) + \prod_{l\in P_i^{\mathrm{u}}} \boldsymbol{W}_l$.

Error Decomposition. For submodel $i = 2, \ldots, \overline{L}$, we calculate the expected error of the composition of the first i sub-models,

$$\begin{aligned}
&\mathbb{E}\left\|\widehat{\mathbf{z}}_i - \bar{\mathbf{z}}_i\right\|_2 = \mathbb{E}\left\|f_i(\widehat{\mathbf{z}}_{i-1}) - \bar{f}_i(\bar{\mathbf{z}}_{i-1})\right\|_2 && (17)\\
&= \mathbb{E}\left\|(f_i(\widehat{\mathbf{z}}_{i-1}) - f_i(\bar{\mathbf{z}}_{i-1})) + \left(f_i(\bar{\mathbf{z}}_{i-1}) - \bar{f}_i(\bar{\mathbf{z}}_{i-1})\right)\right\|_2 && \text{(Rearranging terms)}\\
&\le \underbrace{\mathbb{E}\left\|f_i(\widehat{\mathbf{z}}_{i-1}) - f_i(\bar{\mathbf{z}}_{i-1})\right\|_2}_{A_i} + \underbrace{\mathbb{E}\left\|f_i(\bar{\mathbf{z}}_{i-1}) - \bar{f}_i(\bar{\mathbf{z}}_{i-1})\right\|_2}_{B_i}. && \text{(Applying triangle inequality)}
\end{aligned}$$

Here A_i represents the error resulting from the discrepancy between the first $i-1$ submodels, while B_i represents the error arising from the mismatch between the i-th submodel.

Computing A_i. We start by computing the error introduced by the first $i-1$ submodels, denoted by A_i:

$$\begin{aligned}
A_i &= \mathbb{E}\left\|f_i(\widehat{\mathbf{z}}_{i-1}) - f_i(\bar{\mathbf{z}}_{i-1})\right\|_2 = \mathbb{E}\left\|\mathtt{ReLU}(\widehat{\boldsymbol{W}}_i(\widehat{\mathbf{z}}_{i-1} - \bar{\mathbf{z}}_{i-1}))\right\|_2\\
&\le \mathbb{E}\left\|\widehat{\boldsymbol{W}}_i(\widehat{\mathbf{z}}_{i-1} - \bar{\mathbf{z}}_{i-1})\right\|_2 && \text{(ReLU is 1-Lipschitz)}\\
&\overset{(1)}{\le} \left\|\widehat{\boldsymbol{W}}_i\right\|_{\mathrm{F}} \mathbb{E}\left\|\widehat{\mathbf{z}}_{i-1} - \bar{\mathbf{z}}_{i-1}\right\|_2. && (18)
\end{aligned}$$

Here,

$$\begin{aligned}
\left\|\widehat{\boldsymbol{W}}_i\right\|_{\mathrm{F}} &= \left\|\prod_{l\in P_i^{\mathrm{u}}} \boldsymbol{W}_l + \mathrm{LR}_{RM\wedge\widetilde{R}_i}(\overline{\boldsymbol{W}}_i - \prod_{l\in P_i^{\mathrm{u}}} \boldsymbol{W}_l)\right\|_{\mathrm{F}}\\
&= \left\|\overline{\boldsymbol{W}}_i + \left(\prod_{l\in P_i^{\mathrm{u}}} \boldsymbol{W}_l - \overline{\boldsymbol{W}}_i\right) + \mathrm{LR}_{RM\wedge\widetilde{R}_i}(\overline{\boldsymbol{W}}_i - \prod_{l\in P_i^{\mathrm{u}}} \boldsymbol{W}_l)\right\|_{\mathrm{F}} && \text{(Rearranging terms)}\\
&\le \left\|\overline{\boldsymbol{W}}_i\right\|_{\mathrm{F}} + \left\|\left(\prod_{l\in P_i^{\mathrm{u}}} \boldsymbol{W}_l - \overline{\boldsymbol{W}}_i\right) + \mathrm{LR}_{RM\wedge\widetilde{R}_i}(\overline{\boldsymbol{W}}_i - \prod_{l\in P_i^{\mathrm{u}}} \boldsymbol{W}_l)\right\|_{\mathrm{F}}\\
& && \text{(Applying triangle inequality)}\\
&= \left\|\overline{\boldsymbol{W}}_i\right\|_{\mathrm{F}} + \sqrt{\sum_{j=RM\wedge\widetilde{R}_i+1}^{D} \sigma_j^2(\overline{\boldsymbol{W}}_i - \prod_{l\in P_i^{\mathrm{u}}} \boldsymbol{W}_l)} && (19)\\
& && \text{(By the definition of } \overline{\boldsymbol{W}}_i \text{ and } \mathrm{LR}_{RM\wedge\widetilde{R}_i+1}(\cdot))\\
&\le \max_{k\in[\overline{L}]}(\left\|\overline{\boldsymbol{W}}_k\right\|_{\mathrm{F}} + E_i) := \alpha.
\end{aligned}$$

By combining (18) and (19), we get

$$A_i \leq \max_{k\in[\bar{L}]} \left(\left\|\overline{\boldsymbol{W}}_k\right\|_{\mathrm{F}} + E_i\right) \mathbb{E}\left\|\widehat{\mathbf{z}}_{i-1} - \bar{\mathbf{z}}_{i-1}\right\|_2 \leq \alpha \mathbb{E}\left\|\widehat{\mathbf{z}}_{i-1} - \bar{\mathbf{z}}_{i-1}\right\|_2 . \tag{20}$$

Computing B_i. We proceed to compute the error associated with the i-th submodel, which we denote as B_i. It can be evaluated as follows:

$$\begin{aligned}
B_i &= \mathbb{E}\left\|f_i(\bar{\mathbf{z}}_{i-1}) - \bar{f}_i(\bar{\mathbf{z}}_{i-1})\right\|_2 \\
&\overset{(15)}{=} \mathbb{E}\left\|\texttt{ReLU}\left(\left(\mathrm{LR}_{RM\wedge\widetilde{R}_i}(\overline{\boldsymbol{W}}_i - \prod_{l\in P_i^{\mathrm{u}}} \boldsymbol{W}_l) - (\overline{\boldsymbol{W}}_i - \prod_{l\in P_i^{\mathrm{u}}} \boldsymbol{W}_l)\right)\bar{\mathbf{z}}_{i-1}\right)\right\|_2 \\
&\leq \mathbb{E}\left\|\left(\mathrm{LR}_{RM\wedge\widetilde{R}_i}(\overline{\boldsymbol{W}}_i - \prod_{l\in P_i^{\mathrm{u}}} \boldsymbol{W}_l) - (\overline{\boldsymbol{W}}_i - \prod_{l\in P_i^{\mathrm{u}}} \boldsymbol{W}_l)\right)\bar{\mathbf{z}}_{i-1}\right\|_2 && \text{(ReLU is 1-Lipschitz)} \\
&\overset{(1)}{\leq} \left\|\mathrm{LR}_{RM\wedge\widetilde{R}_i}(\overline{\boldsymbol{W}}_i - \prod_{l\in P_i^{\mathrm{u}}} \boldsymbol{W}_l) - (\overline{\boldsymbol{W}}_i - \prod_{l\in P_i^{\mathrm{u}}} \boldsymbol{W}_l)\right\|_2 \mathbb{E}\left\|\bar{\mathbf{z}}_{i-1}\right\|_2 \\
&= \sigma_{RM\wedge\widetilde{R}_i+1}(\overline{\boldsymbol{W}}_i - \prod_{l\in P_i^{\mathrm{u}}} \boldsymbol{W}_l)\mathbb{E}\left\|\bar{\mathbf{z}}_{i-1}\right\|_2 .
\end{aligned}$$

We can further simplify $\mathbb{E}\left\|\bar{\mathbf{z}}_{i-1}\right\|_2$ as :

$$\begin{aligned}
&\mathbb{E}\left\|\bar{\mathbf{z}}_{i-1}\right\|_2 \\
&= \mathbb{E}\left\|\texttt{ReLU}(\overline{\boldsymbol{W}}_{i-1}\bar{\mathbf{z}}_{i-2} + \bar{\boldsymbol{b}}_{i-1})\right\|_2 \\
&= \mathbb{E}\left\|\overline{\boldsymbol{W}}_{i-1}\bar{\mathbf{z}}_{i-2} + \bar{\boldsymbol{b}}_{i-1}\right\|_2 && \text{(ReLU is 1-Lipschitz)} \\
&\leq \left\|\overline{\boldsymbol{W}}_{i-1}\right\|_{\mathrm{F}} \mathbb{E}\left\|\bar{\mathbf{z}}_{i-2}\right\|_2 + \left\|\bar{\boldsymbol{b}}_{i-1}\right\|_2 && \text{(Applying triangle inequality and (1))} \\
&\leq \left\|\overline{\boldsymbol{W}}_{i-1}\right\|_{\mathrm{F}} \left(\left\|\overline{\boldsymbol{W}}_{i-2}\right\|_{\mathrm{F}} \mathbb{E}\left\|\bar{\mathbf{z}}_{i-3}\right\|_2 + \left\|\bar{\boldsymbol{b}}_{i-2}\right\|_2\right) + \left\|\bar{\boldsymbol{b}}_{i-1}\right\|_2 && \text{(Following the same steps)} \\
&\leq \prod_{j=1}^{i-1}\left\|\overline{\boldsymbol{W}}_j\right\|_{\mathrm{F}} \mathbb{E}\left\|\mathbf{x}\right\|_2 + \sum_{j=1}^{i-1}\prod_{k=j+1}^{i-1}\left\|\overline{\boldsymbol{W}}_k\right\|_{\mathrm{F}}\left\|\bar{\boldsymbol{b}}_j\right\|_2 && \text{(Repeating the same steps)} \\
&= \sqrt{\|\Sigma\|_{\mathrm{F}}}\prod_{j=1}^{i-1}\left\|\overline{\boldsymbol{W}}_j\right\|_{\mathrm{F}} + \sum_{j=1}^{i-1}\prod_{k=j+1}^{i-1}\left\|\overline{\boldsymbol{W}}_k\right\|_{\mathrm{F}}\left\|\bar{\boldsymbol{b}}_j\right\|_2 \leq \beta.
\end{aligned}$$

Therefore, we obtain

$$B_i \leq \beta\sigma_{RM\wedge\widetilde{R}_i+1}(\overline{\boldsymbol{W}}_i - \prod_{l\in P_i^{\mathrm{u}}} \boldsymbol{W}_l).$$

Error Composition. Having established upper bounds for A_i and B_i, we next evaluate the expected error for the composition of the first i adapted submodels.

$$\begin{aligned}
&\mathbb{E}\left\|\widehat{\mathbf{z}}_i - \bar{\mathbf{z}}_i\right\|_2 \overset{(17)}{\leq} A_i + B_i \overset{(20)}{\leq} \alpha\mathbb{E}\left\|\widehat{\mathbf{z}}_{i-1} - \bar{\mathbf{z}}_{i-1}\right\|_2 + B_i \leq \alpha(\alpha\mathbb{E}\left\|\widehat{\mathbf{z}}_{i-2} - \bar{\mathbf{z}}_{i-2}\right\|_2 + B_{i-1}) + B_i \\
&= \alpha^2\mathbb{E}\left\|\widehat{\mathbf{z}}_{i-2} - \bar{\mathbf{z}}_{i-2}\right\|_2 + \alpha B_{i-1} + B_i \leq \cdots \leq \alpha^{i-1}\mathbb{E}\left\|\widehat{\mathbf{z}}_1 - \bar{\mathbf{z}}_1\right\|_2 + \sum_{k=2}^{i}\alpha^{i-k}B_k.
\end{aligned} \tag{21}$$

To compute the overall approximation error of f, which is the composite of all submodels, we have

$$\begin{aligned}
&\mathbb{E}\left\|f(\mathbf{x})-\overline{f}(\mathbf{x})\right\|_2=\mathbb{E}\left\|f(\mathbf{x})-\overline{f}(\mathbf{x})\right\|_2=\mathbb{E}\left\|\widehat{\mathbf{z}}_{\overline{L}}-\overline{\mathbf{z}}_{\overline{L}}\right\|_2\\
&\overset{(21)}{\leq}\alpha^{\overline{L}-1}\mathbb{E}\left\|\widehat{\mathbf{z}}_1-\overline{\mathbf{z}}_1\right\|_2+\sum_{i=2}^{\overline{L}}\alpha^{\overline{L}-i}B_i\\
&\overset{(16)}{\leq}\alpha^{\overline{L}-1}\beta\sigma_{RM\wedge\widetilde{R}_i+1}(\overline{\boldsymbol{W}}_i-\prod_{l\in P_i^{\mathrm{u}}}\boldsymbol{W}_l)+\beta\sum_{i=2}^{\overline{L}}\alpha^{\overline{L}-i}\sigma_{RM\wedge\widetilde{R}_i+1}(\overline{\boldsymbol{W}}_i-\prod_{l\in P_i^{\mathrm{u}}}\boldsymbol{W}_l)\\
&=\beta\sum_{i=1}^{\overline{L}}\alpha^{\overline{L}-i}\sigma_{RM\wedge\widetilde{R}_i+1}(\overline{\boldsymbol{W}}_i-\prod_{l\in P_i^{\mathrm{u}}}\boldsymbol{W}_l)\\
&=\beta\sum_{i=1}^{\overline{L}}\alpha^{\overline{L}-i}\sigma_{RM+1}(\overline{\boldsymbol{W}}_i-\prod_{l\in P_i^{\mathrm{u}}}\boldsymbol{W}_l).
\end{aligned}$$

Substituting α with $\max_{k\in[\overline{L}]}(\left\|\overline{\boldsymbol{W}}_k\right\|_{\mathrm{F}}+E_i)$ concludes the proof. ■

E.3. Approximating Multi-Layer ReLU FNN with General Model Parition

We note that employing this uniform partition strategy for approximating the target model may not always yield optimal results. To illustrate this, we revisit the case considered by Example 1, where $\overline{L}=2$ and $L=4$. Consider a scenario where the first layer of the frozen model has been pretrained to match the first layer of the target model. In this case, we can use just the first layer in f to approximate the first layer in $\overline{f}$, and a zero LoRA-rank is sufficient for the exact representation of the first layer. The remaining three layers in f can then be used to approximate the second layer in $\overline{f}$. Compared to uniform partition, this partition leverages more layers to approximate the second layer in $\overline{f}$, allowing us to achieve the desired performance with a lower LoRA-rank, as per Lemma 11. This suggests that our approximation error bounds could be further optimized by considering partitioning schemes tailored to specific scenarios.

We now extend our results to a more general setting, where we do not assume a uniform partition, and allow each layer in the frozen model to employ adapters with different LoRA-ranks. The rank of the LoRA adapter associated with the l-th layer in the frozen model is denoted by R_l, where $l\in[L]$. This result relies on Assumption 2, an analog of Assumption 1, but revised to include a general model partition.

Assumption 2 *For the given LoRA-rank sequence $(R_l)_{l=1}^{L}$ and partition $\mathcal{P}$, the weight matrices of the frozen model $\boldsymbol{W}_1,\ldots,\boldsymbol{W}_L$ and $\left(\prod_{l\in P_i}\boldsymbol{W}_l\right)+\mathrm{LR}_r(\overline{\boldsymbol{W}}_i-\prod_{l=\min P_i}^{\max P_i-1}\boldsymbol{W}_l)$ are non-singular for all $r\leq\sum_{l=\min P_i}^{\max P_i-1}R_l$ and $i\in[\overline{L}]$.*

Note that $\max P_i$ and $\min P_i$ here represent the maximum and minimum elements in the set P_i, respectively.

Lemma 14 *Let $(\overline{\boldsymbol{W}}_l)_{l=1}^{\overline{L}},(\boldsymbol{W}_l)_{l=1}^{L}\in\mathbb{R}^{D\times D}$ be matrices whose elements are drawn independently from arbitrary continuous distributions. Then, with probability 1, Assumption 2 holds for all $R\in[D]$.*

Proof [Proof of Lemma 14] Following the same steps in the proof of Lemma 13 but replacing the uniform partition with the general partition completes the proof. ■

Now we present our results for general partition.

Theorem 15 *Consider a partition $\mathcal{P}$ for the frozen model. Let Assumption 2 hold. If $\sum_{l\in P_i} R_l \geq \operatorname{rank}(\overline{\boldsymbol{W}}_i - \prod_{l\in P_i} \boldsymbol{W}_l)$ for all $i \in [\bar{L}]$, there exists LoRA adapters $(\Delta \boldsymbol{W}_l)_{l=1}^L$ with $\operatorname{rank}(\Delta \boldsymbol{W}_l) \leq R_l$ and biases $(\widehat{\boldsymbol{b}}_l)_{l=1}^L$ such that the adapted model f can exactly represent the target model.*

Moreover, define the approximation error of the i-th layer as $E_i = \sigma_{\sum_{l\in P_i} R_l+1}(\overline{\boldsymbol{W}}_i - \prod_{l\in P_i} \boldsymbol{W}_l)$, and the magnitude of the parameters and the input as $\beta := \max_{i\in[\bar{L}]} \left(\sqrt{\|\Sigma\|_F} \prod_{j=1}^i \left\|\overline{\boldsymbol{W}}_j\right\|_F + \sum_{j=1}^i \prod_{k=j+1}^{i-1} \left\|\overline{\boldsymbol{W}}_k\right\|_F \left\|\bar{\boldsymbol{b}}_j\right\|_2\right) \bigvee \sqrt{\|\Sigma\|_F}$.

Then, there exists LoRA adapters $(\Delta \boldsymbol{W}_l)_{l=1}^L$ with $\operatorname{rank}(\Delta \boldsymbol{W}_l) \leq R_l$ and biases $(\widehat{\boldsymbol{b}}_l)_{l=1}^L$ such that for any input $\mathbf{x} \in \mathcal{X}$ with $\mathbb{E}\mathbf{x}\mathbf{x}^\top = \Sigma$, the approximation error can be bounded as

$$\mathbb{E}\left\|f(\mathbf{x}) - \bar{f}(\mathbf{x})\right\|_2 \leq \beta \sum_{i=1}^{\bar{L}} \max_{k\in[\bar{L}]} \left(\left\|\overline{\boldsymbol{W}}_k\right\|_F + E_k\right)^{\bar{L}-i} E_i.$$

Proof [Proof of Theorem 15] This proof follows the same steps as the proofs of Theorem 3 and Theorem 5, substituting the uniform partition $\mathcal{P}^{\mathrm{u}}$ with the general partition $\mathcal{P}$ and applying Lemma 12 in place of Lemma 11 to derive the desired outcome. ■

Appendix F. Expressive Power of Transformer Networks with LoRA

In this section, we not only provide the proof for the results outlined in Sec. 3, but also introduce the problem setting for TFNs with single-head attention layers and present the corresponding results.

F.1. Approximating Transformer Network with Single-Head Attention Layers

In this part, we outline the problem setting to investigate the expressive power of LoRA in TFNs that utilize single-head attention layers. The primary distinction between this setting and that of TFNs with multi-head attention layers lies in the weight matrices. Specifically, the $\boldsymbol{W}_{Ol}^h$ matrices for combining different attention heads are absent in this case. Despite this difference, the derived results are consistent, albeit under slightly modified assumptions regarding the weight matrices and a different LoRA adaptation strategy.

We start by introducing necessary notations. For an input matrix $\boldsymbol{X} \in \mathbb{R}^{D\times N}$, where D is the dimension of the token embeddings and N is the number of tokens, the l-th Transformer block using single-head self-attention can be expressed as:

$$\begin{aligned}
\texttt{Attn}_l(\boldsymbol{Z}_{l-1}) &= \boldsymbol{W}_{Vl}\boldsymbol{Z}_{l-1} \cdot \operatorname{softmax}\left((\boldsymbol{W}_{Kl}\boldsymbol{Z}_{l-1})^\top \boldsymbol{W}_{Ql}\boldsymbol{Z}_{l-1}\right), \\
\boldsymbol{Z}_l &:= \boldsymbol{W}_{2l} \cdot \texttt{ReLU}(\boldsymbol{W}_{1l} \cdot \texttt{Attn}_l(\boldsymbol{Z}_{l-1}) + \boldsymbol{b}_{1l}\mathbf{1}_N^\top) + \boldsymbol{b}_{2l}\mathbf{1}_N^\top,
\end{aligned}$$

where the weight matrices $\boldsymbol{W}_{Kl}, \boldsymbol{W}_{Ql}, \boldsymbol{W}_{Vl}, \boldsymbol{W}_{1l}, \boldsymbol{W}_{2l} \in \mathbb{R}^{D\times D}$, bias vectors $\boldsymbol{b}_{1l}, \boldsymbol{b}_{2l\in\mathbb{R}^D}$, $\boldsymbol{Z}_l$ is the output of l-th transformer block, with $\boldsymbol{Z}_0 = \boldsymbol{X}$. The output of the first L Transformer blocks

are subsequently fed into the output layer. This produces the final output of the TFN, given by $\mathrm{softmax}(\boldsymbol{W}_o \boldsymbol{Z}_L)$, where $\boldsymbol{W}_o \in \mathbb{R}^{D\times D}$ represents the weight matrix of the output layer.

For single-head self-attention layers, the target model $\overline{f}$, frozen model f, and the adapted model f can be formally represented as:

$$
\begin{aligned}
\text{Target TFN} \quad & g = \mathtt{TFN}_{L,D}\left(\cdot; \left((\overline{\boldsymbol{W}}_{Vl}, \overline{\boldsymbol{W}}_{Kl}, \overline{\boldsymbol{W}}_{Ql}, \overline{\boldsymbol{W}}_{2l}, \overline{\boldsymbol{W}}_{1l})_{l=1}^{L}, \overline{\boldsymbol{W}}_o\right), (\bar{\boldsymbol{b}}_{1l}, \bar{\boldsymbol{b}}_{2l})_{l=1}^{L}\right), \\
\text{Frozen TFN} \quad & f_0 = \mathtt{TFN}_{L,D}\left(\cdot; \left((\boldsymbol{W}_{Vl}, \boldsymbol{W}_{Kl}, \boldsymbol{W}_{Ql}, \boldsymbol{W}_{2l}, \boldsymbol{W}_{1l})_{l=1}^{L}, \boldsymbol{W}_o\right), (\boldsymbol{b}_{1l}, \boldsymbol{b}_{2l})_{l=1}^{L}\right), \\
\text{Adapted TFN} \quad & f = \mathtt{TFN}_{L,D}\Big(\cdot; \big((\boldsymbol{W}_{Vl} + \Delta\boldsymbol{W}_{Vl}, \boldsymbol{W}_{Kl} + \Delta\boldsymbol{W}_{Kl}, \boldsymbol{W}_{Ql} + \Delta\boldsymbol{W}_{Ql}, \\
& \qquad \boldsymbol{W}_{2l} + \Delta\boldsymbol{W}_{2l}, \boldsymbol{W}_{1l} + \Delta\boldsymbol{W}_{1l})_{l=1}^{L}, \boldsymbol{W}_o + \Delta\boldsymbol{W}_o\big), (\widehat{\boldsymbol{b}}_{1l}, \widehat{\boldsymbol{b}}_{2l})_{l=1}^{L}\Big).
\end{aligned}
$$

Here, $\overline{\boldsymbol{W}}_{Kl}, \overline{\boldsymbol{W}}_{Ql}, \overline{\boldsymbol{W}}_{Vl}$ are the weight matrices for generating key, query, and values in the l-th transformer block of the target TFN; $\overline{\boldsymbol{W}}_{1l}, \overline{\boldsymbol{W}}_{2l}$ and $\bar{\boldsymbol{b}}_{1l}, \bar{\boldsymbol{b}}_{2l}$ serve as the weight matrices and bias vectors, respectively, for the feedforward layer in the same block; $\overline{\boldsymbol{W}}_o$ is the weight matrix for the output layer. For the frozen TFN, the same roles are played by $\boldsymbol{W}_{Kl}, \boldsymbol{W}_{Ql}, \boldsymbol{W}_{Vl}$, $\boldsymbol{W}_{1l}, \boldsymbol{W}_{2l}$, and $\boldsymbol{b}_{1l}, \boldsymbol{b}_{2l}$ for all $l \in [L]$ and $\boldsymbol{W}_o$. For the adapted model, low-rank adapters $\Delta\boldsymbol{W}_{Kl}, \Delta\boldsymbol{W}_{Ql}, \Delta\boldsymbol{W}_{Vl}, \Delta\boldsymbol{W}_{1l}, \Delta\boldsymbol{W}_{2l}, \Delta\boldsymbol{W}_o$ with a rank constraint $R \in [D]$ are added to each weight matrix, and the bias vectors are updated to $\widehat{\boldsymbol{b}}_{1l}, \widehat{\boldsymbol{b}}_{2l}$ for all $l \in [L]$.

Given the problem setting outlined above, we give the non-singularity assumption for TFNs with single-head attention layers.

Assumption 3 (Non-Singularity) *All the weight matrices of both the target model and the frozen model, as well as the following matrices for all $r \in [D]$,*

$$
\begin{gathered}
\boldsymbol{W}_{Kl}^\top \boldsymbol{W}_{Ql} + \mathrm{LR}_r\left(\overline{\boldsymbol{W}}_{Kl}^\top \overline{\boldsymbol{W}}_{Ql} - \boldsymbol{W}_{Kl}^\top \boldsymbol{W}_{Ql}\right), \text{ where } l = 1, \\
\boldsymbol{W}_{Kl}\boldsymbol{W}_{Ql} + \mathrm{LR}_r\left(\boldsymbol{W}_{2,l-1}^{-1\top}\overline{\boldsymbol{W}}_{2,l-1}^\top \overline{\boldsymbol{W}}_{Kl}^\top \overline{\boldsymbol{W}}_{Ql}\overline{\boldsymbol{W}}_{2,l-1}\boldsymbol{W}_{2,l-1}^{-1} - \boldsymbol{W}_{Kl}\boldsymbol{W}_{Ql}\right), \text{ for } l \in [L]/\{1\}, \\
\boldsymbol{W}_{1l}\boldsymbol{W}_{Vl} + \mathrm{LR}_r\left(\overline{\boldsymbol{W}}_{1l}\overline{\boldsymbol{W}}_{Vl} - \boldsymbol{W}_{1l}\boldsymbol{W}_{Vl}\right), \text{ for } l = 1, \\
\boldsymbol{W}_{1l}\boldsymbol{W}_{Vl} + \mathrm{LR}_r\left(\overline{\boldsymbol{W}}_{1l}\overline{\boldsymbol{W}}_{Vl}\overline{\boldsymbol{W}}_{2,l-1}\boldsymbol{W}_{2,l-1}^{-1} - \boldsymbol{W}_{1l}\boldsymbol{W}_{Vl}\right), \text{ for all } l \in [L]/\{1\}, \\
\boldsymbol{W}_o\boldsymbol{W}_{2L} + \mathrm{LR}_r(\overline{\boldsymbol{W}}_o\overline{\boldsymbol{W}}_{2L} - \boldsymbol{W}_o\boldsymbol{W}_{2L}),
\end{gathered}
$$

are non-singular.

Lemma 16 *Let the elements of all weight matrices in target model $\overline{f}$ and the frozen model f be independently sampled from continuous distributions. Then, Assumption 3 holds with probability 1.*

Proof [Proof of Lemma 16] The results can be obtained by replicating the same steps outlined in the proof of Lemma 13. ■

Theorem 17 *Consider the rank of the adapter weight matrices $R \in [D]$. Let Assumption 3 hold. Define the rank-based functionality gap G_i to i-th transformer block ($i \in [L]$) or output layer*

($i = L + 1$) as

$$G_i = \begin{cases} \max_h \left(\text{rank}(\overline{\boldsymbol{W}}_{Ki}^{h\top}\overline{\boldsymbol{W}}_{Qi}^{h} - \boldsymbol{W}_{Ki}^{h\top}\boldsymbol{W}_{Qi}^{h})\right) \vee \max_h \left(\text{rank}(\overline{\boldsymbol{W}}_{1i}\overline{\boldsymbol{W}}_{Vi}^{h} - \boldsymbol{W}_{1i}\boldsymbol{W}_{Vi}^{h})\right), & i = 1, \\ \max_h \left(\text{rank}(\overline{\boldsymbol{W}}_{2,i-1}^{\top}\overline{\boldsymbol{W}}_{Ki}^{h\top}\overline{\boldsymbol{W}}_{Qi}^{h}\overline{\boldsymbol{W}}_{2,i-1} - \boldsymbol{W}_{2,i-1}^{\top}\boldsymbol{W}_{Ki}^{h\top}\boldsymbol{W}_{Qi}^{h}\boldsymbol{W}_{2,i-1})\right) & \\ \qquad \vee \max_h \left(\text{rank}(\overline{\boldsymbol{W}}_{1i}\overline{\boldsymbol{W}}_{Vi}^{h}\overline{\boldsymbol{W}}_{2,i-1} - \boldsymbol{W}_{1i}\boldsymbol{W}_{Vi}^{h}\boldsymbol{W}_{2,i-1})\right), & 2 \le i \le L, \\ \text{rank}(\overline{\boldsymbol{W}}_{o}\overline{\boldsymbol{W}}_{2L} - \boldsymbol{W}_{o}\boldsymbol{W}_{2L}), & i = L + 1. \end{cases}$$

If $R \ge \max_{i\in[L+1]} \lceil \frac{G_i}{2} \rceil$, there exists rank-$R$ or lower weight matrices for low-rank adapters $(\Delta\boldsymbol{W}_{Kl}, \Delta\boldsymbol{W}_{Ql}, \Delta\boldsymbol{W}_{Vl}, \Delta\boldsymbol{W}_{1l})_{l=1}^{L}, \Delta\boldsymbol{W}_{2L}, \Delta\boldsymbol{W}_{o}$ with other low-rank adapters set to $\boldsymbol{O}$, and updated bias vectors: $(\widehat{\boldsymbol{b}}_{1l}, \widehat{\boldsymbol{b}}_{2l})_{l=1}^{L}$, such that for any $\boldsymbol{X} \in \mathbb{R}^{D\times N}$, the adapted model f exactly represents $\overline{f}$, i.e., $f(\boldsymbol{X}) = \overline{f}(\boldsymbol{X})$, with probability 1.

Proof [Proof of Theorem 17] Let $\overline{\boldsymbol{H}}_l \in \mathbb{R}^{D\times N}$ and $\overline{\boldsymbol{Z}}_l \in \mathbb{R}^{D\times N}$ denote the intermediate and final outputs of the l-th transformer block in the target model $\overline{f}$, respectively. Specifically, $\overline{\boldsymbol{H}}_l$ represents the output from the first feedforward layer in the l-th transformer block. They are defined as

$$\begin{aligned} \overline{\boldsymbol{H}}_l &= \texttt{ReLU}\left(\overline{\boldsymbol{W}}_{1l}\overline{\boldsymbol{W}}_{Vl}\overline{\boldsymbol{Z}}_{l-1} \cdot \text{softmax}\left(\overline{\boldsymbol{Z}}_{l-1}^{\top}\overline{\boldsymbol{W}}_{Kl}^{\top}\overline{\boldsymbol{W}}_{Ql}\overline{\boldsymbol{Z}}_{l-1}\right) + \bar{\boldsymbol{b}}_{1l}\boldsymbol{1}_N^{\top}\right), \\ \overline{\boldsymbol{Z}}_l &= \overline{\boldsymbol{W}}_{2l}\overline{\boldsymbol{H}}_l + \bar{\boldsymbol{b}}_{2l}\boldsymbol{1}_N^{\top}, \end{aligned}$$

where $l \in [L]$. For the adapted model f, we introduce $\widehat{\boldsymbol{H}}_l$ and $\widehat{\boldsymbol{Z}}_l$ to denote the corresponding intermediate output of the first feedforward layer and the final output of the l-th transformer block for the adapted model, respectively:

$$\begin{aligned} \widehat{\boldsymbol{H}}_l &= \texttt{ReLU}\Bigg((\boldsymbol{W}_{1l} + \Delta\boldsymbol{W}_{1l})(\boldsymbol{W}_{Vl} + \Delta\boldsymbol{W}_{Vl}) \cdot \widehat{\boldsymbol{Z}}_{l-1} \\ &\qquad \cdot \text{softmax}\left(\widehat{\boldsymbol{Z}}_{l-1}^{\top}(\boldsymbol{W}_{Kl} + \Delta\boldsymbol{W}_{Kl})^{\top}(\boldsymbol{W}_{Ql} + \Delta\boldsymbol{W}_{Ql})\widehat{\boldsymbol{Z}}_{l-1}\right) + \widehat{\boldsymbol{b}}_{1l}\boldsymbol{1}_N^{\top}\Bigg), \\ \widehat{\boldsymbol{Z}}_l &= (\boldsymbol{W}_{2l} + \Delta\boldsymbol{W}_{2l})\widehat{\boldsymbol{H}}_l + \widehat{\boldsymbol{b}}_{2l}\boldsymbol{1}_N^{\top}, \end{aligned}$$

where $l \in [L]$. We note that $\overline{\boldsymbol{Z}}_0 = \widehat{\boldsymbol{Z}}_0 = \boldsymbol{X}$.

In this proof, we set $\Delta\boldsymbol{W}_{2l} = \boldsymbol{O}$ for all $l \in [L]$. Our goal is to show that adding low-rank adapters to self-attention layers and the first feedforward layers in all transformer blocks enables the adapted model f to be functionally equivalent to the target model $\overline{f}$ of the same dimensions. We start by inductively constructing the adapter weight matrices $(\Delta\boldsymbol{W}_{1l}, \Delta\boldsymbol{W}_{Vl}, \Delta\boldsymbol{W}_{Kl}, \Delta\boldsymbol{W}_{Ql}, \widehat{\boldsymbol{b}}_{1l}, \widehat{\boldsymbol{b}}_{2l})_{l=1}^{L}$ such that $\widehat{\boldsymbol{H}}_l = \overline{\boldsymbol{H}}_l$ for all $l \in [L]$. We then select the low-rank adapters for $\boldsymbol{W}_{2L}$ and the $\boldsymbol{W}_o$ to approximate the output of the target model. For unmentioned low-rank adapters, we set them as $\boldsymbol{O}$.

When $l = 1$. To achieve $\widehat{\boldsymbol{H}}_l$ with $\overline{\boldsymbol{H}}_l$ for all $\boldsymbol{X}$, the following conditions must be satisfied:

Bias Vector: $\widehat{\boldsymbol{b}}_{1l} = \bar{\boldsymbol{b}}_{1l}$,

Query and Key: $(\boldsymbol{W}_{Kl} + \Delta\boldsymbol{W}_{Kl})^{\top}(\boldsymbol{W}_{Ql} + \Delta\boldsymbol{W}_{Ql}) = \overline{\boldsymbol{W}}_{Kl}^{\top}\overline{\boldsymbol{W}}_{Ql}$

Value and First Feedforward Layer: $(\boldsymbol{W}_{1l} + \Delta\boldsymbol{W}_{1l})(\boldsymbol{W}_{Vl} + \Delta\boldsymbol{W}_{Vl}) = \overline{\boldsymbol{W}}_{1l}\overline{\boldsymbol{W}}_{Vl}$.

To achieve this, we set $\widehat{\boldsymbol{b}}_{1l} = \overline{\boldsymbol{b}}_{1l}$ to achieve (23), and select rank-R or lower matrices $\Delta\boldsymbol{W}_{Kl}, \Delta\boldsymbol{W}_{Ql}, \Delta\boldsymbol{W}_{1l}, \Delta\boldsymbol{W}_{Vl}$ as suggested by Lemma 9. This ensures $\widehat{\boldsymbol{H}}_l = \overline{\boldsymbol{H}}_l$ for $l = 1$.

When $l > 1$. Now we focus on the cases where $l = 2, \ldots, L$. Assume the induction hypothesis holds for $l-1$, which is $\widehat{\boldsymbol{H}}_{l-1} = \overline{\boldsymbol{H}}_{l-1}$. This implies

$$\overline{\boldsymbol{H}}_{l-1} = \overline{\boldsymbol{W}}_{2,l-1}^{-1}(\overline{\boldsymbol{Z}}_{l-1} - \overline{\boldsymbol{b}}_{2,l-1}\mathbf{1}_N^\top) = \boldsymbol{W}_{2,l-1}^{-1}(\widehat{\boldsymbol{Z}}_{l-1} - \widehat{\boldsymbol{b}}_{2,l-1}\mathbf{1}_N^\top) = \widehat{\boldsymbol{H}}_{l-1}.$$

Using this assumption, we express $\widehat{\boldsymbol{Z}}_{l-1}$ in terms of $\overline{\boldsymbol{Z}}_{l-1}$:

$$\widehat{\boldsymbol{Z}}_{l-1} = \boldsymbol{W}_{2,l-1}\overline{\boldsymbol{W}}_{2,l-1}^{-1}(\overline{\boldsymbol{Z}}_{l-1} - \overline{\boldsymbol{b}}_{2,l-1}\mathbf{1}_N^\top) + \widehat{\boldsymbol{b}}_{2,l-1}\mathbf{1}_N^\top.$$

Let $\widehat{\boldsymbol{b}}_{2,l-1} = \boldsymbol{W}_{2,l-1}\overline{\boldsymbol{W}}_{2,l-1}^{-1}\overline{\boldsymbol{b}}_{2,l-1}$, then we have

$$\widehat{\boldsymbol{Z}}_{l-1} = \boldsymbol{W}_{2,l-1}\overline{\boldsymbol{W}}_{2,l-1}^{-1}\overline{\boldsymbol{Z}}_{l-1}. \tag{22}$$

To achieve $\widehat{\boldsymbol{H}}_l = \overline{\boldsymbol{H}}_l$, we express both $\widehat{\boldsymbol{H}}_l$ and $\overline{\boldsymbol{H}}_l$ in terms of $\overline{\boldsymbol{Z}}_{l-1}$:

$$\begin{aligned}
\overline{\boldsymbol{H}}_l &= \texttt{ReLU}\Big(\overline{\boldsymbol{W}}_{1l}\overline{\boldsymbol{W}}_{Vl} \cdot \overline{\boldsymbol{Z}}_{l-1} \cdot \mathrm{softmax}\left(\overline{\boldsymbol{Z}}_{l-1}^\top \overline{\boldsymbol{W}}_{Kl}^\top \overline{\boldsymbol{W}}_{Ql}\overline{\boldsymbol{Z}}_{l-1}\right) + \overline{\boldsymbol{b}}_{1l}\mathbf{1}_N^\top\Big) \\
\widehat{\boldsymbol{H}}_l &= \texttt{ReLU}\Big((\boldsymbol{W}_{1l} + \Delta\boldsymbol{W}_{1l})(\boldsymbol{W}_{Vl} + \Delta\boldsymbol{W}_{Vl}) \cdot \widehat{\boldsymbol{Z}}_{l-1} \\
&\qquad \cdot \mathrm{softmax}\left(\widehat{\boldsymbol{Z}}_{l-1}^\top (\boldsymbol{W}_{Kl} + \Delta\boldsymbol{W}_{Kl})^\top (\boldsymbol{W}_{Ql} + \Delta\boldsymbol{W}_{Ql})\widehat{\boldsymbol{Z}}_{l-1}\right) + \widehat{\boldsymbol{b}}_{1l}\mathbf{1}_N^\top\Big), \\
&\overset{(22)}{=} \texttt{ReLU}\Bigg((\boldsymbol{W}_{1l} + \Delta\boldsymbol{W}_{1l})(\boldsymbol{W}_{Vl} + \Delta\boldsymbol{W}_{Vl}) \cdot \boldsymbol{W}_{2,l-1}\overline{\boldsymbol{W}}_{2,l-1}^{-1}\overline{\boldsymbol{Z}}_{l-1} \\
&\qquad \cdot \mathrm{softmax}\Big(\overline{\boldsymbol{Z}}_{l-1}^\top \overline{\boldsymbol{W}}_{2,l-1}^{-1\top}\boldsymbol{W}_{2,l-1}^\top (\boldsymbol{W}_{Kl} + \Delta\boldsymbol{W}_{Kl})^\top \\
&\qquad\qquad (\boldsymbol{W}_{Ql} + \Delta\boldsymbol{W}_{Ql})\boldsymbol{W}_{2,l-1}\overline{\boldsymbol{W}}_{2,l-1}^{-1}\overline{\boldsymbol{Z}}_{l-1}\Big) + \widehat{\boldsymbol{b}}_{1l}\mathbf{1}_N^\top\Bigg).
\end{aligned}$$

Therefore, we need to align the following three components:

Bias Vector: $\widehat{\boldsymbol{b}}_{1l} = \overline{\boldsymbol{b}}_{1l}$,

Query and Key: $(\boldsymbol{W}_{Kl} + \Delta\boldsymbol{W}_{Kl})^\top(\boldsymbol{W}_{Ql} + \Delta\boldsymbol{W}_{Ql}) = \boldsymbol{W}_{2,l-1}^{-1\top}\overline{\boldsymbol{W}}_{2,l-1}^\top \overline{\boldsymbol{W}}_{Kl}^\top \overline{\boldsymbol{W}}_{Ql}\overline{\boldsymbol{W}}_{2,l-1}\boldsymbol{W}_{2,l-1}^{-1}$,

Value and First Feedforward Layer: $(\boldsymbol{W}_{1l} + \Delta\boldsymbol{W}_{1l})(\boldsymbol{W}_{Vl} + \Delta\boldsymbol{W}_{Vl}) = \overline{\boldsymbol{W}}_{1l}\overline{\boldsymbol{W}}_{Vl}\overline{\boldsymbol{W}}_{2,l-1}\boldsymbol{W}_{2,l-1}^{-1}$.

By setting $\widehat{\boldsymbol{b}}_{1l}$ based on (25) and adjusting $\Delta\boldsymbol{W}_{Kl}, \Delta\boldsymbol{W}_{Ql}, \Delta\boldsymbol{W}_{1l}, \Delta\boldsymbol{W}_{Vl}$ based on Lemma 9, we satisfy all three conditions above, thereby obtaining $\widehat{\boldsymbol{H}}_l = \overline{\boldsymbol{H}}_l$ for $l \in [L]/\{1\}$.

Output Layer Analysis. By the induction method, we have established $\widehat{\boldsymbol{H}}_l = \overline{\boldsymbol{H}}_l$ for all $l \in [L]$. We will complete the proof by showing that $\overline{f}(\boldsymbol{X}) = f(\boldsymbol{X})$ for all $\boldsymbol{X} \in \mathcal{X}$.

The final output distribution of the target TFN $\overline{f}$ can be written as

$$\overline{f}(\boldsymbol{X}) = \mathrm{softmax}(\overline{\boldsymbol{W}}_o\overline{\boldsymbol{Z}}_L) = \mathrm{softmax}\left(\overline{\boldsymbol{W}}_o\left(\overline{\boldsymbol{W}}_{2L}\overline{\boldsymbol{H}}_L + \overline{\boldsymbol{b}}_{2L}\mathbf{1}_N^\top\right)\right).$$

We can similarly formulate the final output distribution of the adapted model f :

$$\begin{aligned} f(\boldsymbol{X}) &= \operatorname{softmax}((\boldsymbol{W}_o + \Delta\boldsymbol{W}_o)\widehat{\boldsymbol{Z}}_L) \\ &= \operatorname{softmax}\left((\boldsymbol{W}_o + \Delta\boldsymbol{W}_o)\left((\boldsymbol{W}_{2L} + \Delta\boldsymbol{W}_{2L})\widehat{\boldsymbol{H}}_L + \widehat{\boldsymbol{b}}_{2L}\mathbf{1}_N^\top\right)\right), \end{aligned}$$

To align these two expressions, we select $\Delta\boldsymbol{W}_{2L}$ and $\Delta\boldsymbol{W}_o$ based on Lemma 9, and let $\widehat{\boldsymbol{b}}_{2L} = (\boldsymbol{W}_o + \Delta\boldsymbol{W}_o)^{-1}\overline{\boldsymbol{W}}_o\bar{\boldsymbol{b}}_{2L}$, where $\boldsymbol{W}_o + \Delta\boldsymbol{W}_o$ is invertible as shown in the proof of Lemma 9. Thus, the proof is complete. ■

The following corollary identifies the specific LoRA-rank required to achieve exact representation for random model cases in the current setting.

Corollary 18 *Assume that the elements of all the weight matrices of both the target TFN and the frozen TFN are independently drawn from arbitrary continuous distributions. If $R \geq \lceil \frac{D}{2} \rceil$, adding low-rank adapters of rank at most R to weight matrices in $(\Delta\boldsymbol{W}_{Kl}, \Delta\boldsymbol{W}_{Ql}, \Delta\boldsymbol{W}_{Vl}, \Delta\boldsymbol{W}_{1l})_{l=1}^{L}, \Delta\boldsymbol{W}_{2L}, \Delta\boldsymbol{W}_o$ and tuning the bias vectors, enables the adapted model f to exactly represent the target model $\bar{f}$, i.e., $f(\boldsymbol{X}) = \bar{f}(\boldsymbol{X})$ for all $\boldsymbol{X} \in \mathbb{R}^{D\times N}$.*

Proof [Proof of Corollary 18] By combining Lemma 16 and Theorem 17, and following the same steps in the proof of Corollary 4 which yields $\max_i G_i = D$, we can obtain the desired outcome. ■

F.2. Approximating Transformer Network with Multi-Head Attention Layers

In this section, we first provide the non-singularity Assumption 4 for TFN with multi-head attention layers scenarios, which is then validated by Lemma 19. We then analyze the expressive power of LoRA on TFNs with multi-head attention layers, as described in the proof of Theorem 6. Additionally, we introduce a supplementary theorem that amalgamates results for TFNs with both single-head and multi-head attention layers when the weight matrices are randomly initialized. This is articulated in Corollary 20.

Assumption 4 (Non-Singularity) *For a fixed $R \in [D]$, all the weight matrices of both the target model and the frozen model and the following matrices for all $r \in [R]$,*

$$\begin{gathered} \boldsymbol{W}_{Kl}^{h\top}\boldsymbol{W}_{Ql}^h + \mathrm{LR}_r\left(\overline{\boldsymbol{W}}_{Kl}^{h\top}\overline{\boldsymbol{W}}_{Ql}^h - \boldsymbol{W}_{Kl}^{h\top}\boldsymbol{W}_{Ql}^h\right), \text{ for all } h \in [H] \text{ and } l = 1, \\ \boldsymbol{W}_{Kl}^{h\top}\boldsymbol{W}_{Ql}^h + \mathrm{LR}_r\left(\boldsymbol{W}_{2,l-1}^{-1\top}\overline{\boldsymbol{W}}_{2,l-1}^{\top}\overline{\boldsymbol{W}}_{Kl}^{h\top}\overline{\boldsymbol{W}}_{Ql}^h\overline{\boldsymbol{W}}_{2,l-1}\boldsymbol{W}_{2,l-1}^{-1} - \boldsymbol{W}_{Kl}^{h\top}\boldsymbol{W}_{Ql}^h\right), \text{ for all } h \in [H] \text{ and } l \in [L]/\{1\}, \\ \boldsymbol{W}_{Ol}^h\boldsymbol{W}_{Vl}^h + \mathrm{LR}_r\left(\boldsymbol{W}_{1l}^{-1}\overline{\boldsymbol{W}}_{1l}\overline{\boldsymbol{W}}_{Ol}^h\overline{\boldsymbol{W}}_{Vl}^h - \boldsymbol{W}_{Ol}^h\boldsymbol{W}_{Vl}^h\right), \text{ for all } h \in [H] \text{ and } l = 1, \\ \boldsymbol{W}_{Ol}^h\boldsymbol{W}_{Vl}^h + \mathrm{LR}_r\left(\boldsymbol{W}_{1l}^{-1}\overline{\boldsymbol{W}}_{1l}\overline{\boldsymbol{W}}_{Ol}^h\overline{\boldsymbol{W}}_{Vl}^h\overline{\boldsymbol{W}}_{2,l-1}\boldsymbol{W}_{2,l-1}^{-1} - \boldsymbol{W}_{Ol}^h\boldsymbol{W}_{Vl}^h\right), \text{ for all } h \in [H] \text{ and } l \in [L]/\{1\}, \\ \boldsymbol{W}_o\boldsymbol{W}_{2L} + \mathrm{LR}_r(\overline{\boldsymbol{W}}_o\overline{\boldsymbol{W}}_{2L} - \boldsymbol{W}_o\boldsymbol{W}_{2L}), \end{gathered}$$

are non-singular.

Lemma 19 *Let the elements of all weight matrices in the target model $\bar{f}$ and frozen model f_0 be independently sampled from continuous distributions. Then, Assumption 4 holds with probability 1.*

Proof [Proof of Lemma 19] The results can be obtained by replicating the same steps outlined in the proof of Lemma 13. ■

For the reader's reference, we restate Theorem 6 here.

Theorem 6 *Consider a given LoRA-rank $R \in [D]$. Let Assumption 4 hold. Define the rank-based functionality gap G_i to i-th transformer block ($i \in [L]$) or output layer ($i = L+1$) as*

$$G_i = \begin{cases} \max_h \left(\operatorname{rank}(\overline{\boldsymbol{W}}_{Ki}^{h\top}\overline{\boldsymbol{W}}_{Qi}^{h} - \boldsymbol{W}_{Ki}^{h\top}\boldsymbol{W}_{Qi}^{h})\right) \vee \max_h \left(\operatorname{rank}(\overline{\boldsymbol{W}}_{1i}\overline{\boldsymbol{W}}_{Oi}^{h}\overline{\boldsymbol{W}}_{Vi}^{h} - \boldsymbol{W}_{1i}\boldsymbol{W}_{Oi}^{h}\boldsymbol{W}_{Vi}^{h})\right), & i = 1, \\ \max_h \Big(\operatorname{rank}(\overline{\boldsymbol{W}}_{2,i-1}^{\top}\overline{\boldsymbol{W}}_{Ki}^{h\top}\overline{\boldsymbol{W}}_{Qi}^{h}\overline{\boldsymbol{W}}_{2,i-1} - \boldsymbol{W}_{2,i-1}^{\top}\boldsymbol{W}_{Ki}^{h\top}\boldsymbol{W}_{Qi}^{h}\boldsymbol{W}_{2,i-1})\Big) & \\ \qquad \vee \max_h \left(\operatorname{rank}(\overline{\boldsymbol{W}}_{1i}\overline{\boldsymbol{W}}_{Oi}^{h}\overline{\boldsymbol{W}}_{Vi}^{h}\overline{\boldsymbol{W}}_{2,i-1} - \boldsymbol{W}_{1i}\boldsymbol{W}_{Oi}^{h}\boldsymbol{W}_{Vi}^{h}\boldsymbol{W}_{2,i-1})\right), & 2 \le i \le L, \\ \operatorname{rank}(\overline{\boldsymbol{W}}_{o}\overline{\boldsymbol{W}}_{2L} - \boldsymbol{W}_{o}\boldsymbol{W}_{2L}), & i = L+1. \end{cases}$$

If $R \ge \max_{i\in[L+1]} \lceil \frac{G_i}{2} \rceil$, then there exists low-rank adapters with rank lower than $R \in [D]$ $((\Delta\boldsymbol{W}_{Kl}^{h}, \Delta\boldsymbol{W}_{Ql}^{h}, \Delta\boldsymbol{W}_{Vl}^{h}, \Delta\boldsymbol{W}_{Ol}^{h})_{h=1}^{H})_{l=1}^{L}, \Delta\boldsymbol{W}_{2L}, \Delta\boldsymbol{W}_{o}$ with other low-rank adapters set to $\boldsymbol{O}$, and updated bias vectors $(\widehat{\boldsymbol{b}}_{1l}, \widehat{\boldsymbol{b}}_{2l})_{l=1}^{L}$, such that for any $\boldsymbol{X} \in \mathbb{R}^{D\times N}$, the adapted model f exactly represents target model $\overline{f}$, i.e., $f(\boldsymbol{X}) = \overline{f}(\boldsymbol{X})$.

Proof [Proof of Theorem 6] The key idea of this proof is the same as the proof of Theorem 17: our first step is to ensure that, for each transformer block, the output from the first feedforward layer in the target model matches that in the adapted model. Once this is established, we select an appropriate output layer weight matrix to complete the proof.

Similar to the proof of Theorem 17, we define $\overline{\boldsymbol{H}}_l \in \mathbb{R}^{D\times N}$ and $\overline{\boldsymbol{Z}}_l \in \mathbb{R}^{D\times N}$ as the intermediate and final outputs of the l-th transformer block in the target model $\overline{f}$, respectively. In particular, $\overline{\boldsymbol{H}}_l$ corresponds to the output of the first feedforward layer in the l-th transformer block. They are formulated as

$$\begin{aligned} \overline{\boldsymbol{H}}_l &= \texttt{ReLU}\left(\overline{\boldsymbol{W}}_{1l}\left(\sum_{h=1}^{H}\overline{\boldsymbol{W}}_{Ol}^{h}\overline{\boldsymbol{W}}_{Vl}^{h}\cdot\overline{\boldsymbol{Z}}_{l-1}\cdot\operatorname{softmax}\left(\overline{\boldsymbol{Z}}_{l-1}^{\top}\overline{\boldsymbol{W}}_{Kl}^{h\top}\overline{\boldsymbol{W}}_{Ql}^{h}\overline{\boldsymbol{Z}}_{l-1}\right)\right) + \bar{\boldsymbol{b}}_{1l}\boldsymbol{1}_N^{\top}\right), \\ \overline{\boldsymbol{Z}}_l &= \overline{\boldsymbol{W}}_{2l}\overline{\boldsymbol{H}}_l + \bar{\boldsymbol{b}}_{2l}\boldsymbol{1}_N^{\top}. \end{aligned}$$

For the adapted model f, we introduce $\widehat{\boldsymbol{H}}_l$ and $\widehat{\boldsymbol{Z}}_l$ accordingly to denote the intermediate output of the first feedforward layer and the final output of the l-th transformer block for the adapted model, respectively:

$$\begin{aligned} \widehat{\boldsymbol{H}}_l = \texttt{ReLU}\Bigg(\boldsymbol{W}_{1l}\Big(\sum_{h=1}^{H}(\boldsymbol{W}_{Ol}^{h}+\Delta\boldsymbol{W}_{Ol}^{h})(\boldsymbol{W}_{Vl}^{h}+\Delta\boldsymbol{W}_{Vl}^{h})\cdot\widehat{\boldsymbol{Z}}_{l-1} \\ \cdot\operatorname{softmax}\left(\widehat{\boldsymbol{Z}}_{l-1}^{\top}(\boldsymbol{W}_{Kl}^{h}+\Delta\boldsymbol{W}_{Kl}^{h})^{\top}(\boldsymbol{W}_{Ql}^{h}+\Delta\boldsymbol{W}_{Ql}^{h})\widehat{\boldsymbol{Z}}_{l-1}\right)\Big) + \widehat{\boldsymbol{b}}_{1l}\boldsymbol{1}_N^{\top}\Bigg), \\ \widehat{\boldsymbol{Z}}_l = \boldsymbol{W}_{2l}\widehat{\boldsymbol{H}}_l + \widehat{\boldsymbol{b}}_{2l}\boldsymbol{1}_N^{\top}. \end{aligned}$$

Note that $\overline{\boldsymbol{Z}}_0 = \widehat{\boldsymbol{Z}}_0 = \boldsymbol{X}$.

We aim to demonstrate that adding low-rank adapters to the weight matrices allows the adapted TFN f to be functionally equivalent to the target TFN of identical dimensions. We will initiate our proof by inductively constructing the adapter weight matrices

$((\Delta \boldsymbol{W}_{Ol}^h, \Delta \boldsymbol{W}_{Vl}^h, \Delta \boldsymbol{W}_{Kl}^h, \Delta \boldsymbol{W}_{Ql}^h)_{h=1}^H, \widehat{\boldsymbol{b}}_{1l}, \widehat{\boldsymbol{b}}_{2l})_{l=1}^L$ such that $\widehat{\boldsymbol{H}}_l = \overline{\boldsymbol{H}}_l$ for all $l \in [L]$, and then select the $\Delta \boldsymbol{W}_{2L}$ and the low-rank adapter for the output layer $\Delta \boldsymbol{W}_o$ to approximate the output of the target model. For unmentioned low-rank adapters, we set them as $\boldsymbol{O}$.

When $l = 1$. To achieve $\widehat{\boldsymbol{H}}_l$ with $\overline{\boldsymbol{H}}_l$ for all $\boldsymbol{X}$, we must satisfy the following conditions:

$$\begin{aligned}
&\text{Bias Vector: } \widehat{\boldsymbol{b}}_{1l} = \bar{\boldsymbol{b}}_{1l}, \\
&\text{Query and Key: } (\boldsymbol{W}_{Kl}^h + \Delta \boldsymbol{W}_{Kl}^h)^\top (\boldsymbol{W}_{Ql}^h + \Delta \boldsymbol{W}_{Ql}^h) = \overline{\boldsymbol{W}}_{Kl}^{h\top} \overline{\boldsymbol{W}}_{Ql}^h, \\
&\text{Value and Output Projection: } (\boldsymbol{W}_{Ol}^h + \Delta \boldsymbol{W}_{Ol}^h)(\boldsymbol{W}_{Vl}^h + \Delta \boldsymbol{W}_{Vl}^h) = \boldsymbol{W}_{1l}^{-1} \overline{\boldsymbol{W}}_{1l} \overline{\boldsymbol{W}}_{Ol}^h \overline{\boldsymbol{W}}_{Vl}^h.
\end{aligned} \tag{23}$$

To achieve this, we set $\widehat{\boldsymbol{b}}_{1l} = \bar{\boldsymbol{b}}_{1l}$ to achieve (23), and select rank-R or lower matrices $\Delta \boldsymbol{W}_{Kl}^h, \Delta \boldsymbol{W}_{Ql}^h, \Delta \boldsymbol{W}_{Ol}^h, \Delta \boldsymbol{W}_{Vl}^h$ for all $h \in [H]$ as suggested by Lemma 9. This ensures $\widehat{\boldsymbol{H}}_l = \overline{\boldsymbol{H}}_l$ for $l = 1$.

When $l > 1$. Now we focus on the cases where $l = 2, \ldots, L$. Assume the induction hypothesis holds for $l-1$, which is $\widehat{\boldsymbol{H}}_{l-1} = \overline{\boldsymbol{H}}_{l-1}$. Following the same steps in the proof of Theorem 17, we let $\widehat{\boldsymbol{b}}_{2,l-1} = \boldsymbol{W}_{2,l-1} \overline{\boldsymbol{W}}_{2,l-1}^{-1} \bar{\boldsymbol{b}}_{2,l-1}$, thereby obtaining,

$$\widehat{\boldsymbol{Z}}_{l-1} = \boldsymbol{W}_{2,l-1} \overline{\boldsymbol{W}}_{2,l-1}^{-1} \overline{\boldsymbol{Z}}_{l-1}. \tag{24}$$

To achieve $\widehat{\boldsymbol{H}}_l = \overline{\boldsymbol{H}}_l$, we express both $\widehat{\boldsymbol{H}}_l$ and $\overline{\boldsymbol{H}}_l$ in terms of $\overline{\boldsymbol{Z}}_{l-1}$:

$$\begin{aligned}
\overline{\boldsymbol{H}}_l &= \texttt{ReLU}\Big(\overline{\boldsymbol{W}}_{1l}\Big(\sum_{h=1}^H \overline{\boldsymbol{W}}_{Ol}^h \overline{\boldsymbol{W}}_{Vl}^h \cdot \overline{\boldsymbol{Z}}_{l-1} \cdot \operatorname{softmax}\left(\overline{\boldsymbol{Z}}_{l-1}^\top \overline{\boldsymbol{W}}_{Kl}^{h\top} \overline{\boldsymbol{W}}_{Ql}^h \overline{\boldsymbol{Z}}_{l-1}\right)\Big) + \bar{\boldsymbol{b}}_{1l} \boldsymbol{1}_N^\top\Big) \\
\widehat{\boldsymbol{H}}_l &= \texttt{ReLU}\Big(\boldsymbol{W}_{1l}\Big(\sum_{h=1}^H (\boldsymbol{W}_{Ol}^h + \Delta \boldsymbol{W}_{Ol}^h)(\boldsymbol{W}_{Vl}^h + \Delta \boldsymbol{W}_{Vl}^h) \cdot \widehat{\boldsymbol{Z}}_{l-1} \\
&\qquad \cdot \operatorname{softmax}\left(\widehat{\boldsymbol{Z}}_{l-1}^\top (\boldsymbol{W}_{Kl}^h + \Delta \boldsymbol{W}_{Kl}^h)^\top (\boldsymbol{W}_{Ql}^h + \Delta \boldsymbol{W}_{Ql}^h) \widehat{\boldsymbol{Z}}_{l-1}\right)\Big) + \widehat{\boldsymbol{b}}_{1l} \boldsymbol{1}_N^\top\Big), \\
&\overset{(24)}{=} \texttt{ReLU}\Bigg(\boldsymbol{W}_{1l}\Big(\sum_{h=1}^H (\boldsymbol{W}_{Ol}^h + \Delta \boldsymbol{W}_{Ol}^h)(\boldsymbol{W}_{Vl}^h + \Delta \boldsymbol{W}_{Vl}^h) \cdot \boldsymbol{W}_{2,l-1} \overline{\boldsymbol{W}}_{2,l-1}^{-1} \overline{\boldsymbol{Z}}_{l-1} \\
&\qquad \cdot \operatorname{softmax}\Big(\overline{\boldsymbol{Z}}_{l-1}^\top \overline{\boldsymbol{W}}_{2,l-1}^{-1\top} \boldsymbol{W}_{2,l-1}^\top (\boldsymbol{W}_{Kl}^h + \Delta \boldsymbol{W}_{Kl}^h)^\top \\
&\qquad\qquad (\boldsymbol{W}_{Ql}^h + \Delta \boldsymbol{W}_{Ql}^h) \boldsymbol{W}_{2,l-1} \overline{\boldsymbol{W}}_{2,l-1}^{-1} \overline{\boldsymbol{Z}}_{l-1}\Big)\Big) + \widehat{\boldsymbol{b}}_{1l} \boldsymbol{1}_N^\top\Bigg).
\end{aligned}$$

Therefore, we need to align the following three components:

$$\begin{aligned}
&\text{Bias Vector: } \widehat{\boldsymbol{b}}_{1l} = \bar{\boldsymbol{b}}_{1l}, \\
&\text{Query and Key: } (\boldsymbol{W}_{Kl}^h + \Delta \boldsymbol{W}_{Kl}^h)^\top (\boldsymbol{W}_{Ql}^h + \Delta \boldsymbol{W}_{Ql}^h) = \boldsymbol{W}_{2,l-1}^{-1\top} \overline{\boldsymbol{W}}_{2,l-1}^\top \overline{\boldsymbol{W}}_{Kl}^{h\top} \overline{\boldsymbol{W}}_{Ql}^h \overline{\boldsymbol{W}}_{2,l-1} \boldsymbol{W}_{2,l-1}^{-1}, \\
&\text{Value and Output Projection: } \\
&\qquad (\boldsymbol{W}_{Ol}^h + \Delta \boldsymbol{W}_{Ol}^h)(\boldsymbol{W}_{Vl}^h + \Delta \boldsymbol{W}_{Vl}^h) = \boldsymbol{W}_{1l}^{-1} \overline{\boldsymbol{W}}_{1l} \overline{\boldsymbol{W}}_{Ol}^h \overline{\boldsymbol{W}}_{Vl}^h \overline{\boldsymbol{W}}_{2,l-1} \boldsymbol{W}_{2,l-1}^{-1}.
\end{aligned} \tag{25}$$

By setting $\widehat{\boldsymbol{b}}_{1l}$ based on (25) and adjusting $\Delta \boldsymbol{W}_{Kl}^h, \Delta \boldsymbol{W}_{Ql}^h, \Delta \boldsymbol{W}_{Ol}^h, \Delta \boldsymbol{W}_{Vl}^h$ for all $h \in [H]$ based on Lemma 9, we satisfy all three conditions above, thereby obtaining $\widehat{\boldsymbol{H}}_l = \overline{\boldsymbol{H}}_l$ for $l \in [L] / \{1\}$.

Output Layer Analysis. By applying the induction method, we have established $\widehat{\boldsymbol{H}}_l = \overline{\boldsymbol{H}}_l$ for all $l \in [L]$. Lastly, we choose the $\Delta \boldsymbol{W}_o$, $\Delta \boldsymbol{W}_{2L}$ and the bias vector $\widehat{\boldsymbol{b}}_{2L}$ using the same approach as in the proof of Theorem 17. This concludes the proof. ■

The following corollary identifies the specific LoRA-rank required to achieve exact representation for random model cases in the current setting.

Corollary 20 *Assume that the elements of all the weight matrices of both the target TFN and the frozen TFN are independently drawn from arbitrary continuous distributions. If $R \geq \lceil \frac{D}{2} \rceil$, adding low-rank adapters of rank at most R to weight matrices in $((\Delta \boldsymbol{W}_{Kl}^h, \Delta \boldsymbol{W}_{Ql}^h, \Delta \boldsymbol{W}_{Vl}^h, \Delta \boldsymbol{W}_{Ol}^h)_{h=1}^H)_{l=1}^L, \Delta \boldsymbol{W}_{2L}, \Delta \boldsymbol{W}_o$ and tuning the bias vectors, enables the adapted model f to exactly represent the target model $\bar{f}$, i.e., $f(\boldsymbol{X}) = \bar{f}(\boldsymbol{X})$ for all $\boldsymbol{X} \in \mathbb{R}^{D \times N}$.*

Proof [Proof of Corollary 20] By combining Lemma 19 and Theorem 6, and following the same steps in the proof of Corollary 4 which yields $\max_i G_i = D$, we can obtain the desired outcome. ■

Appendix G. Numerical Experiments

Recall that all our theoretical statements are based on our construction of the LoRA adapters presented in their corresponding proofs. To validate these results, here we empirically examine the relationship between approximation error and rank by integrating the LoRA adapters, which are constructed with uniform partition in our proof, into the frozen model. Furthermore, we evaluate the effectiveness of our constructed LoRA adapters by comparing their performance against adapters updated through gradient descent and optimized by Adam.

We tune the learning rate $\in \{10^{-2}, 10^{-3}, 10^{-4}\}$ and the weight decay $\in \{0, 10^{-2}, 10^{-3}, 10^{-4}\}$. The optimal configuration is determined based on the validation loss on a set of 256 samples independently drawn from a standard normal distribution. We run 5000 iterations for each hyperparameter setting, where at each step 256 fresh standard Gaussian samples are generated for loss and gradient computation. We implement the LoRA adapters in the same way as Hu et al. [15]. We employ the *Mean Squared Error* (MSE) to measure the approximation error, denoted by $\widehat{\mathbb{E}}_{\mathbf{x}} \left\| f(\mathbf{x}) - \bar{f}(\mathbf{x}) \right\|_2^2$, where the input $\mathbf{x}$ is independently generated from a Gaussian distribution $\mathcal{N}(0, 1)$, and we generate 5000 samples.

G.1. Linear Model Approximation

In this experiment, we validate our results presented in Sec. D.

Model. We consider linear models with $D = 16$ and $L = 2$. We employ two methods to create synthetic models. **(Random)** The first method involves randomly generating all the weight matrices using the Xavier uniform distribution, which is the default weight initialization method used in PyTorch. **(Pretrained)** The second method aims to simulate scenarios where the frozen model has been trained and is relatively closer to the target model. We achieve this by initially creating the target model and the frozen model in the same way as the first method and then performing full-rank updates on the frozen model via gradient descent to approximate the target model until the approximation error is reduced by 1/3.

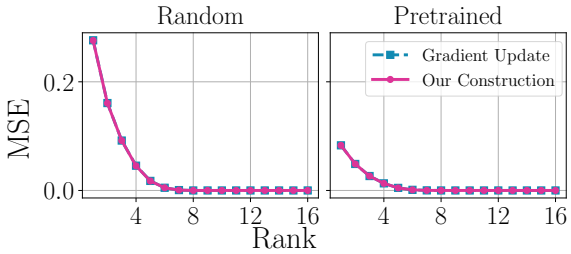
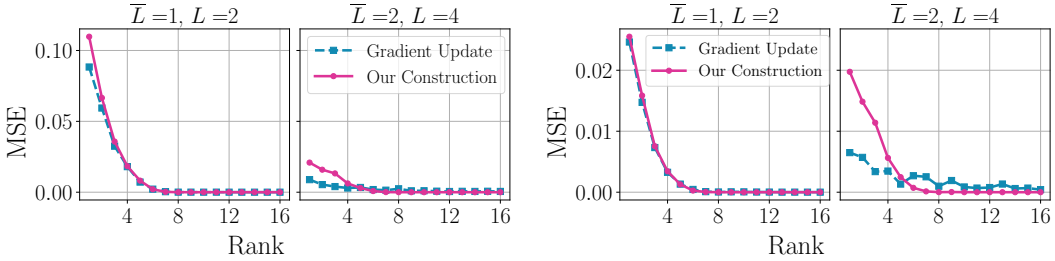


Figure 1: Linear model approximation.



(*a*) Frozen model is randomly generated. (*b*) Frozen model is pretrained.

Figure 2: Approximation error (measured by MSE) versus LoRA-rank on FNNs.

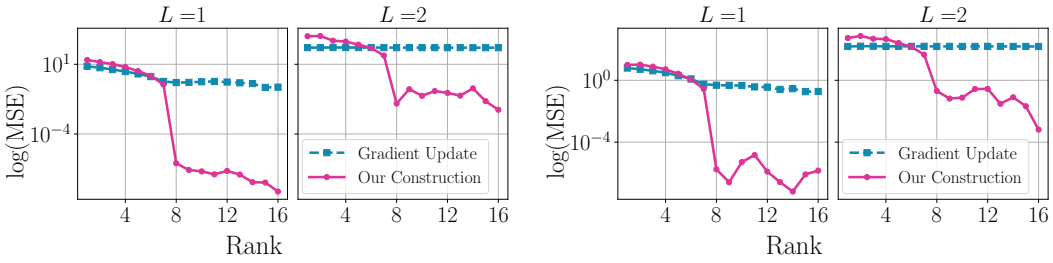
Results. Our results for linear model approximation via LoRA are depicted in Fig.1. Firstly, we observe that the MSE of both two cases is close to zero when $R \geq \frac{D}{L/\overline{L}} = 8$, which corroborates our claims. Meanwhile, a comparison between the left and right columns of Fig.1 suggests that pretraining can further reduce the required rank to achieve near-zero approximation error. Furthermore, the curves of our construction align perfectly with those of the gradient update method in linear model approximation cases, confirming the optimality claimed in Lemma 9.

G.2. FNN Approximation

In this experiment, we assess the effectiveness of our low-rank adapter construction for FNN approximation, which is detailed in the proof of Theorem 5.

Model. The setup and model generation here mirrors that of Sec. G.1, with two depths combination considered: $\overline{L} = 1, L = 2$, and $\overline{L} = 2, L = 4$. It should be noted that for both these cases, we have $M = \lfloor L/\overline{L} \rfloor = 2$ here.

Results. Figure 2 presents the results for FNN approximation. The y limit changes from Fig. 2(a) to Fig. 2(b) suggest that the pretrained frozen model results in less approximation error. Additionally, we observe that our construction's performance aligns closely with the gradient update method when the target model depth $\overline{L} = 1$. However, this alignment is not observed when $\overline{L} = 2$, We conjecture that the suboptimality of our construction in this multi-layer FNN approximation case may be due to an inherent constraint in our construction of LoRA adapters, which is to match the intermediate output of the frozen model with that of the target model. Additionally, the uniform partition could also be one contributing factor.



(*a*) Frozen model is randomly generated. (*b*) Frozen model is pretrained.

Figure 3: Approximation error (measured by MSE) versus LoRA-rank on TFNs.

G.3. TFN Approximation

We assess the effectiveness of our LoRA adapter construction in approximating TFN, as detailed in the proof of Theorem 6.

Setup. We examine target model $\overline{f}$ and frozen model f, both featuring the same architecture with L transformer blocks, a single output layer, two attention heads, and embedding size $D = 16$. We focus on two scenarios: $L = 1$ and $L = 2$. The weight matrices for the attention layers follow a standard Gaussian distribution, while those for the linear layers are initialized using the Xavier uniform distribution, which is PyTorch's default scheme for linear layer initialization.

Results. The observations here align with those from the experiments of FNN approximation. We note that the gradient update method outperforms our approach when the rank is relatively small but lags behind as the rank increases. This advantage of the gradient update method at minimal ranks arises from the inherent complexity of TFNs, which allows for more flexible low-rank adapter construction. Meanwhile, the gradient update method's performance does not significantly improve as the rank increases. This is because our analysis focuses on a simplified version of TFNs, excluding residual connections and layer normalization, making the model difficult to optimize. Nonetheless, our results corroborate the claims made in Theorem 6, as the approximation error must be eradicated when the rank reaches $\lceil \frac{D}{2} \rceil = 8$.

Appendix H. Extended Discussion and Future Work

To the best of our knowledge, this paper is the first to offer a theoretical understanding of LoRA fine-tuning on both FNN and TFN. Our work delivers insightful results, elucidating the impact of rank, depth of the pre-trained model, and the distance between the pre-trained model and the target model on the expressive power of LoRA. Despite these advancements, several intriguing questions still remain open. First, as observed in the numerical experiments, our construction of LoRA adapters for FNN and TFN may not be always optimal. Given that more complex models offer increased flexibility, an open question is whether we can devise a more parameter-efficient scheme to construct the LoRA adapters, thereby deriving a tighter bound on approximation error. Second, for TFN, we have only identified the conditions under which the LoRA-adapted model exactly matches the target model, due to the analytical complexity of TFN. It would be interesting to quantify the approximation error when the rank is lower than required. Furthermore, for TFN, we constrain the target model and the frozen model to have identical embedding size and depth, and we omit the skip connections and

layer norms for simplicity. Another intriguing direction would be to study the expressive power of LoRA under TFN cases with more general settings on TFN architectures.

Appendix I. Extension to Cases with Different Model Dimensions

This discussion only applies to linear model approximation and FNN approximation. As highlighted in Sec. D, our results can be easily extended to scenarios where the target model, $\overline{f}$, and the frozen model, f, have different model dimensions. Specifically, for linear model or FNN approximation, we use $\overline{D}$ to represent the number of hidden neurons per layer in the target model and D for the frozen model. We particularly consider the cases where the frozen model is wider than the target model, i.e., $D \geq \overline{D}$. This is because the frozen model is typically overparameterized in practical applications.

The key idea for extending our analysis to scenarios with different model dimensions is expanding the dimension of the target model. For the sake of simplicity, we focus on the simplest case, the linear model approximation, as an example. In this setting, the difference between the output of the adapted model and the target model can be measured by

$$f\left(\begin{bmatrix}\boldsymbol{x}\\ \boldsymbol{0}\end{bmatrix}\right) - \begin{bmatrix}\overline{f}(\boldsymbol{x})\\ \boldsymbol{0}\end{bmatrix} = \prod_{l=1}^{L}\left(\boldsymbol{W}_l + \Delta\boldsymbol{W}_l\right)\begin{bmatrix}\boldsymbol{x}\\ \boldsymbol{0}\end{bmatrix} - \begin{bmatrix}\overline{\boldsymbol{W}}\boldsymbol{x}\\ \boldsymbol{0}\end{bmatrix}, \tag{26}$$

where $\boldsymbol{x} \in \mathbb{R}^{\overline{D}}$. Consequently, the last $(D - \overline{D})$ columns and rows of $\prod_{l=1}^{L}\left(\boldsymbol{W}_l + \Delta\boldsymbol{W}_l\right)$ does not affect the results at all. Denote the submatrix consisting of the first d rows and d columns of a matrix $\boldsymbol{W}$ by $[\boldsymbol{W}]_d$. Then, to approximate the target model, we aim to solve the following constrained optimization problem for a given LoRA-rank $R \in [D]$:

$$\min_{\mathrm{rank}(\Delta\boldsymbol{W}_l)\leq R}\left\|\left[\prod_{l=1}^{L}\left(\boldsymbol{W}_l + \Delta\boldsymbol{W}_l\right)\right]_{\overline{D}} - \overline{\boldsymbol{W}}\right\|_{\mathrm{F}}.$$

To solve this problem, we first define an expanded target matrix, denoted by $\widetilde{\boldsymbol{W}} \in \mathbb{R}^{D\times D}$. The expanded target matrix $\widetilde{\boldsymbol{W}}$ is constructed such that $\left[\widetilde{\boldsymbol{W}}\right]_{\overline{D}} = \overline{\boldsymbol{W}}$, while the remaining entries matches the corresponding entries in $\prod l = 1^{L}\boldsymbol{W}_l$. Then, the error matrix $\boldsymbol{E} = \widetilde{\boldsymbol{W}} - \prod_{l=1}^{L}\boldsymbol{W}_l$, consists entirely of zeros except for the first $\overline{D}$ rows and $\overline{D}$ columns. Therefore, we obtain $R_{\boldsymbol{E}} = \mathrm{rank}(\boldsymbol{E}) \leq \overline{D}$.

Given the expanded target matrix, we consider the updated constrained optimization problem as follows:

$$\min_{\mathrm{rank}(\Delta\boldsymbol{W}_l)\leq R}\left\|\prod_{l=1}^{L}\left(\boldsymbol{W}_l + \Delta\boldsymbol{W}_l\right) - \widetilde{\boldsymbol{W}}\right\|_{\mathrm{F}}. \tag{27}$$

By Lemma 9, we obtain that when the LoRA-rank $R \geq \lfloor\frac{\overline{D}}{L}\rfloor$, the optimal solution to (27) satisfies $\prod_{l=1}^{L}\left(\boldsymbol{W}_l + \Delta\boldsymbol{W}_l\right) = \widetilde{\boldsymbol{W}}$, given that $\overline{D} \geq R_{\boldsymbol{E}}$. This result implies that $\left[\prod_{l=1}^{L}\left(\boldsymbol{W}_l + \Delta\boldsymbol{W}_l\right)\right]_{\overline{D}} = \overline{\boldsymbol{W}}$ and therefore the approximation error defined in (26) is 0 for all input $\boldsymbol{x}$.

A similar analysis can be conducted for FNN approximation.