

# **ResLoRA**

This is just the beginning of something big.

# Abstract

## Abstract

As one of the most popular parameter-efficient fine-tuning (PEFT) methods, low-rank adaptation (LoRA) is commonly applied to fine-tune large language models (LLMs). However, updating the weights of LoRA blocks effectively and expeditiously is challenging due to the long calculation path in the original model. To address this, we propose ResLoRA, an improved framework of LoRA. By adding residual paths during training and using merging approaches to eliminate these extra paths during inference, our method can achieve better results in fewer training steps without any extra trainable parameters or inference cost compared to LoRA. The experiments on NLG, NLU, and text-to-image tasks demonstrate the effectiveness of our method. To the best of our knowledge, ResLoRA is the first work that combines the residual path with LoRA. The code of our method is available at <https://github.com/microsoft/LMOpns/tree/main/reslora>.

**PEFT(파라미터 효율적 미세조정) 방법들 가운데 가장 널리 쓰이는 방법 중 하나로, 저랭크 적응(LoRA)은 대형 언어 모델(LLM)을 미세조정할 때 흔히 사용된다.**

**그러나 원래 모델에서의 계산 경로가 길기 때문에 LoRA 블록의 가중치를 효과적이고 빠르게 업데이트하는 일은 쉽지 않다.**

**이를 해결하기 위해 우리는 LoRA를 개선한 프레임워크인 ResLoRA를 제안한다.**

# Abstract

## Abstract

As one of the most popular parameter-efficient fine-tuning (PEFT) methods, low-rank adaptation (LoRA) is commonly applied to fine-tune large language models (LLMs). However, updating the weights of LoRA blocks effectively and expeditiously is challenging due to the long calculation path in the original model. To address this, we propose ResLoRA, an improved framework of LoRA. By adding residual paths during training and using merging approaches to eliminate these extra paths during inference, our method can achieve better results in fewer training steps without any extra trainable parameters or inference cost compared to LoRA. The experiments on NLG, NLU, and text-to-image tasks demonstrate the effectiveness of our method. To the best of our knowledge, ResLoRA is the first work that combines the residual path with LoRA. The code of our method is available at <https://github.com/microsoft/LMOpns/tree/main/reslora>.

. **ResLoRA는 학습 과정에서 잔차(residual) 경로를 추가하고, 추론 과정에서는 병합(merging) 기법을 통해 이러한 추가 경로를 제거함으로써, LoRA에 비해 추가로 학습해야 하는 파라미터나 추론 비용 없이도 더 적은 학습 단계에서 더 좋은 성능을 달성할 수 있다.**

**자연어 생성(NLG), 자연어 이해(NLU), 텍스트-이미지 생성 과제에서의 실험 결과는 본 방법의 효과를 보여준다.**

# Introduction

크특정 데이터셋에 맞춰 LLM을 미세조정(fine-tuning) 하면, 추론 단계에서 프롬프트로 지시만 주는 것보다 보통 더 좋은 성능을 얻는다(Xu et al., 2023). 그러나 이 과정은 관련된 파라미터 수가 너무 많기 때문에 비용이 종종 감당하기 어려울 정도로 큼

이 문제를 해결하기 위해 다양한 파라미터 효율적 미세조정(PEFT) 방법들이 제안됨. PEFT 방법은 원래 모델의 모든 파라미터를 고정(freeze)하고, 새로 추가한 모듈 안의 일부 파라미터만 조정함. 그중에서도 가장 널리 쓰이는 PEFT 방법 중 하나가 LoRA(Hu et al., 2022)로, 이는 low-rank adaptation의 약자.

LoRA는 학습 단계에서, 원래의 (고정된) 선형 층(frozen linear layer)과 병렬(parallel)로 놓인 두 개의 행렬을 소수의 학습 파라미터로 두고 업데이트하며, 추론 단계에서는 이들을 하나로 병합(merge) 한다. 병합 이후에는 LoRA가 시간 및 연산 비용을 추가로 유발하지 않으며, 수학적으로도 효과가 증명되었기 때문에(Zeng and Lee, 2023) 적용 범위가 매우 넓다.

# Introduction

그럼에도 기본 LoRA 방법에는 몇 가지 한계가 남아 있다. 기존 연구는 주로 (1) 모델의 서로 다른 층에서 LoRA 모듈의 랭크(rank)를 동적으로 조절하거나(Zhang et al., 2023a), (2) 더 적은 학습 파라미터로 원래 LoRA와 비슷한 효과를 내는 방법(Valipour et al., 2022)에 집중해 왔다. 하지만 이들은 한 가지 잠재적 문제를 간과했다. 즉, 역전파(backward) 경로가 길면 LoRA 블록의 파라미터 업데이트가 방해될 수 있다는 점이다

- 대표적인 방법인 ResNet(He et al., 2016a,b)은 매우 효율적인 것으로 널리 입증되었고, Transformer 모델에서도(Vaswani et al., 2017) 서로 다른 인코더/디코더 블록 사이에 사용된다. 이러한 블록들의 선형 층(linear)과 병렬로 붙는 LoRA 블록 역시, ResNet의 원래 쇼트컷(shortcut) 설계로부터 이점을 얻을 수 있다. 그러나 선형 층과 달리 LoRA 블록은 더 미세한 단위(fine-grained)이다. LoRA 블록 하나는 선형 층 하나에만 대응하므로, 기존 쇼트컷을 그대로 적용하기가 완벽하지 않다.

예를 들어 Transformer의 인코더를 보자. query/key/value 선형 층에 LoRA 블록을 추가하면, 이전 층의 그래디언트는 역전파를 계산할 때 출력 선형 층(output linear)과 Softmax 함수를 반드시 거쳐야 하며, 이 과정에서 그래디언트 소실 또는 폭발이 발생할 수 있다.

# Introduction

본 논문에서는 ResNet의 쇼트컷을 LoRA 블록에 결합한 새로운 프레임워크인 ResLoRA를 제안한다. 서로 다른 잔차(residual) 구조들의 효율을 검증하기 위해, 우리는 세 가지 잔차 구조를 제안하고 다양한 모델과 과제에서 실험을 수행한다.

하지만 이러한 쇼트컷은 비-평면(non-plain) 구조 때문에 원래 네트워크에 그대로 병합할 수 없고, 이는 LoRA의 장점을 약화시킨다. 이를 고려하여, 우리는 ResLoRA를 원래 LoRA 블록으로 변환하기 위한 다양한 병합(merging) 접근법을 논의하며, 추론 시에도 여전히 원래 모듈에 병합될 수 있도록 한다.

병합 이후에는 추가 파라미터도, 추가 계산 복잡도도 발생하지 않는다.

# Related works

**Parameter-efficient fine-tuning (PEFT)** Research on PEFT can be divided into three types. One line of research (Lester et al., 2021; Liu et al., 2023) is to add some special trainable vectors attached to the input sequence, which will increase the length of input and have a gap in results compared to full-finetune. Another line of research is to add serialized modules in original modules both in training and inference stage, called Adapter(Houlsby et al., 2019; Zhang et al., 2023b,b). In contrast, LoRA method (Hu et al., 2022) adds new low-rank matrices parallel to original linear layer in training stage and merges them into the original model during inference, so there is no extra cost when inferring.

## PEFT:

- 큰 모델(LLM)의 대부분 파라미터는 고정하고(freeze), 아주 일부만 학습해서 특정 작업에 맞게 적응시키는 미세조정 방법들의 총칭.

## 대표 유형

### 1. Prompt tuning / Prefix tuning류

입력 시퀀스에 학습 가능한 벡터("가짜 토큰")를 붙여서 모델을 유도

- 장점: 매우 가벼움
- 단점: 입력 길이 증가, full fine-tune만큼 성능이 안 나올 수 있음

### 2. Adapter류

각 Transformer 블록 안에 작은 모듈을 “직렬로” 끼워 넣고 그 부분만 학습

- 장점: 안정적, 표현력 좋음
- 단점: 추론 시에도 모듈이 남아 추론 비용 증가 가능

### 3. LoRA류(병렬 저랭크 업데이트)

원래 선형층 옆에 저랭크 업데이트를 “병렬”로 추가하고 학습 후 합쳐버림(merge)

- 장점: 추론 비용 0에 가깝게 유지 가능

# Related works

**Low-rank training method (LoRA)** Recent studies on LoRA aim to achieve lower cost and better performance. Some researchers explore more flexible and appropriate ranks, such as DyLoRA(Valipour et al., 2022), ReLoRA(Lialin et al., 2023), LoHA(Hyeon-Woo et al., 2021) and LoKr(Yeh et al., 2023). AdaLoRA(Zhang et al., 2023a) design a method to dynamically allocate the rank of LoRA blocks in different layers based on their importance, which can reduce the unimportant rank of LoRA blocks. Other works focus on the combination of LoRA and other approaches, such as AdaMix(Wang et al., 2022) and QLoRA(Dettmers et al., 2023). Besides, LoRAHub(Huang et al., 2023) and LoRAMoE(Anonymous, 2024) focus on how to merge multiple LoRA blocks that are fine-tuned on different tasks respectively. Despite this, no one has focused on the potential barrier of gradient propagation in LoRA.

## LoRA

- 선형층 가중치  $W_{\theta}$ 를 직접 업데이트하지 않고, 저랭크 행렬  $BABABA$ 로만 업데이트를 학습하는 방법.

A handwritten note on a grid background. At the top right, it says  $B \in \mathbb{R}^{d \times r}$  and  $A \in \mathbb{R}^{r \times k}$  with the condition  $\text{rank } A \ll \min(d, k)$ . Below this, the equation  $W_{\theta} + \Delta W = W_{\theta} + BA$  is written, with an arrow pointing from  $W_{\theta}$  to the text "Large Model". To the right of the equation, there is a bracket under  $BA$  with the text "원래 가중치". At the bottom left, it says  $B = 0$ , and at the bottom right, it says  $A = \bigcup$  with the text "가우시안 정규 분포에서 Sampling".

LoRA는 “학습 파라미터 수”는 줄이지만,

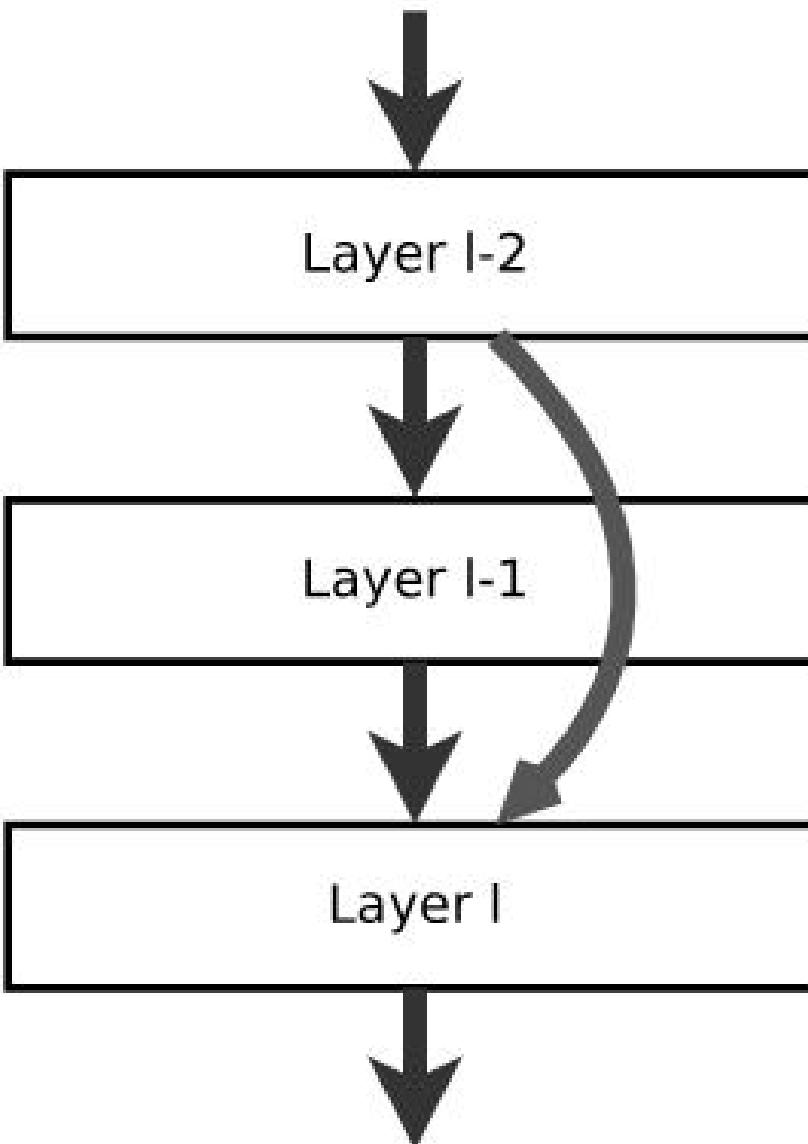
- 역전파 경로가 길어지거나(특히 attention 내부),
- 업데이트가 “잘 전달되는지”(gradient flow)가 층/구조에 따라 까다로울 수 있다는 문제 제기가 있음.

# Related works

**Residual Network(ResNet)** He et al. (2016a) and previous works(Srivastava et al., 2015) first introduced the residual network. This work solves the gradient vanishing or explosion and improves numerical stability during gradient updating. Considering that the extra shortcut path requires extra computational cost, some works(Ding et al., 2021) attempt to remove extra paths in the inference stage. Inspired by them, we first extend the main idea of ResNet to LoRA to achieve a faster and more stable training stage, and then design merging approaches to preserve the plain structure of LoRA, so as to

## 잔차 네트워크

- 어떤 층이 “변환”만 학습하게 만들고, 원래 입력은 그대로 더해주는 지름길(**Shortcut**)을 둬서 학습을 안정/가속하는 구조.



# Method

ResLoRA 프레임워크에서는 크게 두 가지 부분으로 나뉘어진다.

- (1) ResLoRA 블록: LoRA 블록 내부에 다양한 잔차(residual) 경로를 추가하여, 주로 학습 단계에서 사용된다.
- (2) 병합(merging) 접근법: 추가된 잔차 경로를 제거하여 ResLoRA 블록을 LoRA 블록으로 변환하며, 주로 추론 단계에서 사용된다.

# Method

먼저 LoRA 방법을 다시 정리한다. 사전학습 모델의 선형층(linear layer)에 있는 원래 가중치 행렬을  $W_n \in \mathbb{R}^{p \times q}$ 라 하자. 여기서  $p$ 와  $q$ 는 각각 출력과 입력 차원을 의미한다. 그러면 원래 선형층의 출력은

$$h_n = W_n x_n,$$

으로 쓸 수 있다. 여기서  $x_n$ 는 입력 벡터,  $h_n$ 는 출력 하든 벡터,  $n$ 은 층(layer)의 인덱스이다. 우리는 LoRA 블록을 원래 행렬과 병렬(parallel)로 추가되는 부가 블록으로 정의한다. LoRA 블록은 두 개의 새로운 행렬을 포함한다.

- 다운-프로젝션  $A \in \mathbb{R}^{r \times p}$
- 업-프로젝션  $B \in \mathbb{R}^{q \times r}$

이들은 고랭크 행렬 업데이트를 저랭크 행렬로 분해하기 위한 것이다. 학습 중에는  $W_n$ 은 고정(frozen)되고  $A$ 와  $B$ 의 가중치만 업데이트된다. 추론 시에는 추가 파라미터를  $W_n + B_n A_n W_n + B_n A_n$  형태로 원래 파라미터에 병합하여, 추론 지연(latency)이 발생하지 않도록 한다. 따라서 LoRA는 다음 식 (1)로 표현된다.

$$h_n = W_n x_n + B_n A_n x_n$$

# Method

ResNet에서 영감을 받아, 우리는 LoRA 블록에 잔차 경로를 도입한다. 구조와 메커니즘에 따라 영향이 다를 수 있으므로, 우리는 세 가지 유형의 블록을 설계하고 구현한다. 각각 입력-ショート컷 (input-shortcut, is), 블록-ショート컷(block-shortcut, bs), 중간-ショート컷(middle-shortcut, ms) 이다. 그림 2는 각 유형의 구체 구조를 보여준다.

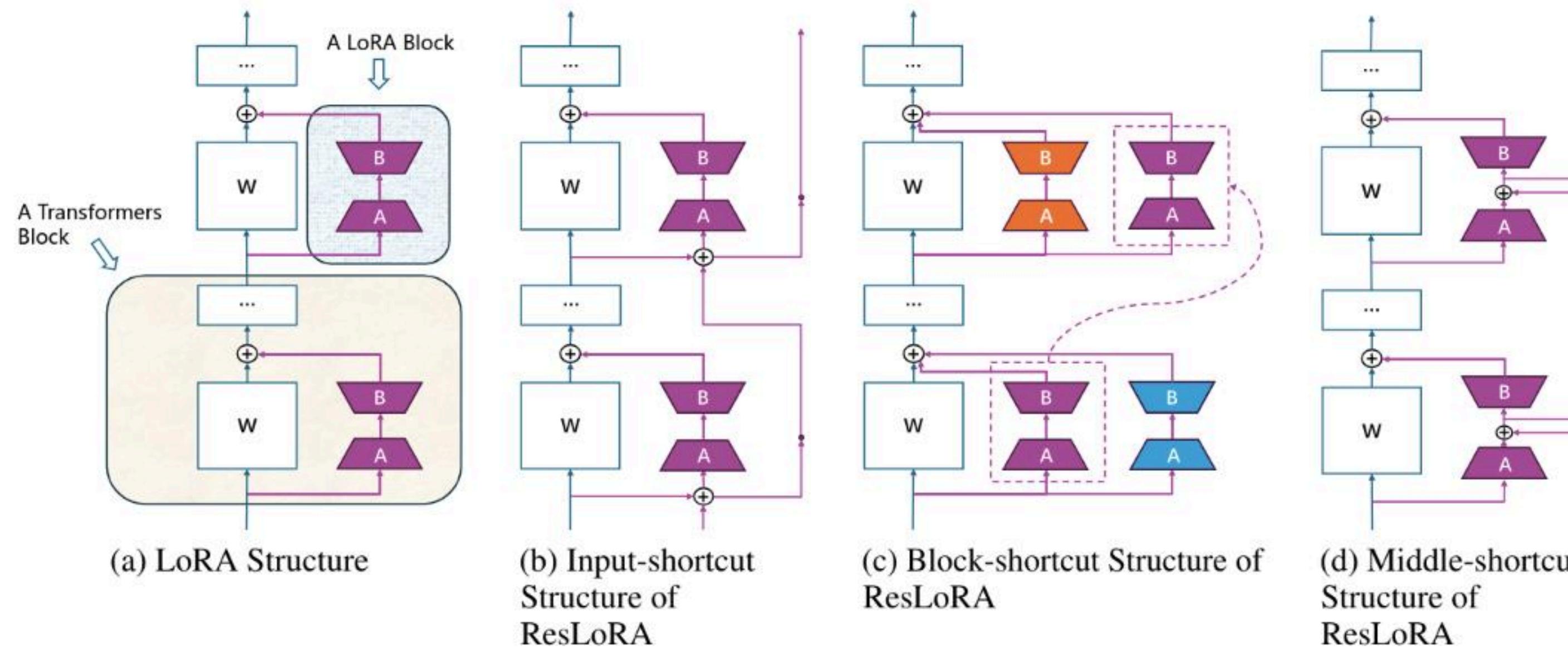


Figure 2: Structures of LoRA and ResLoRA

# Method

입력-ショート컷(input-shortcut) 구조는 서로 다른 LoRA 블록의 입력 벡터들 사이에 직접 쇼트컷을 둔다는 뜻이다. 구체적으로, 이전 LoRA 블록의 입력 벡터를 현재 LoRA 블록의 입력 벡터에 더한다. 이는 이전 입력을 현재 입력에 더하는 원래 ResNet 아이디어에서 착안했다.

다만 ResNet의 순전파처럼 LoRA 블록의 “입력”을 “출력”에 단순히 더하지는 않는다. 그렇게 하면 LoRA 블록의 순전파뿐 아니라 원래 선형층의 순전파까지 영향을 받아, 그래디언트를 계산하기 위한 순전파 단계에서 손실 값이 지나치게 커지고, 결과적으로 LoRA 블록을 학습시키는데 실패할 수 있기 때문이다. 이를 피하기 위해 우리는 LoRA 블록들 사이에서만 쇼트컷을 사용한다. 따라서 입력-ショート컷 타입 ResLoRA를 적용한 선형층 출력은 다음과 같다.

$$h_n = W_n x_n + B_n A_n (x_n + x_{n-1})$$

# Method

블록-ショート컷(block-shortcut) 구조는 쇼트컷을 입력 벡터가 아니라 LoRA 블록의 가중치(혹은 업데이트 항) 쪽에 두는 방식이다. 입력-ショート컷 구조는 잔차 경로 아이디어를 구현하지만, 추가적인 순전파 경로가 생겨서 ResLoRA를 원래 LoRA 블록으로 직접 변환할 수 없고, 별도의 병합 접근법이 필요하다. 이 과정에서 성능 손실이 생길 수 있다.

LoRA와 잔차 네트워크의 장점을 동시에 얻기 위해, 우리는 DenseNet(Huang et al., 2017)과 유사한 블록-ショート컷 구조를 설계한다. 현재 층의 입력 벡터  $x_n$ 에 대해, 현재 LoRA 블록뿐 아니라 여러 개의 이전 LoRA 블록을 동시에 사용해 계산에 참여시킨다. 이 방식은 입력에 대한 추가 순전파 경로를 만들지는 않지만, 역전파에서 중간 층을 “건너뛰는” 형태로 그래디언트를 더 직접 전달할 수 있게 해, 역전파 계산에서 발생할 수 있는 장애를 줄여준다. 이 구조의 출력은 다음과 같다.

$$h_n = W_n x_n + \left( \sum_{k=0}^m B_{n-k} A_{n-k} \right) x_n$$

# Method

중간-ショートカット(middle-shortcut) 구조는 LoRA 블록의 중간 결과(intermediate results)에 쇼트컷을 추가한다. LoRA 블록은 A와 B 두 행렬로 구성되며, A는 B보다 입력 벡터에 더 가깝다. Transformer 블록 사이에는 이미 쇼트컷이 존재하므로, A행렬은 이러한 쇼트컷의 이점을 더 쉽게 받을 가능성이 있고, 그래디언트 전파 문제를 덜 겪을 수 있다. 따라서 우리는 B행렬도 쇼트컷의 이점을 받게 하는 새로운 구조를 시도한다.

각 층에서 middle-shortcut은 A행렬의 수정 자체에 집중하기보다, B로 들어가기 직전의 중간 벡터에 쇼트컷을 준다. 즉, 이전 층들의 A 출력 벡터들을 현재 A 출력 벡터에 더한 뒤, 그 합을 현재 B에 입력으로 넣는다. 요약하면 과정은 다음과 같이 표현된다.

$$h_n = W_n x_n + B_n \left( \sum_{k=0}^m A_{n-k} x_{n-k} \right)$$

# Method-merging approaches

추가 쇼트컷은 ResLoRA에 이점을 주지만, 몇 가지 문제가 있다. 가장 중요한 문제 중 하나는 비-평면(non-plain) 구조가 만들어진다는 점이다. 우리가 평면(plain) 구조라고 부르는 원래 LoRA 블록은, 원래 층과 독립적인 추가 순전파 경로가 필요하지 않기 때문에 선형층에 바로 병합될 수 있다. 즉 LoRA 블록의 순전파는 원래 선형층의 순전파와 형태가 유사하다.

하지만 ResLoRA는 서로 다른 층의 ResLoRA 블록 사이에 추가 쇼트컷을 사용하므로, 원래 순전파 경로와 동일하지 않다. 따라서 ResLoRA 블록을 LoRA 블록으로 변환하기 위한 병합 접근법을 설계해야 한다.

# Method-merging approaches

그렇다면 어떻게 변환할 수 있을까? 블록-쇼트컷 구조의 경우, 현재 ResLoRA 블록은 이전 ResLoRA 블록들의 “가중치”만 필요로 하고 추가 순전파 경로가 없기 때문에, 다음과 같이 쉽게 병합할 수 있다.

$$W_n^* = W_n + \sum_{k=0}^m A_{n-k} B_{n-k}$$

그러나 나머지 두 구조(입력-쇼트컷, 중간-쇼트컷)는 현재 ResLoRA 블록이 이전 층의 입력 벡터를 필요로 하며, 이는 추가 순전파 경로를 만들어 원래 선형층과 달라지므로 직접 병합이 불가능하다.

$x_{n-1} = \alpha x_n$ , 처럼 표현할 수 있다면( $\alpha$ 는 스케일링 계수), 이전 입력 벡터를 현재 입력 벡터로 변환해 ResLoRA 블록을 LoRA 블록으로 바꿀 수 있다. 하지만  $x_{n-1}$ 가 달라지면 이를 만족하는  $\alpha$ 도 달라지므로 정확한  $\alpha$ 를 얻을 수 없다.

# Method-merging approaches

우리의 목표는  $x_{n-1} \approx \alpha^* x_n$ .

를 만족하는  $\alpha^*$ 를 찾는 것이다.

예를 들어 입력-쇼트컷 구조의 경우 다음과 같이 유도할 수 있다.

$$\begin{aligned} h_n &= W_n x_n + B_n A_n (x_n + x_{n-1}) \\ &\approx W_n x_n + B_n A_n (x_n + \alpha^* x_n) \\ &= W_n x_n + (1 + \alpha^*) B_n A_n x_n \end{aligned}$$

따라서 현재 선형층의 새로운 가중치는 다음과 같이 표현된다.

$$W_n^* = W_n + (1 + \alpha^*) B_n A_n$$

# Method-merging approaches

$\alpha^*$ 를 추정하는 두 가지 접근법이 존재하는데

$$\frac{x_n}{f_n} \approx \frac{x_{n-1}}{f_{n-1}}$$

Based on this, we can get  $\alpha^*$  to be:

$$\alpha^* = \frac{f_{n-1}}{f_n}$$

## (1) 입력 기반 병합(Merge Based on Input)

한 접근은  $x_n$ 과  $x_{n-1}$ 를 기반으로  $\alpha^*$ 를 직접 계산하는 것이다. 학습 단계에서 각 층마다 슬라이딩 윈도우를 사용해 최신 입력 벡터  $x_n$ 들을 수집한다.

이후 추론 단계에서 모든 입력 벡터의 Frobenius 노름을 계산하고, 각 슬라이딩 윈도우 내 Frobenius 노름의 평균을 구한다. 이때  $f_n$ 은  $n$ 번째 층에서의 Frobenius 노름 평균을 의미한다. 우리는 이 값이 입력 벡터의 크기를 나타낸다고 보고 다음 관계를 둔다.

# Method-merging approaches

layer after merging. Therefore, this relationship between  $x_n$  and  $x_{n-1}$  can be expressed as:

$$\frac{x_n}{x_{n-1}} \approx \frac{f_{n-1}^*}{f_{n-2}^*} \quad (10)$$

Based on this, we can get  $\alpha^*$  to be:

$$\alpha^* = \frac{f_{n-2}^*}{f_{n-1}^*} \quad (11)$$

For ResLoRA<sub>ms</sub>, we simply modify those merging approaches to adapt to the new structure. For merge based on input, we compute  $\alpha^*$  by:

$$\alpha^* = \sum_{k=1}^m \alpha_{n-k}^* \quad (12)$$

두 번째 접근법으로는, 입력 벡터 대신, 이전 ResLoRA 를  
록(또는 병합된 선형층)의 가중치 크기(Frobenius 노름)  
를 이용해

이전 입력  $x_{n-1}$ 과 현재 입력  $x_n$ 의 스케일 비율  
 $\alpha^*$ 을 근사한다.

직관적으로는  $h_{n-2}$ 가  $x_{n-1}$ 의 크기를 대표한  
다고 보고,

입력의 영향은 무시한 채 가중치의 크기만으로 스케일을  
추정한다.