# LLM fine tuning

Presented by

Amir Shariatmadari, Guangzhi Xiong, Sabit Ahmed, Shiyu Feng

University of Virginia | ENGINEERING Department of Computer Science

# Overview

- Instruction Tuning for Large Language Models: A Survey
- Delta tuning: A comprehensive study of parameter efficient methods for pre-trained language models
- DoRA: Weight-Decomposed Low-Rank Adaptation
- Recent Large Language Models Reshaping the Open-Source Arena

# Instruction Tuning for Large Language Models: A Survey

presented by Shiyu Feng (eus5fy)

University of Virginia | ENGINEERING
Department of Computer Science

# Overview

Instruction tuning refers to the process of further training LLMs on a dataset consisting of pairs.

- Methodology of IT
- Construction of Instruction Tuning Datasets
- Instruction Tuned Models
- Multi-Modality Instruction Finetuning
- Applications in Different Domains
- Efficient Tuning Techniques
- Conclusion

https://github.com/xiaoya-li/Instruction-Tuning-Survey

University of Virginia | ENGINEERING Department of Computer Science

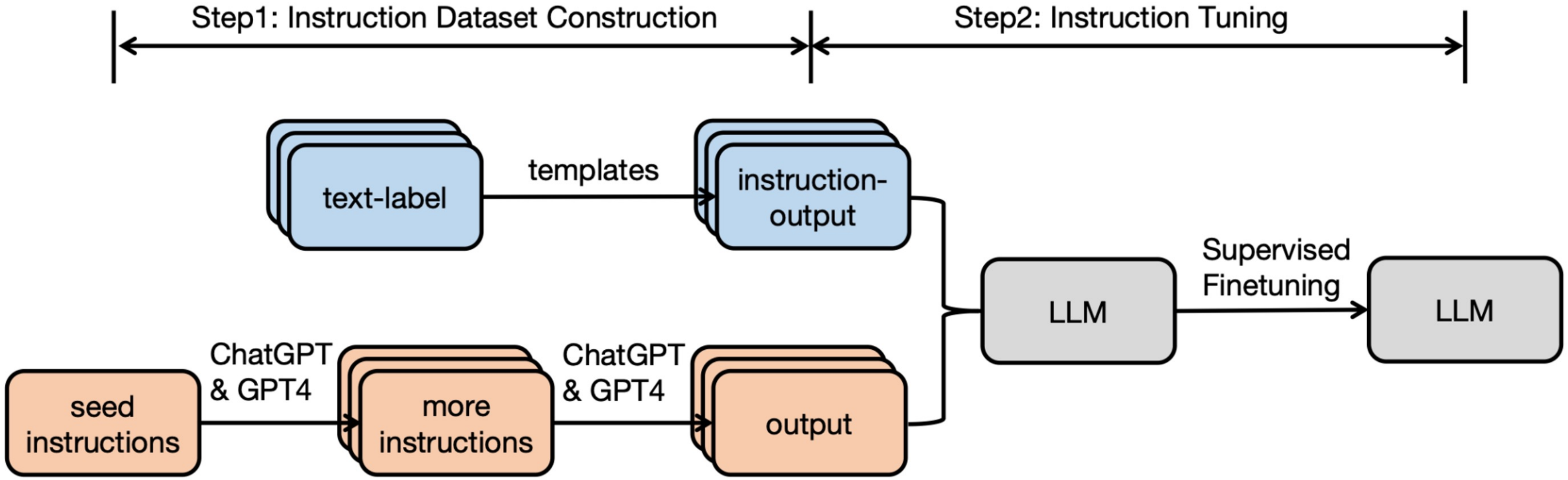
# Example form Instruction tuning Dataset

| | |
|---|---|
| Task Type | Grammar Error Correction |
| Task ID | task1557_jfleg_grammar_error_correction |
| Definition | In this task, you will be shown an incorrect English sentence. You need to generate a corrected form of the input sentence. |
| Positive Example | **Input**: The car's wheel are loose.<br>**Output**: The car's wheel is loose.<br>**Explanation**: The instance of are is replaced by the word is. This makes the sentence grammatically correct. |
| Negative Example | **Input**: This way is the way to go.<br>**Output**: This way may be the way to go.<br>**Explanation**: The example does not correct the misuse of the word way. Instead, it should shorten the sentence to: this is the way to go. |
| Instance | **Input**: I think it 's harder for successful preson to risk somethnig , thay coluld lost much more then others .<br>**Valid Output**: ["I think it 's harder for a successful person to risk something becuase they could lose much more than others ."] |

instruction

# General pipeline of instruction tuning

- instruction + (optional input) + anticipated output
  - Data integration from **annotated natural language datasets**
  - Generating outputs using **LLMs**



# Instruction Tuning Datasets

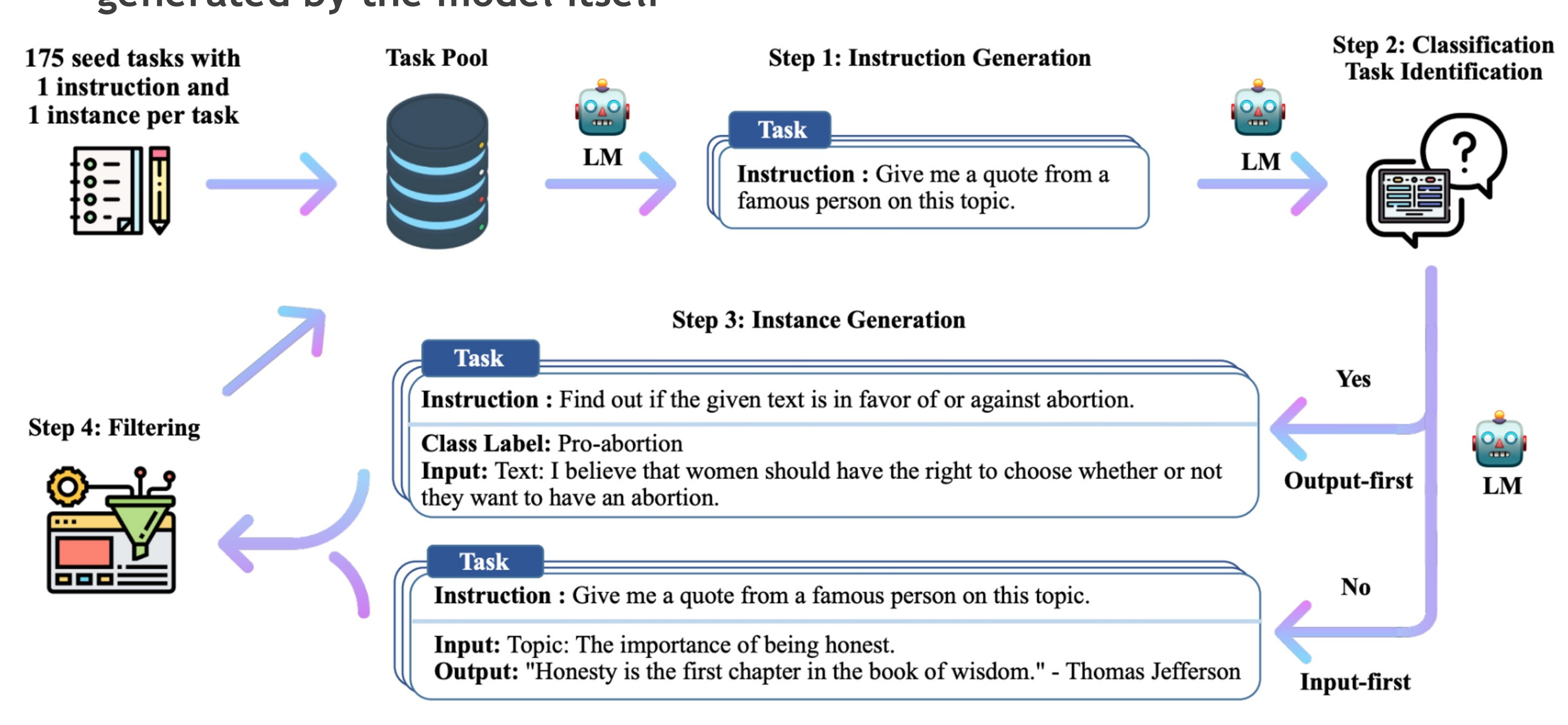
| Type | Dataset Name | # of Instances | # of Lang | Construction | Open-source |
|---|---|---|---|---|---|
| Human-Crafted | UnifiedQA (Khashabi et al., 2020)[1] | 750K | En | human-crafted | Yes |
| | UnifiedSKG (Xie et al., 2022)[3] | 0.8M | En | human-crafted | Yes |
| | Natural Instructions (Honovich et al., 2022)[4] | 193K | En | human-crafted | Yes |
| | Super-Natural Instructions (Wang et al., 2022f)[5] | 5M | 55 Lang | human-crafted | Yes |
| | P3 (Sanh et al., 2021)[6] | 12M | En | human-crafted | Yes |
| | xP3 (Muennighoff et al., 2022)[7] | 81M | 46 Lang | human-crafted | Yes |
| | Flan 2021 (Longpre et al., 2023)[8] | 4.4M | En | human-crafted | Yes |
| | COIG (Zhang et al., 2023a)[9] | - | - | - | Yes |
| | InstructGPT (Ouyang et al., 2022) | 13K | Multi | human-crafted | No |
| | Dolly (Conover et al., 2023a)[16] | 15K | En | human-crafted | Yes |
| | LIMA (Zhou et al., 2023)[18] | 1K | En | human-crafted | Yes |
| | ChatGPT (OpenAI, 2022) | - | Multi | human-crafted | No |
| | OpenAssistant (Köpf et al., 2023)[20] | 161,443 | Multi | human-crafted | Yes |
| Synthetic Data (Distillation) | OIG (LAION.ai, 2023)[2] | 43M | En | ChatGPT (No technique reports) | Yes |
| | Unnatural Instructions (Honovich et al., 2022)[10] | 240K | En | InstructGPT-Generated | Yes |
| | InstructWild (Xue et al., 2023)[12] | 104K | - | ChatGPT-Generated | Yes |
| | Evol-Instruct / WizardLM (Xu et al., 2023a)[13] | 52K | En | ChatGPT-generated | Yes |
| | Alpaca (Taori et al., 2023a)[14] | 52K | En | InstructGPT-generated | Yes |
| | LogiCoT (Liu et al., 2023a)[15] | - | En | GPT-4-Generated | Yes |
| | GPT-4-LLM (Peng et al., 2023)[17] | 52K | En&Zh | GPT-4-Generated | Yes |
| | Vicuna (Chiang et al., 2023) | 70K | En | Real User-ChatGPT Conversations | No |
| | Baize v1 (Conover et al., 2023b)[21] | 111.5K | En | ChatGPT-Generated | Yes |
| | UltraChat (Ding et al., 2023a)[22] | 675K | En&Zh | GPT 3/4-Generated | Yes |
| | Guanaco (JosephusCheung, 2021)[19] | 534,530 | Multi | GPT (Unknown Version)-Generated | Yes |
| | Orca (Mukherjee et al., 2023)[23] | 1.5M | En | GPT 3.5/4-Generated | Yes |
| | ShareGPT[24] | 90K | Multi | Real User-ChatGPT Conversations | Yes |
| | WildChat[25] | 150K | Multi | Real User-ChatGPT Conversations | Yes |
| | WizardCoder (Luo et al., 2023) | - | Code | LLaMa 2-Generated | No |
| | Magicoder (Wei et al., 2023b)[26] | 75K/110K | Code | GPT-3.5-Generated | Yes |
| | WaveCoder (Yu et al., 2023) | - | Code | GPT 4-Generated | No |
| | Phi-1 (Gunasekar et al., 2023)[27] | 6B Tokens | Code Q and A | GPT-3.5-Generated | Yes |
| | Phi-1.5 (Li et al., 2023h) | - | Code Q and A | GPT-3.5-Generated | No |
| | Nectar (Zhu et al., 2023a)[28] | 183K | En | GPT 4-Generated | Yes |
| Synthetic Data (Self-Improvement) | Self-Instruct (Wang et al., 2022c)[11] | 52K | En | InstructGPT-Generated | Yes |
| | Instruction Backtranslation (Li et al., 2023g) | 502K | En | LLaMa-Generated | No |
| | SPIN (Chen et al., 2024b)[29] | 49.8K | En | Zephyr-Generated | Yes |

- Human-Crafted
  - manually annotated or sourced from the internet
  - manual gathering and verification
  - costly
  - limited diversity
  - lack creativity (for novel task) and expertise (for writing solutions)

- Synthetic Data
  - pre-trained models
  - faster and more cost-effective
  - high quality and variety

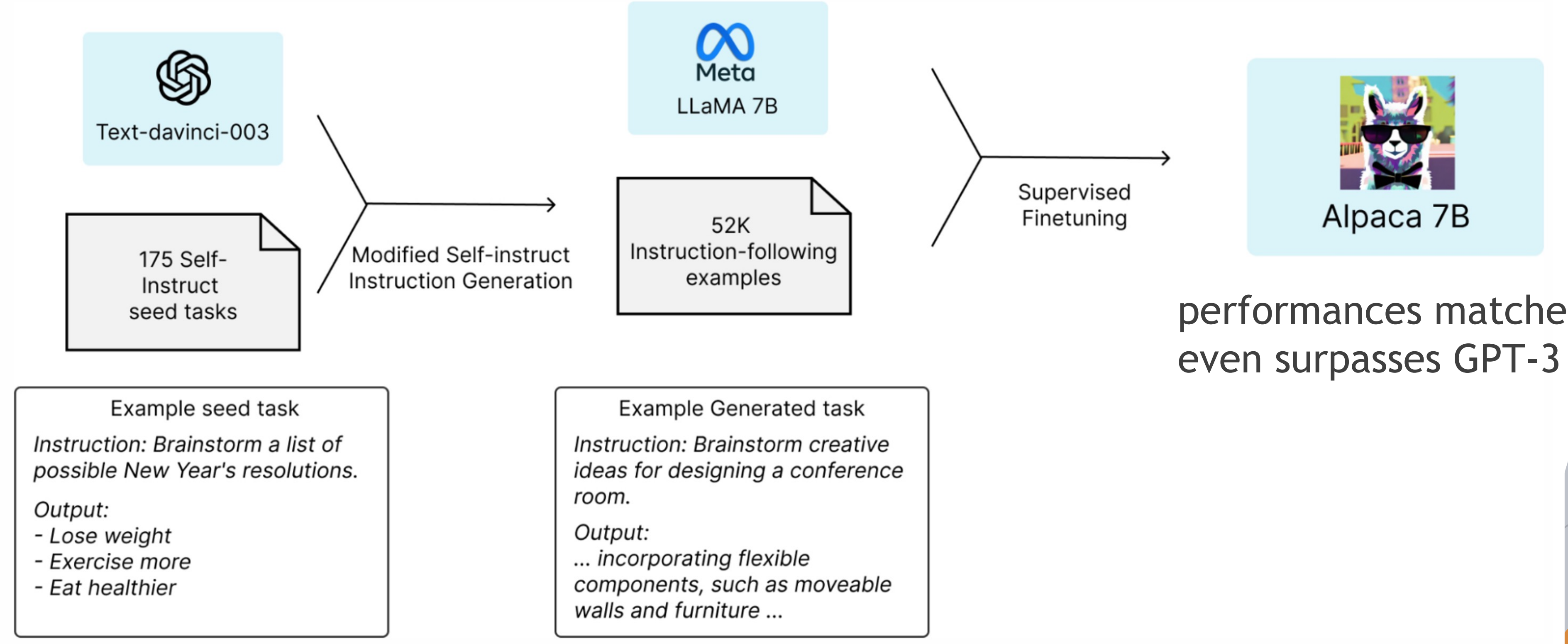
# Pipeline of Synthetic Data generation

- Semi-automated process for instruction tuning a pretrained LM using instructions **generated by the model itself**



# Instruction tuned model

- gather queries from fine-tuned LLMs —> use these queries as a basis to fine-tune subsequent LLMs
- impart knowledge from a highly capable teacher model to a less complex, more computationally efficient student model




performances matches or even surpasses GPT-3

# Multi-modality instruction fine-tuned LLMs

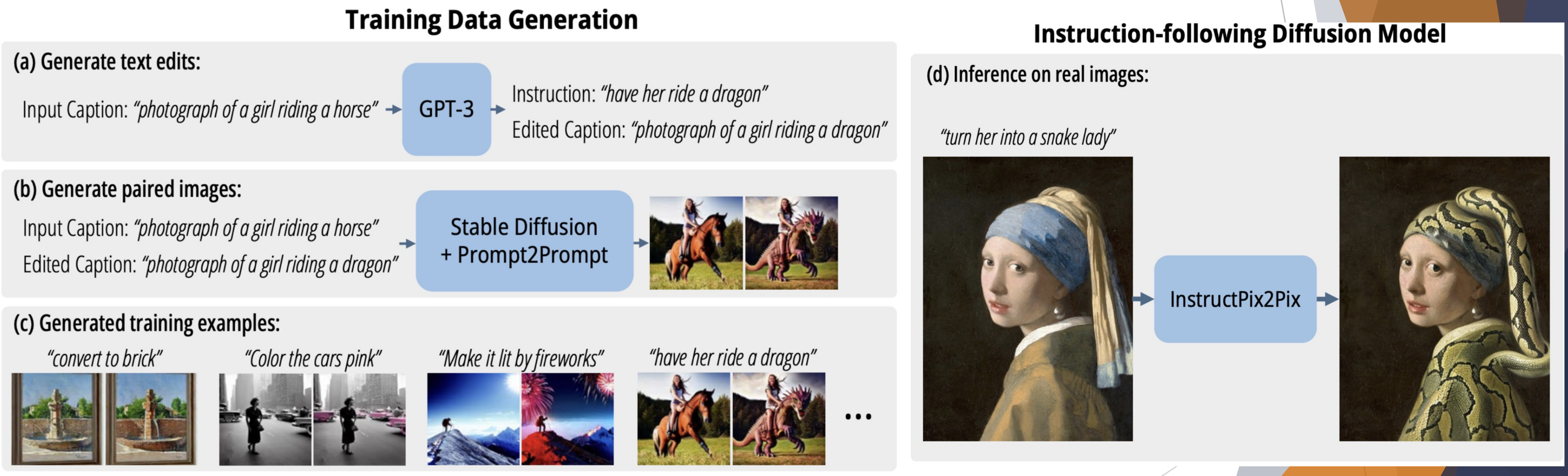
- Instruction tuning has expanded to multi-modal tasks, with datasets combining instructions with images, text, video or audio inputs and outputs.

| Multi-modality Instruction Fine-tuned LLMs | # Params | Modality | Base Model: Model Name | Base Model: # Params | Fine-tuning Trainset: Self-build | Fine-tuning Trainset: Size |
|---|---|---|---|---|---|---|
| InstructPix2Pix (Brooks et al., 2022)[1] | 983M | I/T | Stable Diffusion | 983M | Yes | 450K |
| LLaVA (Liu et al., 2023b)[2] | 13B | I/T | CLIP (Radford et al., 2021)<br>LLaMA (Touvron et al., 2023a)<br>LLaMA (Touvron et al., 2023a) | 400M<br>7B<br>7B | Yes | 158K |
| Video-LLaMA (Zhang et al., 2023b)[3] | - | I/T/V/A | BLIP-2 (Li et al., 2023d)<br>ImageBind (Girdhar et al., 2023)<br>Vicuna (Chiang et al., 2023) | -<br>-<br>7B/13B | No | - |
| InstructBLIP (1.2B) (Dai et al., 2023)[4] | - | I/T/V | BLIP-2 (Li et al., 2023d) | - | No | - |
| Otter (Li et al., 2023b)[5] | - | I/T/V | OpenFlamingo (Awadalla et al., 2023) | 9B | Yes | 2.8M |
| MultiModal-GPT (Gong et al., 2023)[6] | - | I/T/V | OpenFlamingo (Awadalla et al., 2023) | 9B | No | - |

# Image Editing: InstructPix2Pix

generate an image editing dataset, and train a diffusion model on that dataset

finetuned GPT-3, text-to-image model with prompt-to-prompt method (make the generations similar)



# Domain-specific instruction fine-tuned LLMs

- These applications highlight the adaptability and performance improvements achieved through instruction tuning

| Domain Type | Domain-specific Instruction Fine-tuned LLMs | Base Model: Model Name | Base Model: # Params | Trainset Size |
|---|---|---|---|---|
| Dialogue | InstructDial (Gupta et al., 2022)[1] | T0 (Sanh et al., 2021) | 3B | - |
| Classification | LINGUIST (Rosenbaum et al., 2022) | AlexaTM (Soltan et al., 2022) | 5B | 13K |
| Information extraction | InstructUIE (Wang et al., 2023b)[2] | FlanT5 (Chung et al., 2022) | 11B | 1.0M |
| Sentiment analysis | IT-MTL (Varia et al., 2022)[3] | T5 (Raffel et al., 2019) | 220M | - |
| Writing | Writing-Alpaca-7B (Zhang et al., 2023d)[4] | LLaMA (Touvron et al., 2023a) | 7B | - |
| | CoEdIT (Raheja et al., 2023)[5] | FlanT5 (Chung et al., 2022) | 11B | |
| | CoPoet (Chakrabarty et al., 2022)[6] | T5 (Raffel et al., 2019) | 11B | |
| Medical | Radiology-GPT (Liu et al., 2023c)[7] | Alpaca (Taori et al., 2023a) | 7B | 122K |
| | ChatDoctor (Li et al., 2023i)[8] | LLaMA (Touvron et al., 2023a) | 7B | 100K |
| | ChatGLM-Med (Haochun Wang, 2023)[9] | ChatGLM (Du et al., 2022) | 6B | - |
| Arithmetic | Goat (Liu and Low, 2023)[10] | LLaMA (Touvron et al., 2023a) | 7B | 1.0M |
| Code | WizardCoder (Luo et al., 2023)[11] | StarCoder (Li et al., 2023f) | 15B | 78K |

# Efficient Tuning Techniques for LLMs

optimize LLMs for downstream tasks by adjusting a small fraction of parameters, which makes LLM fine-tuning more effective and scalable for various applications

- HINT (Hypernetwork-based Instruction Tuning)
  - Incorporates long instructions and additional few-shots without increasing compute
- LOMO (LOw-Memory Optimization)
  - Enables full parameter fine-tuning of LLMs using limited computational resources
- Delta-tuning
  - Applies optimal control principles to guide model behavior on downstream tasks
- LoRA (Low-Rank Adaptation)
  - Reduces the number of trainable parameters and memory usage
- Qlora (Quantization and Memory Optimization for LLMs Fine-Tuning)
  - Enables training LLMs on limited computational resources with no degradation

# Conclusion

- Benefits:
  - aligns LLMs' next-word prediction with user instruction objectives
  - more controlled and predictable model behavior
  - rapid adaptation to specific domains without extensive retraining
  - IT models perform well with minimal training data

- Limitations:
  - datasets may lack quantity, diversity, and comprehensive evaluation methodologies
  - IT models may focus on surface-level patterns rather than understanding underlying tasks
  - Models imitating proprietary styles may lack generalization without diverse instruction datasets, emphasizing the need for improving base model quality and instruction diversity

UNIVERSITY of VIRGINIA | ENGINEERING Department of Computer Science

# Delta Tuning: A Comprehensive Study of Parameter Efficient Methods for Pre-trained Language Models

Ding et. al, 2022

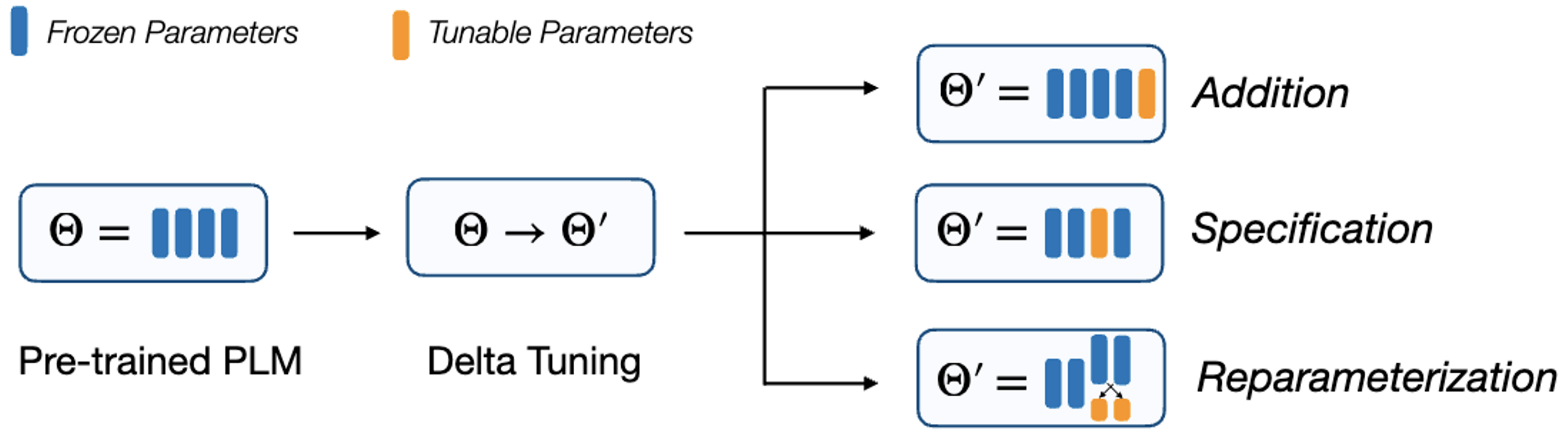
Tsinghua University

University of Virginia | ENGINEERING
Department of Computer Science

# Motivation:

- Fine-tuning pretrained language models (PLMs) models to specific tasks or domains have shown pretty impressive results in many downstream tasks.
- There is a huge computational toll when fine-tuning all the parameters of a PLM, which makes it impractical to do in many circumstances.
- This problem has led to a branch of research dedicated to adapting PLMs to specific tasks in a parameter efficient manner.
- The authors coin the new term “Delta Tuning”.
- These adaptive methods all essentially learn a set of adaptive or ‘delta’ parameters in the adaptation phase of learning.

# What are the different types of delta-tuning?




UNIVERSITY of VIRGINIA | ENGINEERING Department of Computer Science

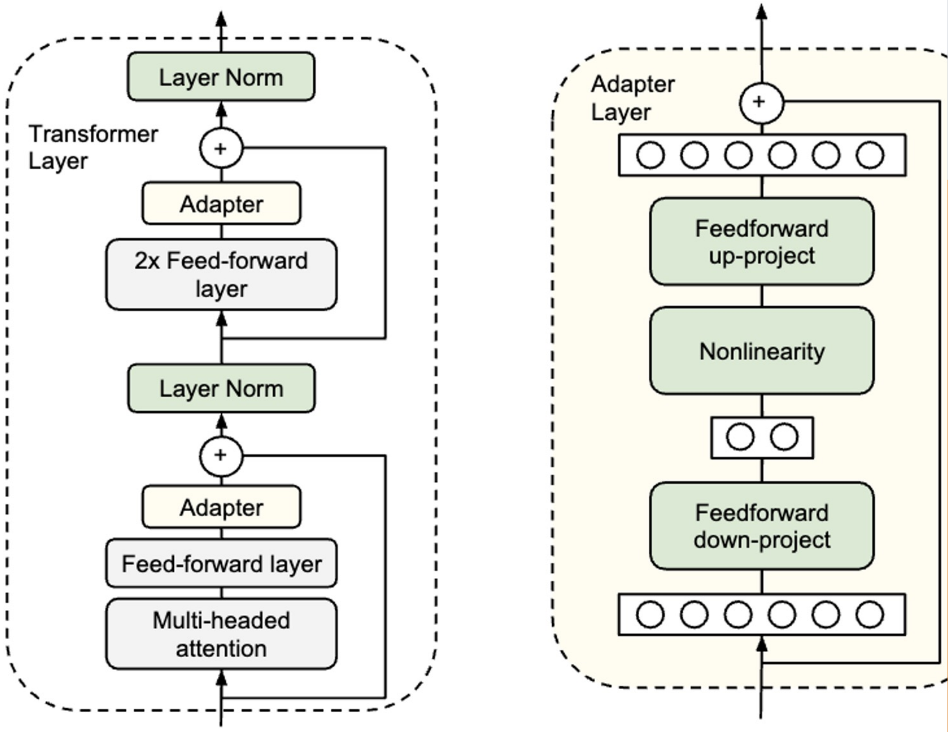
# Addition-based delta-tuning

- Introduce M additional parameters that don’t exist to the original model.

$$M \geq N \text{ and } \Delta\Theta = \{w_{N+1}, w_{N+2}, \ldots, w_M\}$$

# Addition-based delta-tuning :Adapter-based tuning

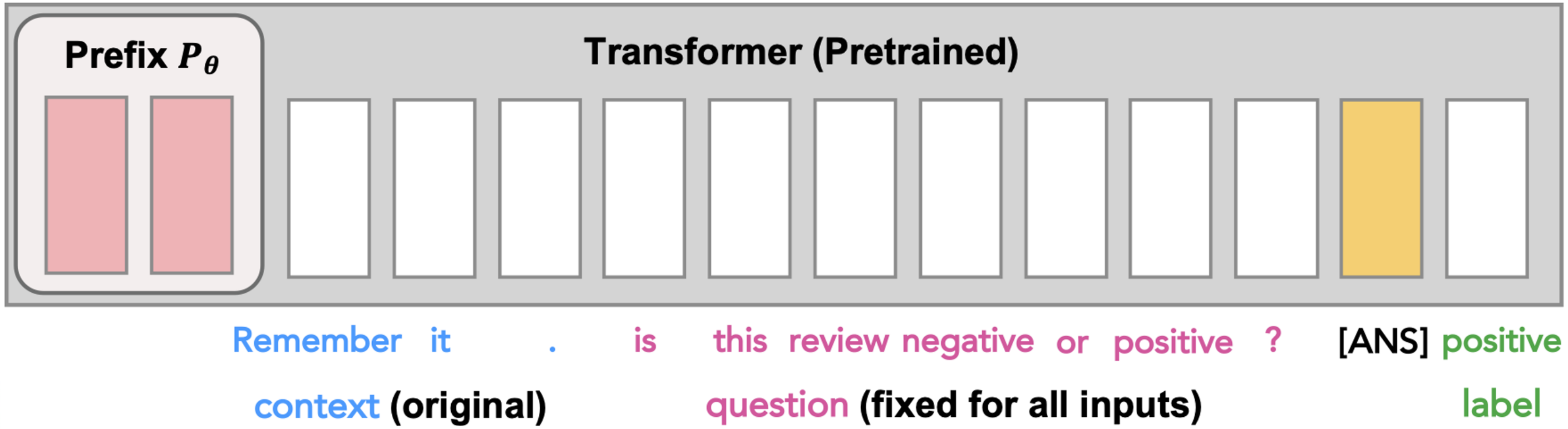
- This method involves adding neural modules called adapters to certain parts of the PLM.
- Adapters usually contain down-projection and up-projection components.
- Residual connection is added to the end of the up projection to preserve the original information and promote learning stability.
- Using adapters, often only about 0.5% to 8% of the total model parameters need tuning.
- Adapter-based tuning is advantageous in multi-task learning settings




(Houlsby et al.,2019)

# Addition-based delta-tuning: Prompt-tuning

- Prompt-based tuning doesn't involve modifying the internal structure of Transformer models but instead, it involves wrapping the input with additional context, known as prompts, to guide the model's output.
- These prompts are essentially continuous tokens or sequences that are added to the input to mimic pre-trained objectives, which helps in leveraging the knowledge captured during the model's initial extensive pre-training on vast amounts of data.



# Specification-based delta-tuning

- These methods specify some parameters of the original model to be frozen, while others should remain trainable.

$$\Delta\Theta = \{\Delta w_1, \Delta w_2, \ldots, \Delta w_N\}.$$

- "When $w_i \in W$, $\Delta w_i$ is the incremental value from $w_i$ to $w'_i$, else, $\Delta w_i = 0$"

# Specification-based delta-tuning:

- In heuristic specification, certain parameters are directly specified for optimization based on simple yet effective strategies.
  - Only fine-tuning one-fourth of the final layers of BERT about 90% of the performance compared to full parameter fine-tuning. Or Just optimizing bias terms, while keeping other parameters frozen, could still yield over 95% performance on several benchmarks.
- Other methods using algorithms to identify and optimize a selective set of parameters:
  - Diff pruning: Fine-tuning, but the number of parameters changed is minimized with the L0 Norm.
  - Masking method: learning selective masks that determine which weights of the model should be updated for specific tasks.

# Reparameterization-based delta-tuning

- Goal is to reparameterize original weights W to a more efficient form via some transformation function:

$$R(w_i) = \{u_1, u_2, ..., u_{N_i}\}$$

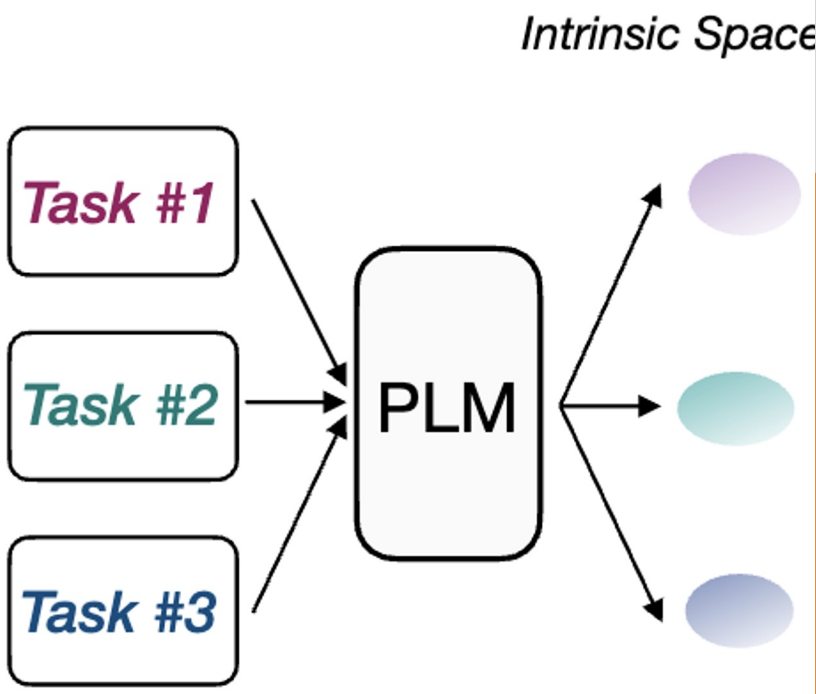
Union of new reparameterized weights

$$\Delta\Theta = (\Theta \setminus \mathcal{W}) \cup \mathcal{U}, \text{ where } \mathcal{U} = \{u_j | \exists w_i \in \mathcal{W}, u_j \in R(w_i)\}.$$

Non-reparameratized weights

# Reparameterization-based delta-tuning: Intrinsic Dimensions of PLM Adaptation

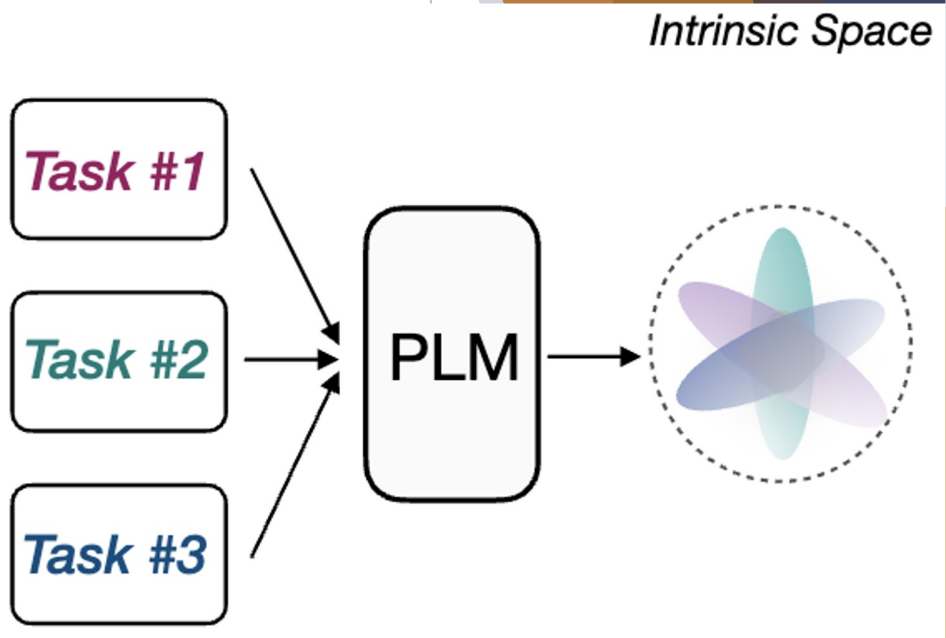
- This method is based on the finding that the full-parameter fine-tuning of pre-trained models (PLMs) can be effectively reparameterized into a low-dimensional subspace.
- By transforming parameters to a low-dimensional subspace, we can retain up to 85% performance when compared to traditional fine-tuning.




University of Virginia | ENGINEERING Department of Computer Science

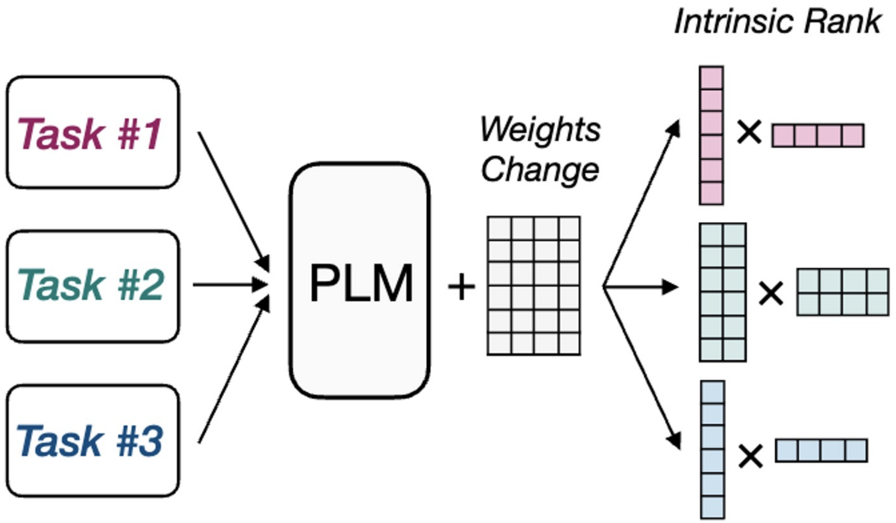
# Reparameterization-based delta-tuning: Intrinsic Space of Multiple Adaptations

- This approach takes the reparameterization concept further by hypothesizing that adaptations to multiple tasks can be optimized within a shared low-dimensional intrinsic subspace.
- Instead of creating separate adaptations for each task, it's possible to reparameterize these adaptations within a single low-dimensional subspace.
- They showed that by tuning only 250 parameters in this subspace, they could recover 97% and 83% of full prompt tuning performance for 100 seen and 20 unseen tasks, respectively.



# Reparameterization-based delta-tuning: Intrinsic Rank of Weight Differences

- Inspired by the concept of intrinsic dimensions, this method, specifically LoRA, hypothesizes that weight changes during model tuning have a low intrinsic rank.
- It involves optimizing a low-rank decomposition of the original weight matrices specifically within self-attention modules.
    - Less weights to train.
- This method has matched the performance of traditional fine-tuning on the GLUE benchmark.
- Effectiveness demonstrated across various scales and architectures of PLMs: focusing on critical components rather than the entire model can yield efficient and effective adaptation.



# Performance

Methodology:

- Comparison involved vanilla fine-tuning (FT) and four delta tuning methods—Prompt Tuning (PT), Prefix-Tuning (PF), LoRA (LR), and Adapter (AP) across over 100 diverse NLP tasks.

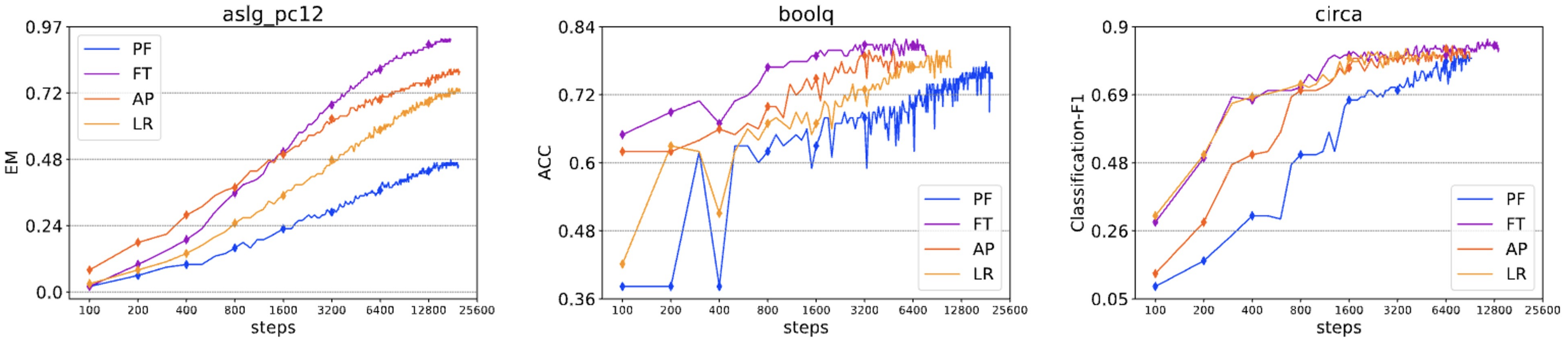
**Performance Results:**

- General finding: Delta tuning methods, with fewer tunable parameters, often underperform compared to FT. However, the performance gap is not vast, indicating their potential in large-scale applications.
- Relative Performance: FT generally outperforms delta methods, with the rank order being FT > LR > AP > PF > PT.
- The structure of delta tuning methods appears more influential than the sheer number of tunable parameters in determining performance.
- Larger model sizes (e.g., T5LARGE) show improved performance for PT, suggesting that the scale can mitigate some performance deficits.

# Convergence

**Convergence Results:**

- Visualization of training progress reveals that FT converges fastest, followed closely by AP and LR, with PF showing slower convergence rates.
- Convergence Dependency: The convergence of delta tuning methods is not highly sensitive to the number of tunable parameters but is influenced more by the structure of the tuning approach.
- Scaling Benefits: As PLM scales increase, delta tuning methods show faster convergence, which corroborates the performance benefits seen at larger scales.



# Efficiency

**Efficiency Results:**

- Delta tuning methods significantly reduce GPU memory usage compared to FT, particularly at smaller batch sizes—saving up to three-fourths of GPU memory.
- Efficiency across Scales: Even at larger batch sizes, delta tuning maintains a substantial memory efficiency advantage, saving at least one-third of GPU memory.

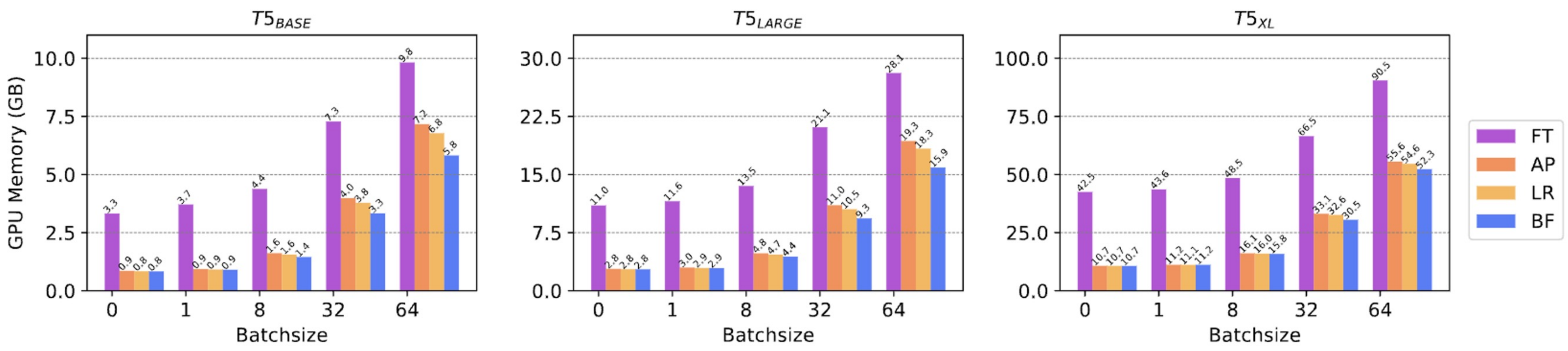



**Figure 9:** GPU memory consumed by each delta tuning methods compared with fine-tuning.

# The Power of Scale for Delta Tuning

Scaling Impact on Performance and Convergence:

- Significant improvements in both performance and convergence as PLM size increases from T5SMALL to T5XXL.
- Enhanced effects seen across various delta tuning methods on NLP tasks like MNLI, QNLI, and SST-2.

Different Delta-tuning Performance Across Scales:

- Prompt tuning underperforms on smaller-scale models but matches fine-tuning performance on models over 10 billion parameters.
- Other delta tuning methods competitive with fine-tuning even at smaller scales.

# Applications

- Fast Training and Shareable Checkpoints
  - Delta tuning makes for reduced training time memory efficient adaptation, and facilitates the sharing of trained checkpoints through platforms like AdapterHub and OpenDelta, promoting community-wide accessibility to efficient model tuning.
- Multi-Task Learning
  - Supports the development of versatile AI systems capable of handling multiple tasks simultaneously.
- Mitigation of Catastrophic Forgetting
  - By tuning minimal parameters, delta tuning helps maintain the knowledge acquired during pre-training, reducing the risk of catastrophic forgetting.
- Language Models as Services and In-Batch Parallel Computing
  - Delta tuning's lightweight nature makes it ideal for PLM services, reducing computation and storage requirements for service providers.
  - Enhances the practicality of services by supporting in-batch parallel computing, allowing simultaneous training or evaluation of instances from multiple users.

# Conclusion

1. Categorize and discuss the various delta tuning methods
2. They run some experiments and analysis on a variety of delta-tuning methods.
3. Discussion on applications of delta-tuning.

# DoRA: Weight-Decomposed Low-Rank Adaptation

NVIDIA

Presenter: Guangzhi Xiong

University of Virginia | Engineering Department of Computer Science

# Pattern Analysis of LoRA and FT

Research Question:

- Why is there an accuracy gap between LoRA and FT

Analysis Method (Weight Decomposition Analysis):

- The authors examine the updates in both **magnitude** and **direction** of the *LoRA* and *FT* weights relative to the *pre-trained* weights
- The weight decomposition of $W \in \mathbb{R}^{d \times k}$ can be formulated as

$$W = m\frac{V}{||V||_c} = ||W||_c \frac{W}{||W||_c}$$

$m \in \mathbb{R}^{1 \times k}$, $V \in \mathbb{R}^{d \times k}$

University of Virginia Engineering Department of Computer Science

# Pattern Analysis of LoRA and FT (cont.)

Analysis Method (cont.):

- Model: VL-BART. LoRA: Q/V matrix in SelfAttn.
- The authors decompose
  - The pretrained weight W_0
  - The full fine-tuned weight W_FT
  - The merged LoRA weight W_LoRA
- The magnitude and directional variation between W_0 and W_FT

$$\Delta M_{\mathrm{FT}}^{t} = \frac{\sum_{n=1}^{k} |m_{\mathrm{FT}}^{n,t} - m_0^n|}{k}$$

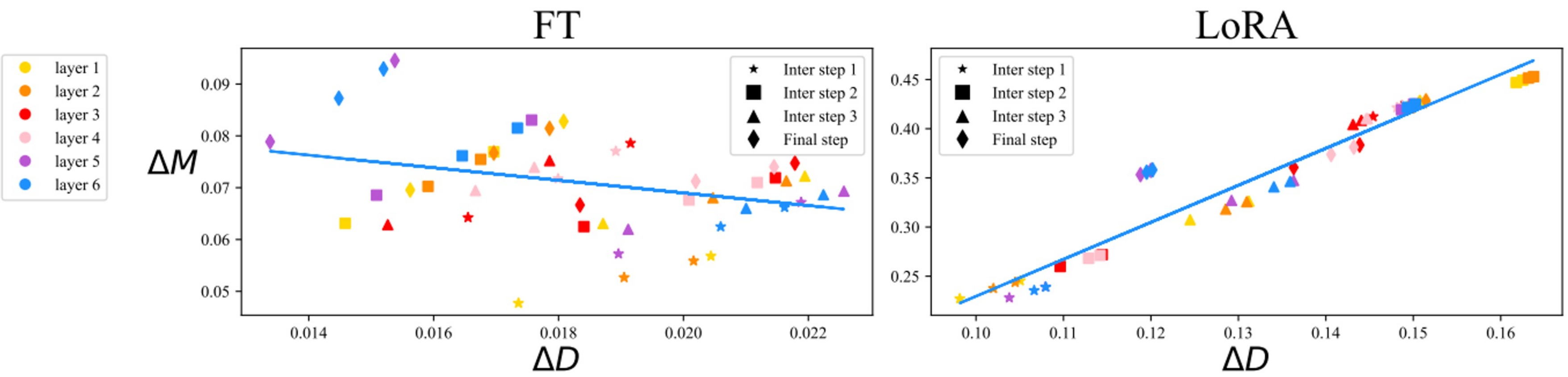
$$\Delta D_{\mathrm{FT}}^{t} = \frac{\sum_{n=1}^{k} (1 - \mathbf{cos}(V_{\mathrm{FT}}^{n,t}, W_0^n))}{k}$$

t: training step
n: column index
k: num. of columns

# Pattern Analysis of LoRA and FT (cont.)

Analysis Results:

- LoRA exhibits a consistent **positive** slope trend across all the intermediate steps.
- FT displays a more varied learning pattern with a relatively **negative** slope.
- LoRA does not show proficiency in executing slight directional changes alongside more significant magnitude alterations, or vice versa.




(a) (b)

UNIVERSITY of VIRGINIA | ENGINEERING Department of Computer Science

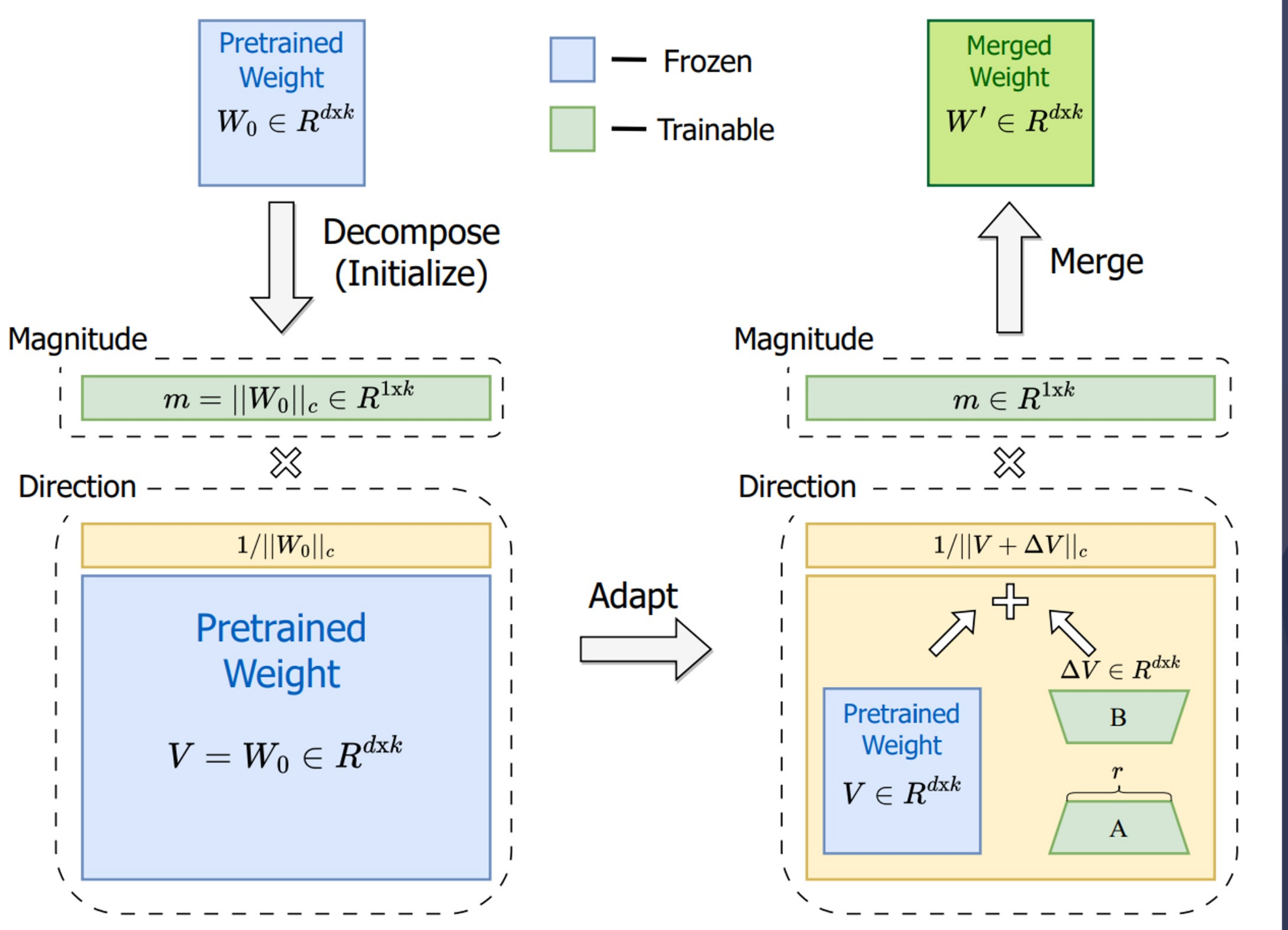
# Weight-Decomposed Low-Rank Adaptation (DoRA)

Method:

- decomposes the pretrained weight into its **magnitude** and **directional** components
- decompose the directional component with **LoRA**

Formula:

$$W' = \underline{m}\frac{V + \Delta V}{||V + \Delta V||_c} = \underline{m}\frac{W_0 + \underline{BA}}{||W_0 + \underline{BA}||_c}$$



# Weight-Decomposed Low-Rank Adaptation (cont.)

Visualization results

- DoRA, and FT are characterized by a distinct negative slope
  - FT: pre-trained weights possess substantial knowledge → having a larger magnitude or direction alteration alone is sufficient

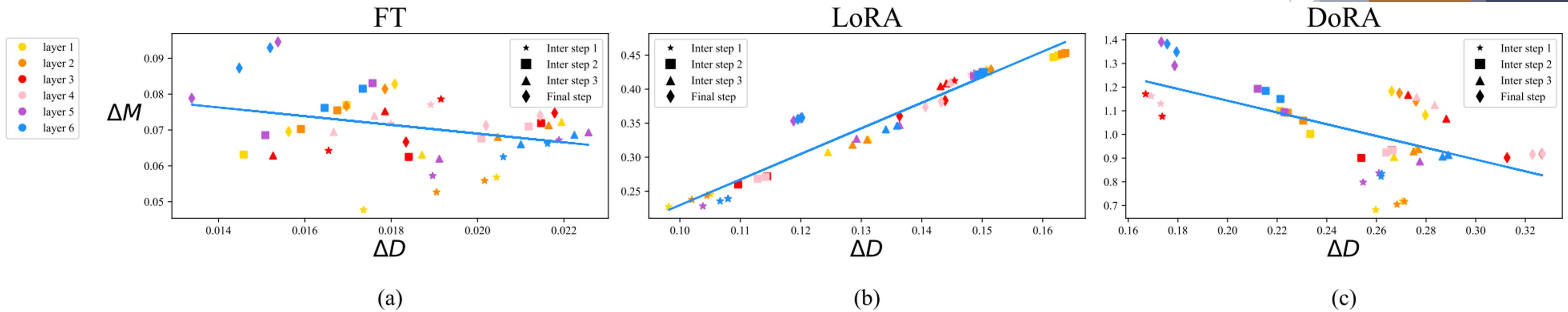



*Figure 2.* Magnitude and direction updates of (a) FT, (b) LoRA, and (c) DoRA of the query matrices across different layers and intermediate steps. Different markers represent matrices of different training steps and different colors represent the matrices of each layer.

# Experiments (Commonsense Reasoning)

*Table 1.* Accuracy comparison of LLaMA 7B/13B with various PEFT methods on eight commonsense reasoning datasets. Results of all the baseline methods are taken from (Hu et al., 2023). DoRA†: the adjusted version of DoRA with the rank halved.

| Model | PEFT Method | # Params (%) | BoolQ | PIQA | SIQA | HellaSwag | WinoGrande | ARC-e | ARC-c | OBQA | Avg. |
|---|---|---|---|---|---|---|---|---|---|---|---|
| ChatGPT | - | - | 73.1 | 85.4 | 68.5 | 78.5 | 66.1 | 89.8 | 79.9 | 74.8 | 77.0 |
| LLaMA-7B | Prefix | 0.11 | 64.3 | 76.8 | 73.9 | 42.1 | 72.1 | 72.9 | 54.0 | 60.6 | 64.6 |
| | Series | 0.99 | 63.0 | 79.2 | 76.3 | 67.9 | 75.7 | 74.5 | 57.1 | 72.4 | 70.8 |
| | Parallel | 3.54 | 67.9 | 76.4 | 78.8 | 69.8 | 78.9 | 73.7 | 57.3 | 75.2 | 72.2 |
| | LoRA | 0.83 | 68.9 | 80.7 | 77.4 | 78.1 | 78.8 | 77.8 | 61.3 | 74.8 | 74.7 |
| | DoRA† (Ours) | 0.43 | 70.0 | 82.6 | 79.7 | 83.2 | 80.6 | 80.6 | 65.4 | 77.6 | **77.5** |
| | DoRA (Ours) | 0.84 | 68.5 | 82.9 | 79.6 | 84.8 | 80.8 | 81.4 | 65.8 | 81.0 | **78.1** |
| LLaMA-13B | Prefix | 0.03 | 65.3 | 75.4 | 72.1 | 55.2 | 68.6 | 79.5 | 62.9 | 68.0 | 68.4 |
| | Series | 0.80 | 71.8 | 83 | 79.2 | 88.1 | 82.4 | 82.5 | 67.3 | 81.8 | 79.5 |
| | Parallel | 2.89 | 72.5 | 84.9 | 79.8 | 92.1 | 84.7 | 84.2 | 71.2 | 82.4 | 81.4 |
| | LoRA | 0.67 | 72.1 | 83.5 | 80.5 | 90.5 | 83.7 | 82.8 | 68.3 | 82.4 | 80.5 |
| | DoRA† (Ours) | 0.35 | 72.5 | 85.3 | 79.9 | 90.1 | 82.9 | 82.7 | 69.7 | 83.6 | **80.8** |
| | DoRA (Ours) | 0.68 | 72.4 | 84.9 | 81.5 | 92.4 | 84.2 | 84.2 | 69.6 | 82.8 | **81.5** |

# Experiments (Vision)

## Image/Video-Text Understanding

*Table 2.* The multi-task evaluation results on VQA, GQA, NVLR$^2$ and COCO Caption with the VL-BART backbone.

| Method | # Params (%) | VQA$^{v2}$ | GQA | NVLR$^2$ | COCO Cap | Avg. |
|---|---|---|---|---|---|---|
| FT | 100 | 66.9 | 56.7 | 73.7 | 112.0 | 77.3 |
| LoRA | 5.93 | 65.2 | 53.6 | 71.9 | 115.3 | 76.5 |
| DoRA (Ours) | 5.96 | 65.8 | 54.7 | 73.1 | 115.9 | **77.4** |

*Table 3.* The multi-task evaluation results on TVQA, How2QA, TVC, and YC2C with the VL-BART backbone.

| Method | # Params (%) | TVQA | How2QA | TVC | YC2C | Avg. |
|---|---|---|---|---|---|---|
| FT | 100 | 76.3 | 73.9 | 45.7 | 154 | 87.5 |
| LoRA | 5.17 | 75.5 | 72.9 | 44.6 | 140.9 | 83.5 |
| DoRA (Ours) | 5.19 | 76.3 | 74.1 | 45.8 | 145.4 | **85.4** |

## Video Instruction Tuning

*Table 12.* Visual instruction tuning evaluation result of DoRA, LoRA, and FT for LLaVA-1.5-7B on a wide range of 7 vision-language tasks.

| Method | # Params (%) | VQA$^{v2}$ | GQA | VisWiz | SQA | VQA$^T$ | POPE | MMBench | Avg. |
|---|---|---|---|---|---|---|---|---|---|
| FT | 100 | 78.5 | 61.9 | 50.0 | 66.8 | 58.2 | 85.9 | 64.3 | 66.5 |
| LoRA | 4.61 | 79.1 | 62.9 | 47.8 | 68.4 | 58.2 | 86.4 | 66.1 | 66.9 |
| DoRA (Ours) | 4.63 | 78.6 | 62.9 | 52.2 | 69.9 | 57 | 87.2 | 66.1 | **67.6** |

# Experiments (Compatibility)

DVoRA = VeRA + DoRA

*Table 5.* Average scores on MT-Bench assigned by GPT-4 to the answers generated by fine-tuned LLaMA-7B/LLaMA2-7B.

| Model | PEFT Method | # Params (%) | Score |
|---|---|---|---|
| LLaMA-7B | LoRA | 2.31 | 5.1 |
| | DoRA (Ours) | 2.33 | **5.5** |
| | VeRA | 0.02 | 4.3 |
| | DVoRA (Ours) | 0.04 | **5.0** |
| LLaMA2-7B | LoRA | 2.31 | 5.7 |
| | DoRA (Ours) | 2.33 | **6.0** |
| | VeRA | 0.02 | 5.5 |
| | DVoRA (Ours) | 0.04 | **6.0** |

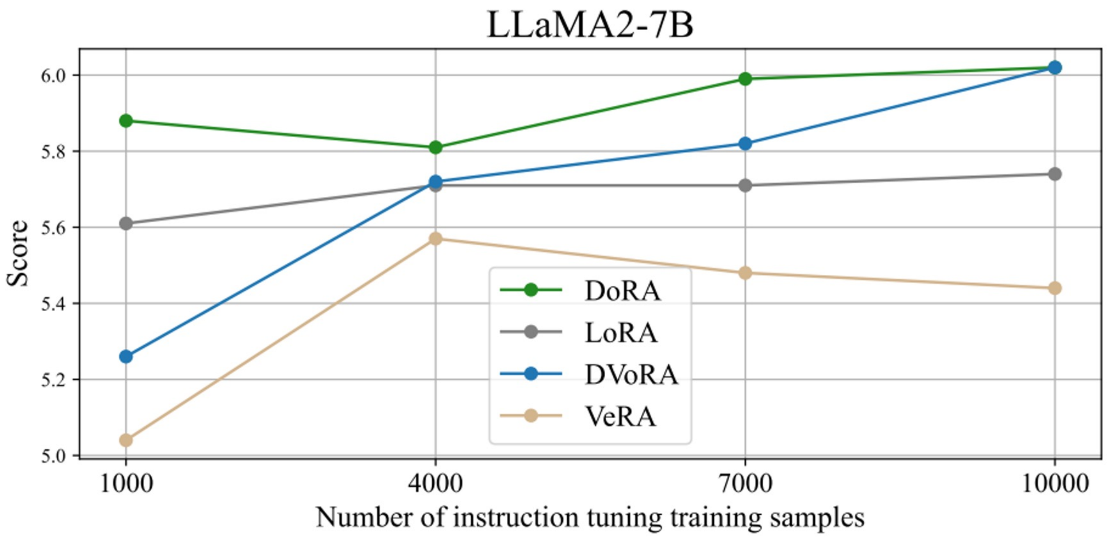



*Figure 3.* Performance of fine-tuned LLaMA2-7B on MT-Bench using different numbers of Alpaca training samples.

University of Virginia | ENGINEERING
Department of Computer Science

# Experiments (Robustness)

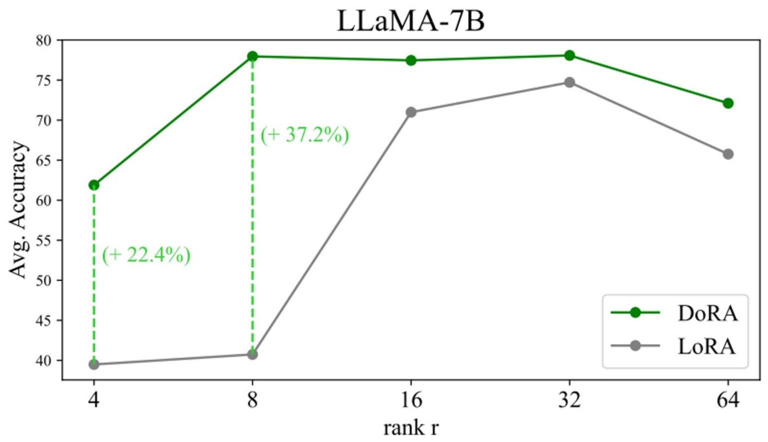



*Table 15.* Accuracy comparison of LoRA and DoRA with varying ranks for LLaMA-7B on the commonsense reasoning tasks.

| PEFT Method | rank r | # Params (%) | BoolQ | PIQA | SIQA | HellaSwag | WinoGrande | ARC-e | ARC-c | OBQA | Avg. |
|---|---|---|---|---|---|---|---|---|---|---|---|
| LoRA | 4 | 0.10 | 2.3 | 46.1 | 18.3 | 19.7 | 55.2 | 65.4 | 51.9 | 57 | 39.5 |
| | 8 | 0.21 | 31.3 | 57.0 | 44.0 | 11.8 | 43.3 | 45.7 | 39.2 | 53.8 | 40.7 |
| | 16 | 0.42 | 69.9 | 77.8 | 75.1 | 72.1 | 55.8 | 77.1 | 62.2 | 78.0 | 70.9 |
| | 32 | 0.83 | 68.9 | 80.7 | 77.4 | 78.1 | 78.8 | 77.8 | 61.3 | 74.8 | 74.7 |
| | 64 | 1.64 | 66.7 | 79.1 | 75.7 | 17.6 | 78.8 | 73.3 | 59.6 | 75.2 | 65.8 |
| DoRA (Ours) | 4 | 0.11 | 51.3 | 42.2 | 77.8 | 25.4 | 78.8 | 78.7 | 62.5 | 78.6 | 61.9 |
| | 8 | 0.22 | 69.9 | 81.8 | 79.7 | 85.2 | 80.1 | 81.5 | 65.7 | 79.8 | 77.9 |
| | 16 | 0.43 | 70.0 | 82.6 | 79.7 | 83.2 | 80.6 | 80.6 | 65.4 | 77.6 | 77.5 |
| | 32 | 0.84 | 68.5 | 82.9 | 79.6 | 84.8 | 80.8 | 81.4 | 65.8 | 81.0 | 78.1 |
| | 64 | 1.65 | 69.9 | 81.4 | 79.1 | 40.7 | 80.0 | 80.9 | 65.5 | 79.4 | 72.1 |

# Recent Large Language Models Reshaping the Open-Source Arena

Blog Post by Deci Team

Presenter: Sabit Ahmed

University of Virginia | Engineering Department of Computer Science

# Overview

There is explosion of LLMs in current time. Models are released on a daily basis.

This article helps to-

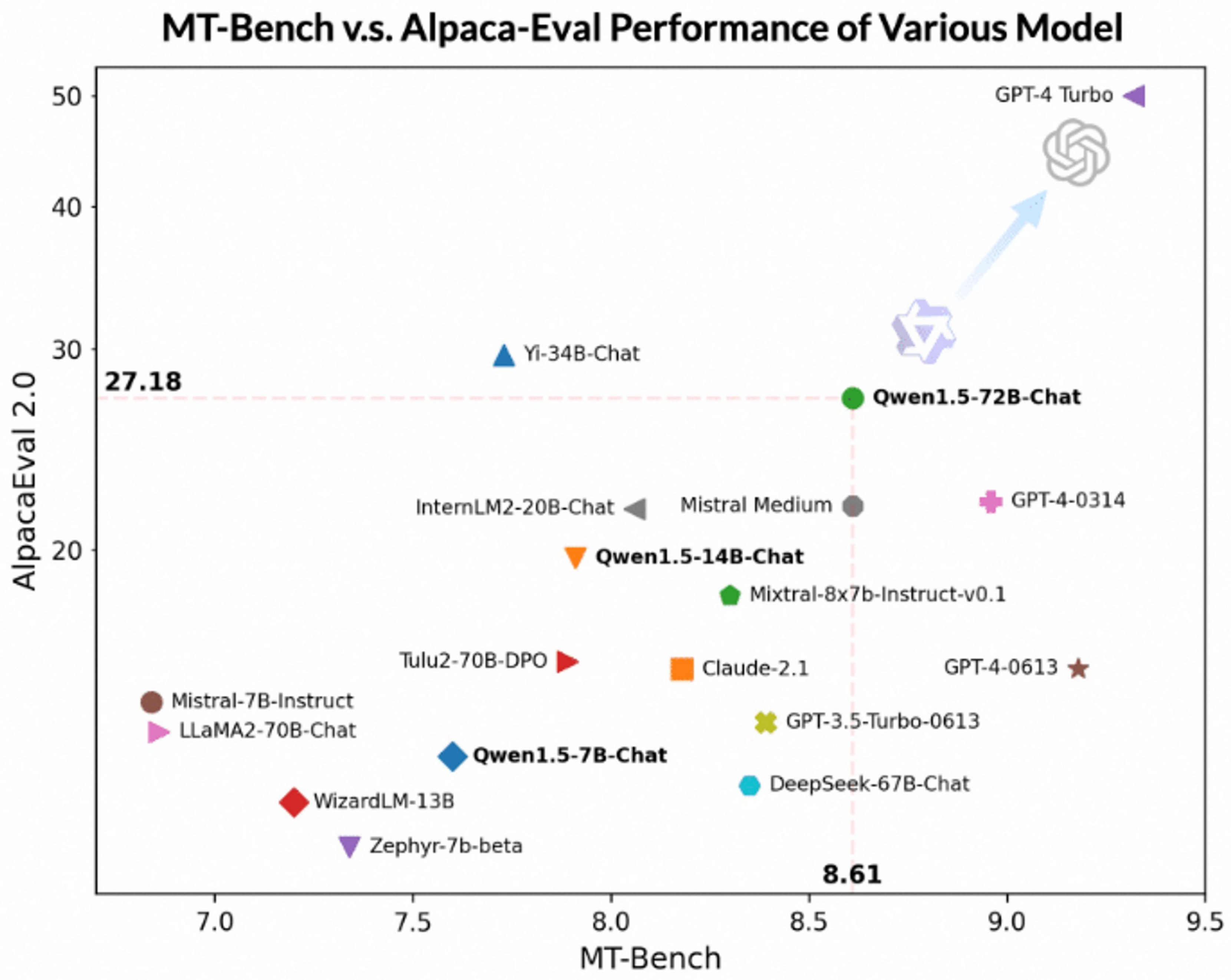
- Reflect the latest developments in **open source** LLMs
- Curate and select a list of intriguing and influential models
- Provide an in-depth exploration with key details i.e., architectural design, benchmark scores, etc.

# Background

- Most commonly used architecture: Llama 2 7B
- Attentions used: Multi-head attention (MHA), multi-query attention (MQA), group-query attention (GQA)
- Alignments:
  - Direct Preference Optimization (DPO)
- Benchmark used to evaluate:
  - MT-Bench (provides a score, the higher the better)
  - Chatbot Arena Leaderboard (ranks LLMs based on human pairwise comparisons)

# Qwen1.5 (Alibaba Cloud)

- **Version**: Base and chat (Sizes: 0.5B, 1.8B, 4B, 7B, 14B, 72B)
- **Fine-tuning and Alignment Details**: Alignment with DPO (Direct preference Optimization)
- **Architectural Notes**: Uses Transformer architecture, SwiGLU activation, attention QKV bias, GQA, and combines sliding window attention with full attention
- **Performance:** Qwen1.5-72B-chat outperforms Claude-2.1, GPT-3.5-Turbo, Mixtral-8x7b-instruct, etc (MT-Bench and AlpacaEval).
- **Interesting Facts**: Base model supports 12 languages



MT-Bench v.s. Alpaca-Eval Performance of Various Model
AlpacaEval 2.0
MT-Bench
50
40
30
20
7.0
7.5
8.0
8.5
9.0
9.5
27.18
8.61
GPT-4 Turbo
Yi-34B-Chat
Qwen1.5-72B-Chat
InternLM2-20B-Chat
Mistral Medium
GPT-4-0314
Qwen1.5-14B-Chat
Mixtral-8x7b-Instruct-v0.1
Tulu2-70B-DPO
Claude-2.1
GPT-4-0613
Mistral-7B-Instruct
LLaMA2-70B-Chat
GPT-3.5-Turbo-0613
Qwen1.5-7B-Chat
DeepSeek-67B-Chat
WizardLM-13B
Zephyr-7b-beta

# Yi (01.AI)

- **Versions:** Base and chat (Sizes: 6B, 9B, 34B)
- **Pretraining Data:** A curated dataset of 3.1 trillion English and Chinese tokens derived from CommonCrawl through cascaded data deduplication and quality filtering
- **Fine-tuning and Alignment Details:** Base models underwent SFT using 10K multi-turn instruction-response dialogue pairs
- **Architectural Notes:** SwiGLU activation, GQA, and RoPE
- **Performance:** Y-34B performs close to GPT 3.5. (#18 on Chatbot Arena leaderboard)
- **Interesting Facts:** Innovative data cleaning pipeline, 200k context window

# Smaug (Abacus.AI)

- **Versions**: Chat (Sizes: 72B, 34B)
- **Pretraining Data**: 72B - same as Qwen 1.5; 34B - same as Yi 34B
- **Fine-tuning and Alignment Details**: Alignment with Direct Preference Optimization-Positive (DPOP)
- **Architectural Notes**: Smaug-72B is based on Qwen-72B; Smaug-34B is based on Yi-34B
- **Interesting Facts**: First model to surpass an average of 80% on Open LLM Leaderboard, One of the top models.

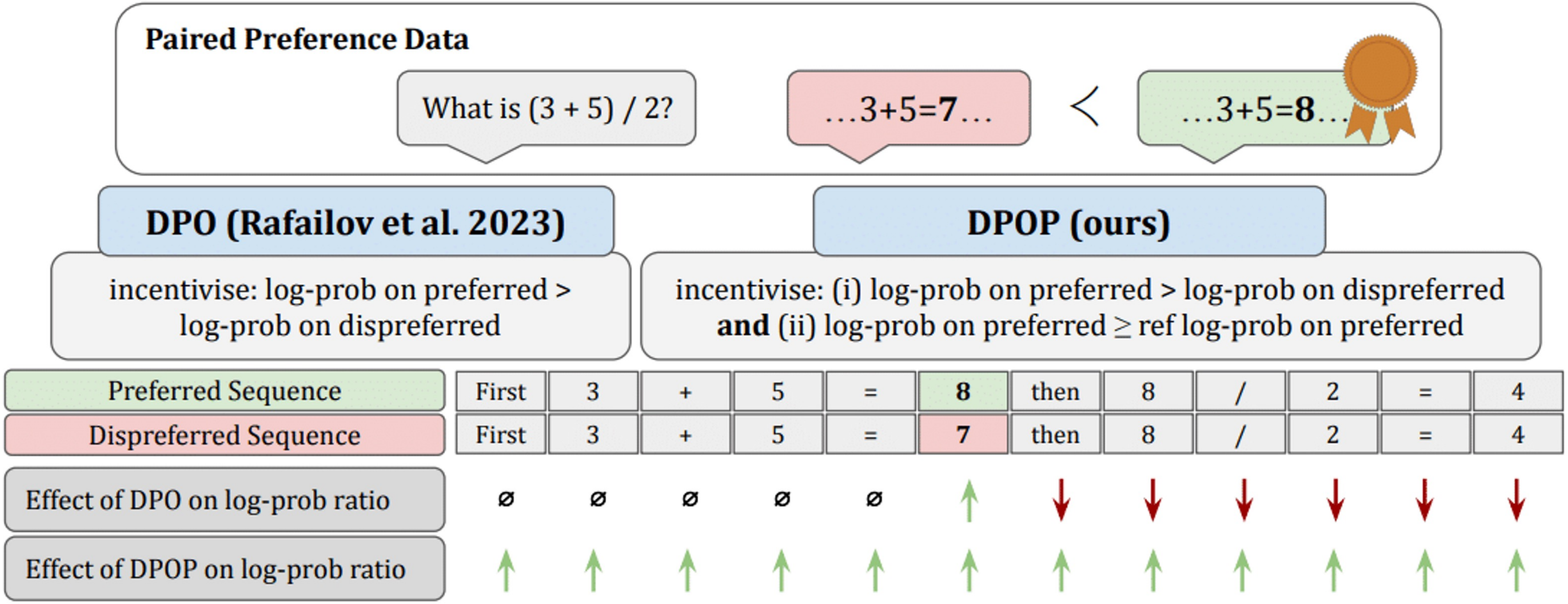



Figure 1: **DPOP avoids a failure mode of DPO.** When preference pairs differ on only a few tokens, DPO receives no loss incentive at all for the early tokens, and a loss incentive that in some cases can lead to degradation of the log-probs of later tokens (Section 3). We introduce DPOP, which adds a new term to the loss which leads every token to be incentivised toward the preferred completion (Section 4).

# Mixtral-8x7B (mistralai)

- **Sizes**: 46.7B parameters, uses only 12.9B parameters per token
- **Versions**: Base and instruct
- **Pretraining Data**: Undisclosed
- **Fine-tuning and Alignment Details**: Undisclosed
- **Architectural Notes**: Mixture of Experts (MoE) using 8 Mixtral-7B models
- **Performance:** MT Bench score of 8.3. In terms of human evaluation, it is tied with Claude-2.1, GPT-3.5 Turbo 0613 and Gemini Pro on the Chatbot Arena leaderboard.

# DBRX (Databricks)

- **Sizes**: 132B parameters; uses only 36B per input
- **Versions:** Base and instruct
- **Pretraining Data:** Carefully curated dataset comprising 12T tokens from text and code data; employed curriculum learning strategies
- **Fine-tuning and Alignment Details:** Undisclosed
- **Architectural Notes:** Uses GLU, RoPE, and GQA; GPT-4 tokenizer
- **Performance:** Surpass Mixtral-8x7b-instruct-v0.1
- **Interesting Facts**: Fine-grained MoE model, using 4 out of 16 experts per input (For Mixtral 2 out of 8 experts were used)

# SOLAR-10.7B (Upstage AI)

- **Versions**: Base and instruct (**Sizes**: 10.7B)
- **Fine-tuning and Alignment Details**: Mix of open-source datasets along with a specially synthesized math QA dataset aimed at boosting the model's mathematical abilities (i.e., Math-Instruct datasets)
- **Architectural Notes**: Depth upscaling, starting with a Llama 2 7B architecture with Mistral 7B weights, adding layers to increase model depth, followed by continued pretraining
- **Performance:** SOLAR-10.7B-v1.0-instruct is #30 on Chatbot Arena leaderboard

UNIVERSITY of VIRGINIA | ENGINEERING
Department of Computer Science

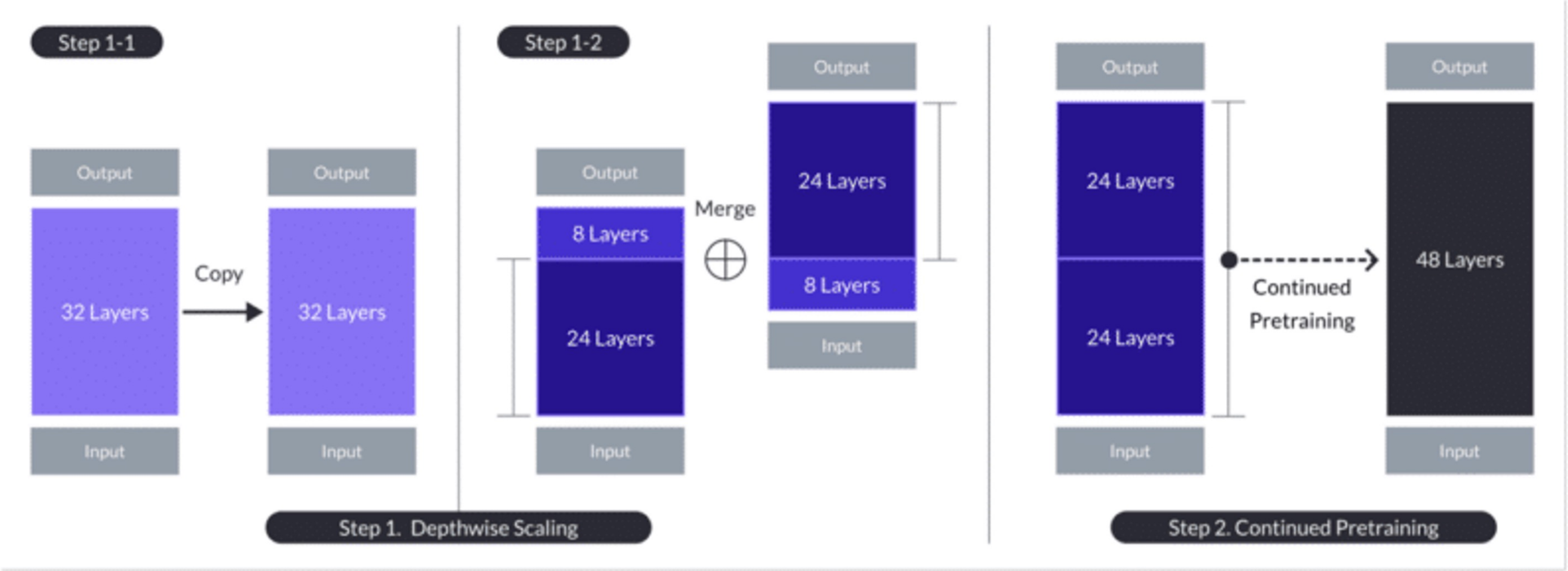



Figure 1: Depth up-scaling for the case with $n = 32$, $s = 48$, and $m = 8$. Depth up-scaling is achieved through a dual-stage process of depthwise scaling followed by continued pretraining.

# TULU v2 (Allen Institute for AI)

- **Sizes**: 7B, 13B, 70B
- **Versions**: Instruct and chat
- **Pretraining Data**: Same as Llama 2
- **Fine-tuning and Alignment Details:** SFT on the TULU-v2-mix dataset; DPO alignment on the UltraFeedback dataset
- **Architectural Notes**: Same as Llama 2
- **Interesting Facts**: DPO significantly enhances model performance on AlpacaEval benchmark while maintaining performance on other tasks
- **Performance:** MT Bench score of 7.89. Tied with Yi-34-B-Chat, GPT-3.5-Turbo models based on Chatbot Arena leaderboard

# WizardLM (Microsoft)

- **Sizes**: A series of models fine-tuned from Llama 7B, 13B, 30B, 70B
- **Versions**: Base and instruct
- **Fine-tuning and Alignment Details**: Fine-tuning using the Evol-Instruct approach, which uses LLMs to generate complex instructions
  - Evol-Instruct: autonomously generate open-domain instructions across different complexity levels
  - In-depth evolving and in-breadth evolving
- **Architectural Notes**: Same as Llama
- **Performance:** Outperforms ChatGPT in certain complex tasks.
- **Interesting Facts**: Use of LLMs to automatically rewrite an initial set of instructions into more complex ones

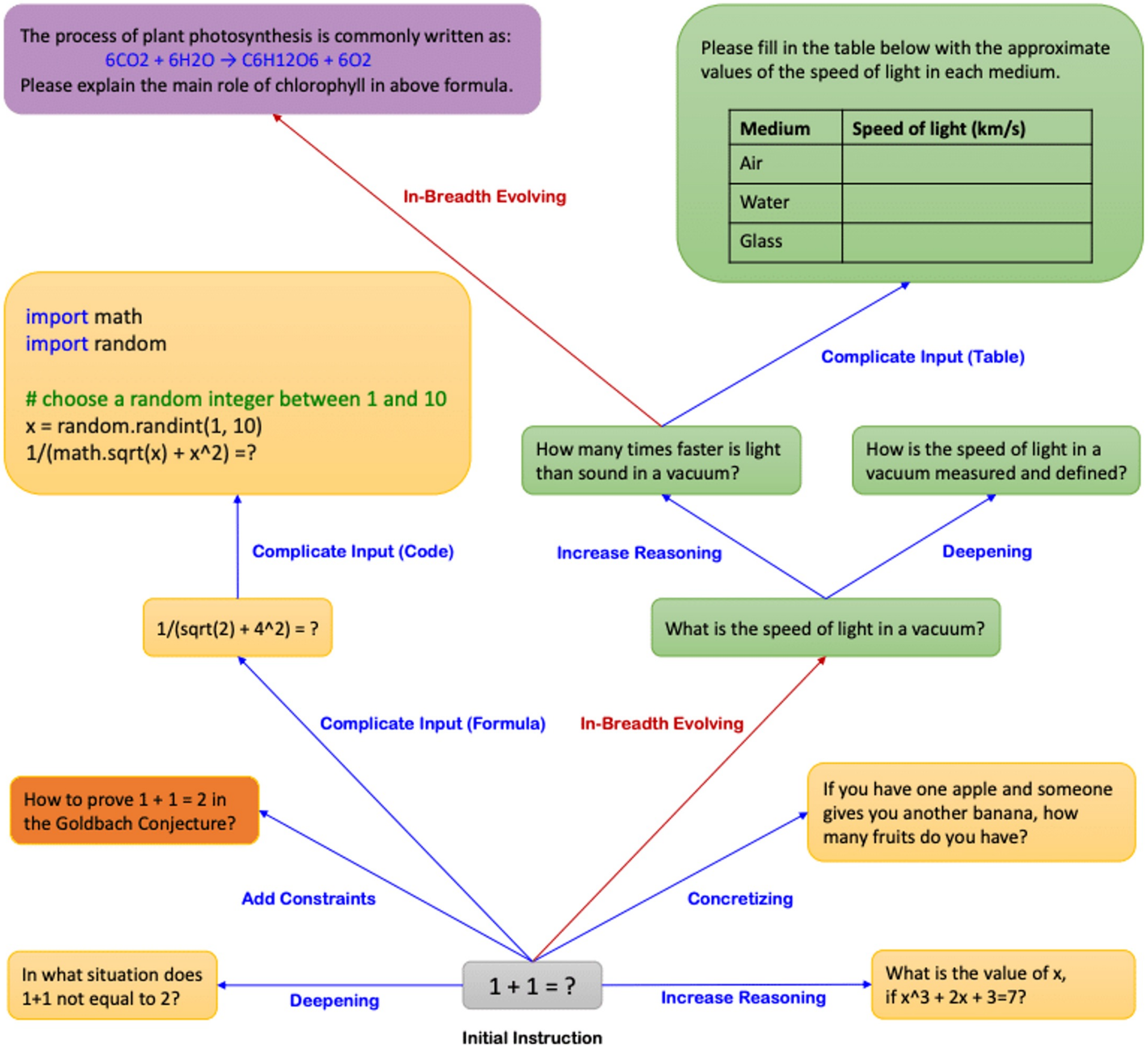



Figure 1: Running Examples of *Evol-Instruct*.

| Model Family Name | Created By | Sizes | Versions | Pretraining Data | Fine-tuning and Alignment Details | License | What's interesting | Architectural Notes |
|---|---|---|---|---|---|---|---|---|
| OLMo | Allen Institute for AI | 1B, 7B | Base, SF and instruct | Trained on Dolma using the AdamW optimized | SFT using the TULU 2 dataset followed by aligning with distilled preference data using DPO | Apache 2.0 | Release fosters collaborative research, providing training data, training and evaluation code, and intermediate checkpoints | SwiGLU activation, RoPE, and BPE-based tokenizer |
| Gemma | Google Deepmind | 2B, 7B | Base and instruct | 6T tokens of text, using similar training recipes as Gemini | SFT on a mix of synthetic and human-generated text and RLHF | Gemma Terms of Use | Instruct model uses formatter that adds extra information during training and inference | GeGLU activations, RoPE and RMSNorm; 2B uses MQA and 7B uses MHA |
| DeciLM-7B | Deci | 7B | Base and instruct | Undisclosed | LoRA finetuned on SlimOrca | Apache 2.0 | Use of Variable GQA and efficient architecture generated using NAS technology | SwiGLU activations, RoPE, and Variable GQA |

Figure: Other Notable Large Language Models

# Top Ten List of LLMs in Open Source Arena

deci

| Model Family Name | Created By | Sizes | Versions | Pretraining Data | Fine-tuning and Alignment Details | License | What's interesting | Architectural Notes |
|---|---|---|---|---|---|---|---|---|
| Qwen 1.5 | Alibaba Cloud | 0.5B, 1.8B, 4B, 7B, 14B, 72B | Base and chat | Undisclosed | Alignment with DPO | Tongyi Qianwen | Models excel in 12 languages; Qwen 1.5 72B Chat currently the top non-proprietary model on Chatbot Arena | Uses SwiGLU activation, attention QKV bias, GQA, and combines sliding window attention with full attention |
| Yi | 01.AI | 6B, 9B, 34B | Base and chat | A curated dataset of 3.1 trillion English and Chinese tokens derived from CommonCrawl through cascaded data deduplication and quality filtering | Base models underwent SFT using 10K multi-turn instruction-response dialogue pairs, refined through several iterations based on feedback | Yi Series Models Community License Agreement | Innovative data cleaning pipeline and data quality over quantity for fine tuning; 200k context window | SwiGLU activation, GQA, and RoPE |
| Smaug | Abacus.AI | 72B, 34B | Chat | 72B - same as Qwen 1.5 72B; 34B - same as Yi 34B | Alignment with Direct Preference Optimization-Postivie (DPOP) | 72B - Tongyi Qianwen; 34B - Yi Series Models Community License Agreement | First model to surpass an average of 80% on Open LLM Leaderboard | 72B - same as Qwen 1.5 34B - same as Yi |
| Mixtral-8x7B | mistralai | 46.7B parameters, uses only 12.9B parameters per token | Base and instruct | Undisclosed | Undisclosed | Apache 2.0 | Sparse Mixture of Experts (MoE) model; MT Bench score of 8.3 | MoE using 8 Mistral-7B models |
| DBRX | Databricks | 132B parameters; uses only 36B per input | Base and instruct | Carefully curated dataset comprising 12T tokens from text and code data; employed curriculum learning strategies | Undisclosed | Databricks Open Model License | Fine-grained MoE model, using 4 out of 16 experts per input | Uses GLU, RoPE, and GQA; GPT-4 tokenizer |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| SOLAR-10.7B | Upstage | 10.7B | Base and instruct | Same as Mistral 7B (undisclosed) | Instruction tuning employed Alpaca-GPT4, OpenOrca, and Synth. Math-Instruct datasets; alignment tuning used Orca DPO Pairs, Ultrafeedback Cleaned, and Synth. Math-Alignment datasets | Apache 2.0 | Depth upscaling, starting with a Llama 2 7B architecture with Mistral 7B weights, adding layers to increase model depth, followed by continued pretraining | Depth upscaled Llama 2 7B architecture |
| TÜLU v2 | Allen Institute for AI | 7B, 13B, 70B | Instruct and chat | Same as Llama 2 | SFT on the TULU-v2-mix dataset; DPO alignment on the UltraFeedback dataset | AI2 ImpACT Low-risk license | DPO significantly enhances model performance on AlpacaEval benchmark while maintaining performance on other tasks | Same as Llama 2 |
| WizardLM | WizardLM | 7B, 13B, 30B, 70B | Base and instruct | Same as Llama | Fine-tuning using the Evol-Instruct approach, which uses LLMs to generate complex instructions | Llama 2 Community License | Use of LLMs to automatically rewrite an initial set of instructions into more complex ones | Same as Llama |
| Starling 7B Alpha | Berkeley | 7B | Chat | Same as Mistral 7B | Trained from Openchat 3.5 7B using RLAIF and Advantage-induced Policy Alignment (APA) | LLaMA license | Use of Nectar dataset consisting of 3.8M GPT4 labeled pairwise comparisons to train a reward model; MT Bench score of 8.09 | Same as Mistral 7B |
| OLMo | Allen Institute for AI | 1B, 7B | Base, SF and instruct | Trained on Dolma using the AdamW optimized | SFT using the TULU 2 dataset followed by aligning with distilled preference data using DPO | Apache 2.0 | Release fosters collaborative research, providing training data, training and evaluation code, and intermediate checkpoints | SwiGLU activation, RoPE, and BPE-based tokenizer |
| Gemma | Google Deepmind | 2B, 7B | Base and instruct | 6T tokens of text, using similar training recipes as Gemini | SFT on a mix of synthetic and human-generated text and RLHF | Gemma Terms of Use | Instruct model uses formatter that adds extra information during training and inference | GeGLU activations, RoPE and RMSNorm; 2B uses MQA and 7B uses MHA |
| DeciLM-7B | Deci | 7B | Base and instruct | Undisclosed | LoRA finetuned on SlimOrca | Apache 2.0 | Use of Variable GQA and efficient architecture generated using NAS technology | SwiGLU activations, RoPE, and Variable GQA |

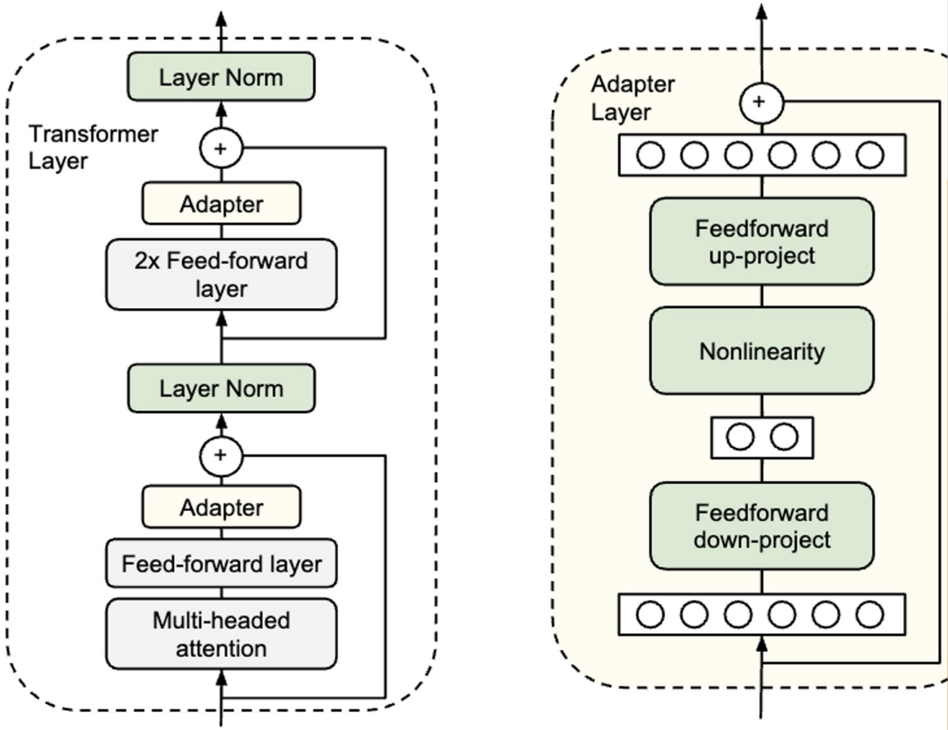
# Thank you!

University of Virginia | ENGINEERING
Department of Computer Science

# Appendix

# Addition-based delta-tuning :Adapter-based tuning

- This method involves adding neural modules called adapters to certain parts of the PLM.
- Adapters usually contain down-projection and up-projection components:
- This component projects input features from a high-dimensional space *d* to a lower-dimensional space *r* using a parameter matrix *Wd*. A nonlinear function *f*(·) is then applied to this reduced representation.
- Following the down-projection and nonlinear transformation, the up-projection component maps the data back from the r-dimensional space to the original d-dimensional space using another parameter matrix Wu .

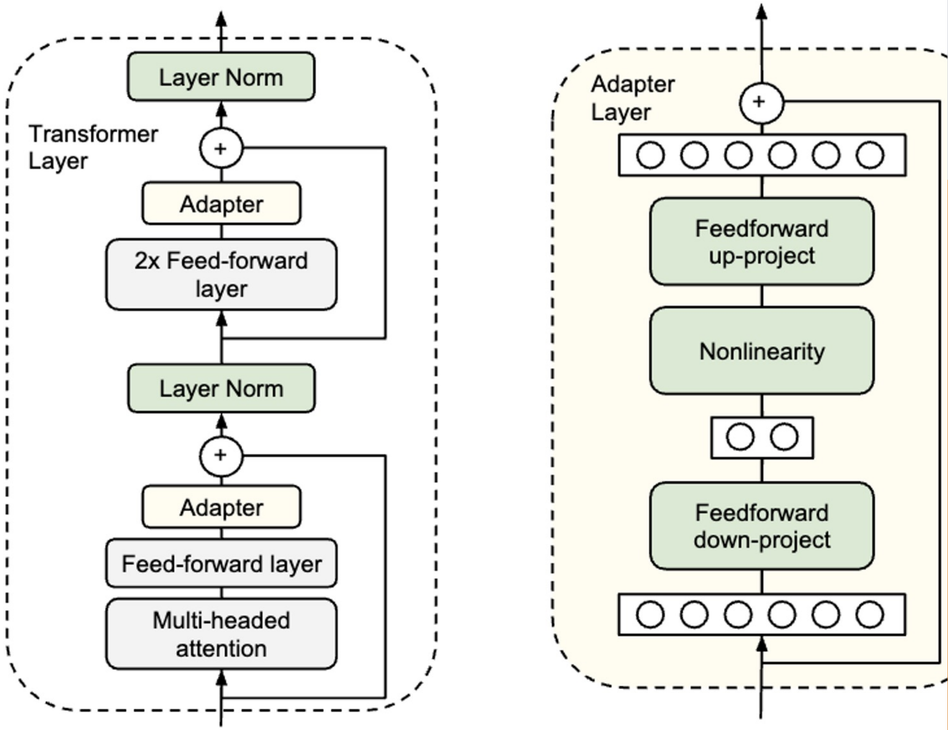



(Houlsby et al.,2019)

UNIVERSITY of VIRGINIA | ENGINEERING Department of Computer Science

# Addition-based delta-tuning :Adapter-based tuning

- Residual connection is added to the end of the up projection to preserve the original information and promote learning stability.
- Using adapters, often only about 0.5% to 8% of the total model parameters need tuning.
- Adapter-based tuning is advantageous in multi-task learning settings, where different adapter modules can be trained for different tasks and combined to leverage cross-task knowledge.

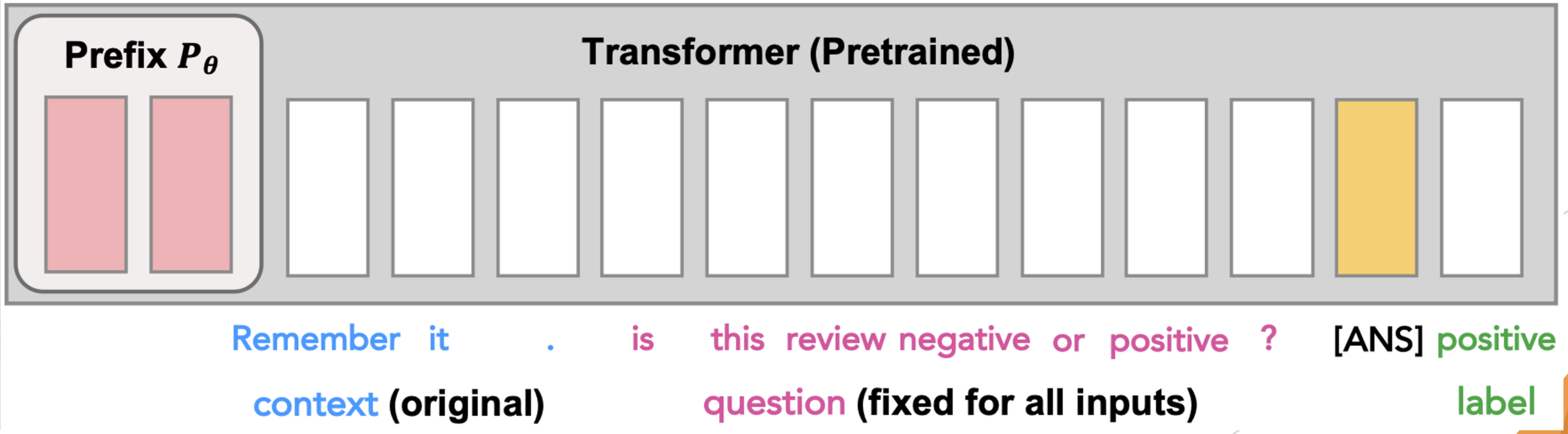



(Houlsby et al.,2019)

UNIVERSITY of VIRGINIA | ENGINEERING Department of Computer Science

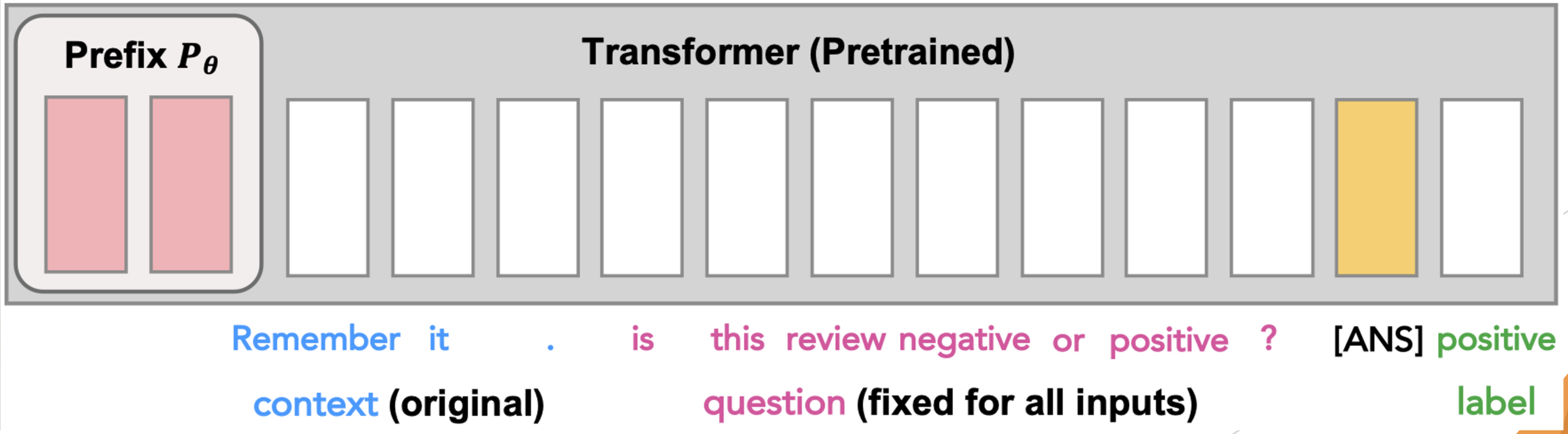
# Addition-based delta-tuning: Prompt-tuning

- Prefix Tuning: this technique involves prepending trainable prefixes to the input and hidden states of each Transformer layer. These prefixes are represented by a parameter matrix P and are optimized during training while the original model parameters remain unchanged. This method can be applied to both autoregressive and encoder-decoder models.
- Prompt Tuning: this method simplifies the concept by adding soft prompts only at the input layer. These prompts are also trainable and are optimized via gradient descent and the original model parameters are kept frozen.



# Addition-based delta-tuning: Prompt-tuning

- Both prefix and prompt tuning are shown to achieve promising performance, particularly in low-data scenarios, demonstrating that small-scale tuning can be effective.
- But,....
- Despite their advantages, prompt-based methods can be challenging to optimize, particularly with smaller datasets and model sizes.
- Training of soft prompts often converges slower than traditional fine-tuning methods, making it a critical area for further research and optimization.



# Specification-based delta-tuning: Heuristic-based tuning

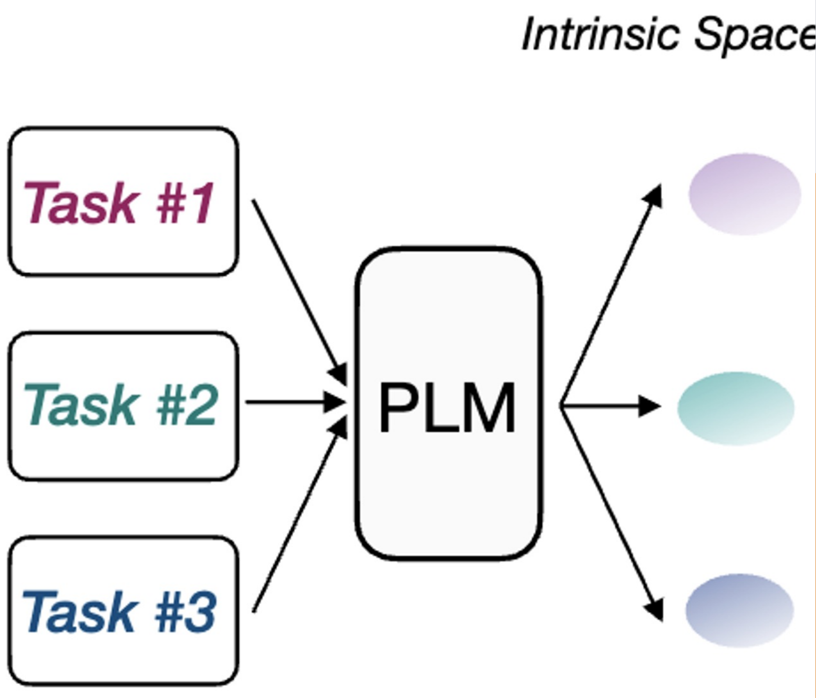
- In heuristic specification, certain parameters are directly specified for optimization based on simple yet effective strategies.
- Lee et al. (2019): Demonstrated significant performance by only fine-tuning one-fourth of the final layers of BERT and RoBERTa, achieving about 90% of the performance compared to full parameter fine-tuning.
- BitFit (Zaken et al., 2021): Showed that just optimizing bias terms, while keeping other parameters frozen, could still yield over 95% performance on several benchmarks. It was noted, however, that this strategy mainly showed effectiveness in smaller-scale models.

# Specification-based delta-tuning: Learn the specification

- Instead of manually choosing which parameters to optimize, this approach involves using algorithms to identify and optimize a selective set of parameters:
- Diff Pruning: Reparameterizes the model's parameters by adding a difference vector to the pre-trained parameters, aiming for sparsity in this difference vector. This method uses a differentiable approximation to the L0-norm to encourage fewer parameters to change, although it requires more GPU memory.
- Masking Method: Involves learning selective masks that determine which weights of the model should be updated for specific tasks. A binary matrix controls the updates through threshold functions and noisy estimators during back-propagation.
- These can efficiently adapt models to new tasks with fewer parameters being tuned, but there are practical challenges like increased memory demands (as seen in diff pruning) and potential performance limitations in larger models.

# Reparameterization-based delta-tuning: Intrinsic Dimensions of PLM Adaptation

- This method is based on the finding that the full-parameter fine-tuning of pre-trained models (PLMs) can be effectively reparameterized into a low-dimensional subspace.
- Some experiments show fine-tuning in a substantially lower-dimensional space can still achieve over 85% of the performance of traditional fine-tuning methods.
- It suggests that PLMs might function as compression frameworks, simplifying optimization from high-dimensional to low-dimensional spaces. This property becomes more pronounced in larger models, indicating that pre-training might inherently reduce a model's intrinsic dimensionality.



# Theoretical Perspective into Delta Tuning: Optimization Perspective for Delta

- Objective of Delta Tuning:
  - Fine-tune a small subset of parameters ($\delta$) to achieve performance similar to full model fine-tuning.
  - Reduce memory and computational costs compared to tuning all parameters ($\theta$).
- Optimization Framework:
  - Original function: F(θ) for the entire model.
  - Delta tuned function: $\tilde{\mathcal{F}}(\theta, \delta)$ focusing on a subset of parameters.
  - Initial State: $(\theta_0, \delta_0)$ where ideally $\tilde{\mathcal{F}}(\theta, \delta_0) = \mathcal{F}(\theta)$
- Optimization Strategy:
  - Analyze effects using conditions where $\tilde{\mathcal{F}}$ is Lipschitz continuously differentiable.
  - Emphasize optimization in a lower-dimensional subspace for efficiency.

# Theoretical Perspective into Delta Tuning: Optimization Perspective for Delta

- Low-Dimensional Approaches:
  - Solution Space: Implement techniques like LoRA and BitFit, focusing on critical parameter subsets such as low-rank matrices or bias terms.
  - Functional Space: Utilize adaptations in data flow via methods like Adapter and Prompt Tuning, modifying the input or feature space effectively.
- Practical Benefits:
  - Efficiency and Stability: More efficient and stable training processes due to reduced parameter count and focused tuning.
  - Scalability: Lower resource demands make it feasible for broader applications, including on less capable hardware.
- Theoretical Insights and Challenges:
  - Error Bound: Small deviations in $\delta$ lead to minor performance impacts, underlining robustness.
  - Transferability: Demonstrated potential for adaptability across different tasks, though effectiveness can vary by specific conditions.
- Unified Perspective:
  - Delta tuning methods share a common approach of low-dimensional modifications, optimizing critical aspects of the data flow in large models.

# Theoretical Perspective into Delta Tuning: Optimal Control Perspective for Delta Tuning

Connection to Optimal Control:

- Delta tuning viewed through optimal control, using control problem frameworks to model the training of deep learning networks.
- Core Concept: The discrete-time control problem uses a sequence of parameter updates to minimize loss over iterations.

Theoretical Background:

- Discrete-Time Pontryagin's Maximum Principle (PMP): Minimizes a cost function over a sequence of actions controlled by parameters θt
- Ensures that trajectory of state xt and co-state pt optimizes the Hamiltonian Ht

Method of Successive Approximations (MSA):

- Iterative optimization technique equated to the backpropagation used in training neural networks.
- Highlights how small, controlled changes in parameters (δ) guide the model to desired outputs efficiently.

# Comparisons and Experimental Discoveries

1. Performance Comparisons:
   - They conduct thorough comparisons among four representative delta tuning methods and traditional fine-tuning. This includes assessments of performance, convergence, and efficiency.
1. Combinability Analysis:
   - They explore the combinability of three representative delta tuning methods by assessing their performance when methods are combined simultaneously and sequentially.
1. Scaling Law Investigation:
   - They investigate the scaling laws, likely analyzing how changes in the size of the model or dataset affect the performance and efficiency of delta tuning methods.
1. Transferability Studies:
   - They examine the transferability of delta tuning methods across different downstream tasks to see how well methods adapted for one task perform on others.

# Combinations of Delta Tuning Methods

Sequential Combination Results:

- Conducted by splitting the tuning process into three stages, each optimizing a different method while freezing previous ones.
- Tested on RoBERTaLARGE with the SST-2 task.
- Found that while performance could improve with subsequent delta tuning methods, no optimal sequential combination emerged consistently across settings.

Generalization Gap Analysis:

- Delta tuning methods showed smaller generalization gaps compared to full fine-tuning, indicating less overfitting.
- Combining delta methods enlarged the generalization gap to levels comparable with fine-tuning, suggesting effective memorization with fewer parameters.
- Manual templates did not significantly affect the generalization gap.

# The Power of Scale for Delta Tuning

Innovative Delta Tuning Approaches:

- Last Layer Tuning: Optimizes the last encoder layer of T5, showing improved outcomes at larger scales.
- Selective Module Tuning: Random selection of modules for tuning enhances performance, especially in large-scale models.

Theoretical Insights and Implications:

- Larger PLMs with smaller intrinsic dimensionalities require fewer parameter adjustments for effective performance.
- Over-parameterization and comprehensive pre-training help prevent PLMs from getting stuck in local optima, speeding up convergence.

# Task-level Transferability Evaluation

Delta Tuning Methods Studied:

- Four methods: prompt tuning, prefix-tuning, adapter, and LoRA.
- Applied across 12 tasks within five different categories: sentiment analysis, natural language inference, paraphrase identification, question answering, and summarization.

Findings:

- Performance is measured by the ratio of zero-shot transferring performance to the original performance on the training task.
- <u>Within Same Task Category</u>: *Good performance* when transferring delta parameters among tasks of the same category (e.g., from one sentiment analysis task to another).
- <u>Across Different Task Types</u>: Generally *poor performance* when transferring parameters among tasks of different types (e.g., from sentiment analysis to paraphrase identification).
- Notable **exception** where parameters trained on text generation tasks (like question answering and summarization) *transfer effectively to sentiment analysis tasks*. This suggests that text generation tasks may encapsulate broader linguistic knowledge useful for other types of tasks.

# Task-level Transferability Evaluation

- The findings support the notion of a common subspace among various tasks, as previously introduced.
- Demonstrates promising potential for utilizing trained delta parameters for knowledge transfer across similar tasks, enhancing the utility of delta tuning methods in diverse applications.