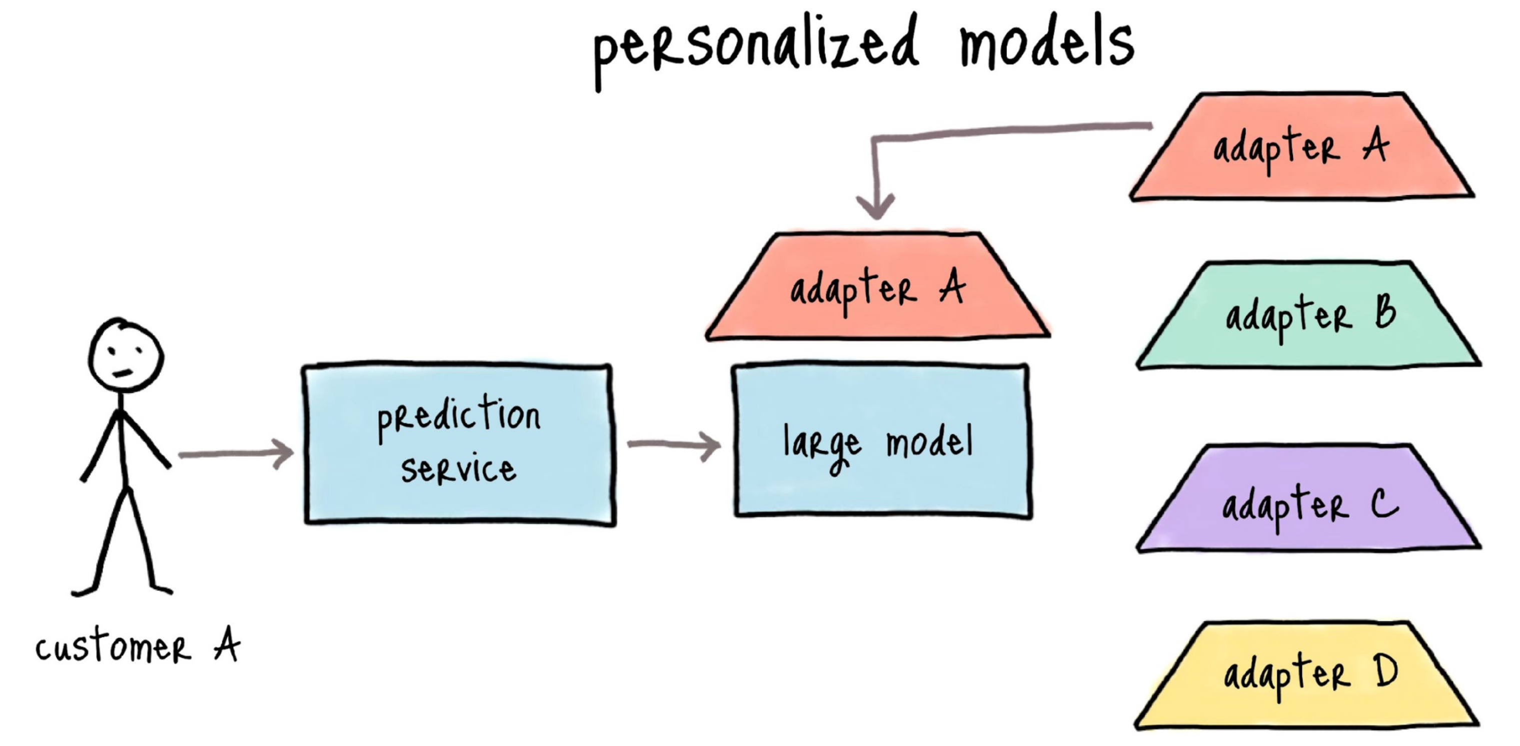
# LORA: LOW-RANK ADAPTATION OF LARGE LANGUAGE MODELS

Edward J. Hu and Yelong Shen and Phillip Wallis and Zeyuan Allen-Zhu and Yuanzhi Li and Shean Wang and Lu Wang and Weizhu Chen

Presented by,
Anagha M B, Gokul P V, and Chandu Dadi


personalized models
adapter A
adapter A
adapter B
prediction service
large model
adapter C
customer A
adapter D

# Challenges with Existing Adaptation Methods

**Adapter Layers:**

Small modules inserted into transformer layers.Their purpose is to modify the model's behavior for specific tasks without fine-tuning the entire network.

**Advantages:** Parameter-efficient; supports multi-task learning

**Problems:**

- **Inference Latency:** Additional depth increases sequential computations.
- **Scalability Issues:** Requires GPU synchronization (e.g., AllReduce) in large-scale deployments.

**Prompt Tuning:**

Modifies **input tokens** with task-specific prefixes.

**Example**:

Input sequence: "Translate English to French: What is your name?"

With prompt tuning: "[TASK_SPECIFIC] Translate English to French: What is your name?"

**Advantages:** Keeps model weights frozen; lightweight.

**Problems:**

- **Optimization Difficulty:** Training prompt embeddings is unstable.
- **Sequence Length Reduction:** Prefix tokens reduce space for task input.

# Eckart-Young Theorem

Given a matrix $A \in R^{m\times n}$ and its singular value decomposition:

$$A = U\Sigma V^T$$

where $\Sigma = \text{diag}(\sigma_1, \sigma_2, ..., \sigma_r)$, with $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_r > 0$ (singular values), the best rank-k approximation of A, in the Frobenius norm, is:

$$A_k = U_k \Sigma_k V_k^T$$

where $\Sigma_k$ is the diagonal matrix containing the top k singular values, and $U_k$, $V_k$ are the corresponding singular vectors.

# Key Ideas of LoRA

- Instead of updating all model parameters, LoRA injects trainable low-rank matrices into the pretrained weights.
- Pretrained weights remain frozen, reducing computational overhead.
- Parameter updates are represented as the product of two low-rank matrices.
- Gradients scaled by α/r

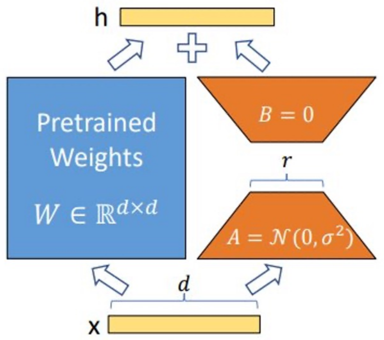



Figure 1: Our reparametrization. We only train $A$ and $B$.

$$h = W_0 x + \Delta W x = W_0 x + BAx$$

# LoRA Memory Efficiency

- The low-rank structure significantly reduces memory overhead and computational complexity during training.

  Storage: $d \times k \rightarrow r(d+k)$
  - Example: For $d=512$, $k=512$, $r=8$
  - Original: 262,144 parameters
  - LoRA: 8,192 parameters (96.9% reduction)

- Maintains expressiveness of full-rank updates with far fewer trainable parameters.

https://colab.research.google.com/drive/1DzArDRxgcPiu32fwWd3BgREy9SKRcOJ_

https://colab.research.google.com/drive/142Wy4B5cVUWnzS2JaSt9jPsyVcYTtB3O

# Applications of LoRA

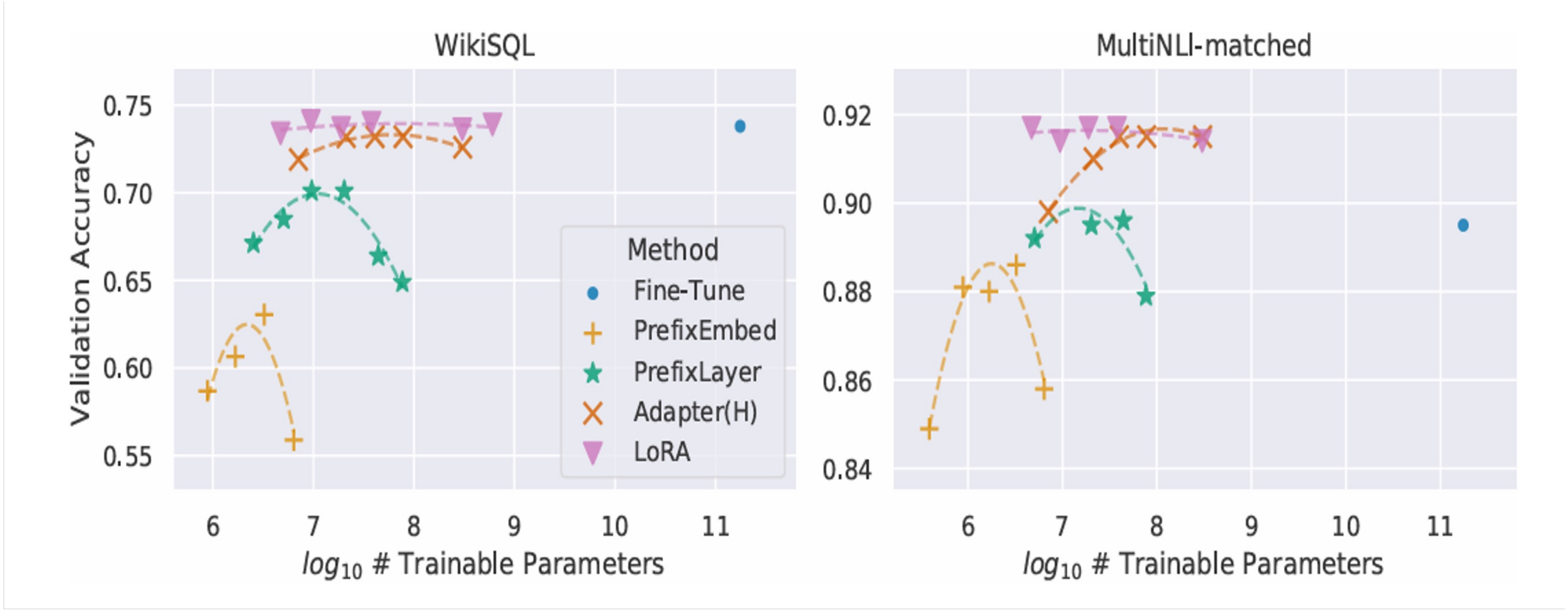
1. Natural Language Processing (NLP)
    a. **Text Classification**: Fine-tuning LLMs like RoBERTa, BERT, or GPT for sentiment analysis, spam detection, or topic classification with minimal parameter updates.
    b. **Question Answering (QA)**: Tuning models like RoBERTa or GPT for domain-specific Q&A applications, such as customer support or medical FAQs.
2. Personalized AI and Chatbots
3. Generative AI

# EMPIRICAL EXPERIMENTS

| Model & Method | # Trainable Parameters | MNLI | SST-2 | MRPC | CoLA | QNLI | QQP | RTE | STS-B | Avg. |
|---|---|---|---|---|---|---|---|---|---|---|
| $\text{RoB}_{\text{base}}$ (FT)* | 125.0M | **87.6** | 94.8 | 90.2 | **63.6** | 92.8 | **91.9** | 78.7 | 91.2 | 86.4 |
| $\text{RoB}_{\text{base}}$ (BitFit)* | 0.1M | 84.7 | 93.7 | **92.7** | 62.0 | 91.8 | 84.0 | 81.5 | 90.8 | 85.2 |
| $\text{RoB}_{\text{base}}$ ($\text{Adpt}^{\text{D}}$)* | 0.3M | $87.1_{\pm.0}$ | $94.2_{\pm.1}$ | $88.5_{\pm1.1}$ | $60.8_{\pm.4}$ | $93.1_{\pm.1}$ | $90.2_{\pm.0}$ | $71.5_{\pm2.7}$ | $89.7_{\pm.3}$ | 84.4 |
| $\text{RoB}_{\text{base}}$ ($\text{Adpt}^{\text{D}}$)* | 0.9M | $87.3_{\pm.1}$ | $94.7_{\pm.3}$ | $88.4_{\pm.1}$ | $62.6_{\pm.9}$ | $93.0_{\pm.2}$ | $90.6_{\pm.0}$ | $75.9_{\pm2.2}$ | $90.3_{\pm.1}$ | 85.4 |
| $\text{RoB}_{\text{base}}$ (LoRA) | 0.3M | $87.5_{\pm.3}$ | $\mathbf{95.1}_{\pm.2}$ | $89.7_{\pm.7}$ | $63.4_{\pm1.2}$ | $\mathbf{93.3}_{\pm.3}$ | $90.8_{\pm.1}$ | $\mathbf{86.6}_{\pm.7}$ | $\mathbf{91.5}_{\pm.2}$ | **87.2** |
| $\text{RoB}_{\text{large}}$ (FT)* | 355.0M | 90.2 | **96.4** | **90.9** | 68.0 | 94.7 | **92.2** | 86.6 | 92.4 | 88.9 |
| $\text{RoB}_{\text{large}}$ (LoRA) | 0.8M | $\mathbf{90.6}_{\pm.2}$ | $96.2_{\pm.5}$ | $\mathbf{90.9}_{\pm1.2}$ | $\mathbf{68.2}_{\pm1.9}$ | $\mathbf{94.9}_{\pm.3}$ | $91.6_{\pm.1}$ | $\mathbf{87.4}_{\pm2.5}$ | $\mathbf{92.6}_{\pm.2}$ | **89.0** |
| $\text{RoB}_{\text{large}}$ ($\text{Adpt}^{\text{P}}$)† | 3.0M | $90.2_{\pm.3}$ | $96.1_{\pm.3}$ | $90.2_{\pm.7}$ | $\mathbf{68.3}_{\pm1.0}$ | $\mathbf{94.8}_{\pm.2}$ | $\mathbf{91.9}_{\pm.1}$ | $83.8_{\pm2.9}$ | $92.1_{\pm.7}$ | 88.4 |
| $\text{RoB}_{\text{large}}$ ($\text{Adpt}^{\text{P}}$)† | 0.8M | $\mathbf{90.5}_{\pm.3}$ | $\mathbf{96.6}_{\pm.2}$ | $89.7_{\pm1.2}$ | $67.8_{\pm2.5}$ | $\mathbf{94.8}_{\pm.3}$ | $91.7_{\pm.2}$ | $80.1_{\pm2.9}$ | $91.9_{\pm.4}$ | 87.9 |
| $\text{RoB}_{\text{large}}$ ($\text{Adpt}^{\text{H}}$)† | 6.0M | $89.9_{\pm.5}$ | $96.2_{\pm.3}$ | $88.7_{\pm2.9}$ | $66.5_{\pm4.4}$ | $94.7_{\pm.2}$ | $92.1_{\pm.1}$ | $83.4_{\pm1.1}$ | $91.0_{\pm1.7}$ | 87.8 |
| $\text{RoB}_{\text{large}}$ ($\text{Adpt}^{\text{H}}$)† | 0.8M | $90.3_{\pm.3}$ | $96.3_{\pm.5}$ | $87.7_{\pm1.7}$ | $66.3_{\pm2.0}$ | $94.7_{\pm.2}$ | $91.5_{\pm.1}$ | $72.9_{\pm2.9}$ | $91.5_{\pm.5}$ | 86.4 |
| $\text{RoB}_{\text{large}}$ (LoRA)† | 0.8M | $\mathbf{90.6}_{\pm.2}$ | $96.2_{\pm.5}$ | $\mathbf{90.2}_{\pm1.0}$ | $68.2_{\pm1.9}$ | $\mathbf{94.8}_{\pm.3}$ | $91.6_{\pm.2}$ | $\mathbf{85.2}_{\pm1.1}$ | $\mathbf{92.3}_{\pm.5}$ | **88.6** |
| $\text{DeB}_{\text{XXL}}$ (FT)* | 1500.0M | 91.8 | **97.2** | 92.0 | 72.0 | **96.0** | 92.7 | 93.9 | 92.9 | 91.1 |
| $\text{DeB}_{\text{XXL}}$ (LoRA) | 4.7M | $\mathbf{91.9}_{\pm.2}$ | $96.9_{\pm.2}$ | $\mathbf{92.6}_{\pm.6}$ | $\mathbf{72.4}_{\pm1.1}$ | $\mathbf{96.0}_{\pm.1}$ | $\mathbf{92.9}_{\pm.1}$ | $\mathbf{94.9}_{\pm.4}$ | $\mathbf{93.0}_{\pm.2}$ | **91.3** |

# EMPIRICAL EXPERIMENTS

GPT-3 175 B



# CONCLUSION

Fine-tuning enormous language models is expensive in terms of the hardware required and the storage/switching cost for hosting independent instances for different tasks.

LoRA, an efficient adaptation strategy that neither introduces inference latency nor reduces input sequence length while retaining high model quality.

# References


- https://arxiv.org/pdf/2106.09685
- https://magazine.sebastianraschka.com/p/practical-tips-for-finetuning-llms
- https://towardsdatascience.com/understanding-lora-low-rank-adaptation-for-finetuning-large-models-936bce1a07c6
- https://github.com/huggingface/smol-course

# Thank you!