A Tutorial On Parameter Efficient Fine-tuning

The 7th International Conference on Computational Intelligence and Networks (KIIT),

March 13, 2026

Krishna Garg

About Me

- Postdoctoral Research Assistant, Department of Engineering Science, University of Oxford
  - Research Focus: **Medical AI and AI for Science**
- PhD and MS, Department of Computer Science, University of Illinois Chicago
  - Research Focus: **Natural Language Processing, Deep Learning**
- B.E. Hons, Computer Science, Bits Pilani, Pilani Campus
- Industry Experience:
  - Adobe Research, Bengaluru
  - Samsung Research Institute, Noida
- Webpage: kgarg8.github.io

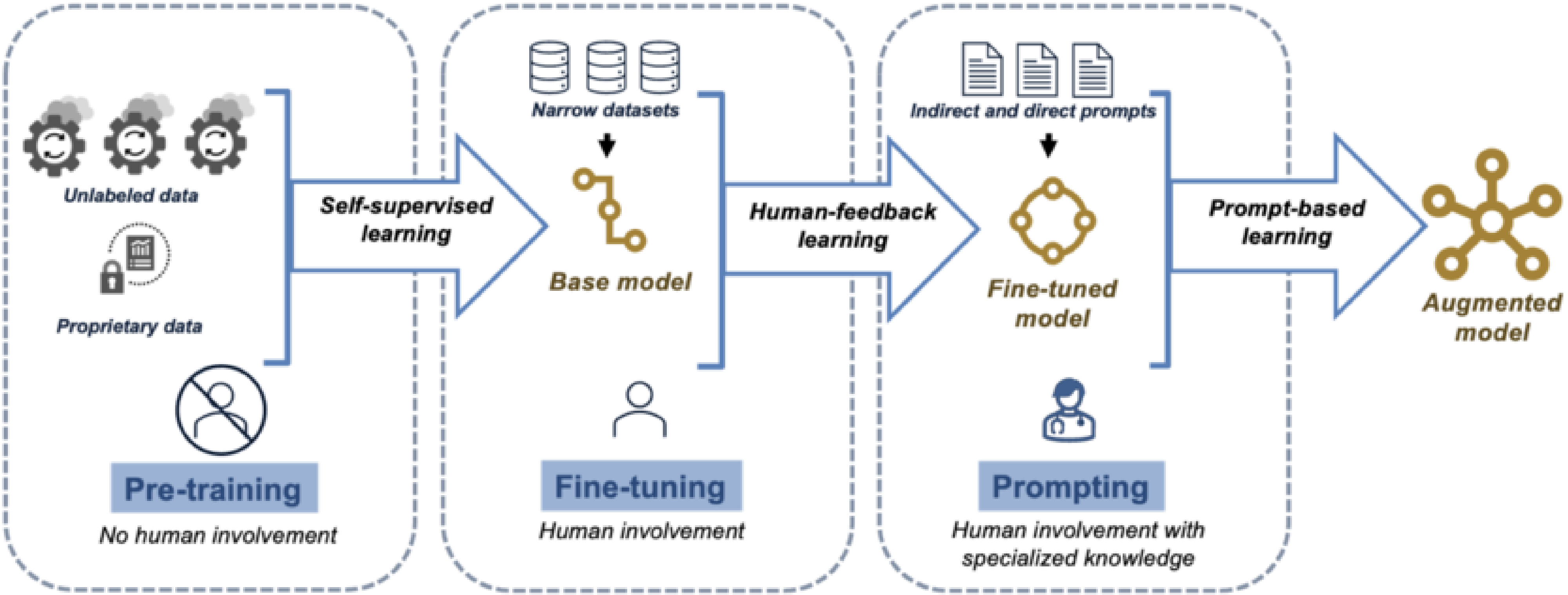
Outline

- Motivation for fine-tuning LLMs
- Motivation for PEFT
- Methods for PEFT
- Hands-on with Jupyter Notebook
- Conclusion

Outline

- **Motivation for fine-tuning LLMs**
- Motivation for PEFT
- Methods for PEFT
- Hands-on with Jupyter Notebook
- Conclusion

The three ways to adapt LLMs: Pre-training, Fine-tuning, Prompting



https://www.linkedin.com/pulse/intro-pre-trainingfine-tuning-llms-part-1-esmail-a-gumaan-wdiff/

Why do we fine-tune LLMs?

Pre-training
- Learns general knowledge (e.g., English)
- Very large compute (weeks, thousands of GPUs)
- Massive unlabeled web-scale data

Fine-tuning
- Specialized knowledge (e.g., medicine, law)
- Much smaller compute (hours–days, few GPUs)
- Smaller labeled or domain-specific dataset

https://in.pinterest.com/pin/png-clipart-free-download-largest-transparent-clip-art-images-collection-clipartmax--781093747779661570/
https://www.lymmhigh.org.uk/careers-education-information-and-guidance/

Outline

- Motivation for fine-tuning LLMs
- **Motivation for PEFT**
- Methods for PEFT
- Hands-on with Jupyter Notebook
- Conclusion

Motivating Example

- **Example: 100B Parameter Model (fp16)**
- Weights: ~200 GB
- Gradients: ~200 GB
- Adam optimizer states: ~400 GB
- **Total training memory ≈ 800 GB**

- **Fp4 -> 50GB (Quantization) + LoRA -> QLoRA**

Motivating Example contd.

- **Memory Explosion**
  - Training requires:
    - ~800 GB memory for a single run
    - 8–16 high-end GPUs (80GB each)
- **Storage Explosion**
  - Suppose you fine-tune for 50 domains/ enterprise clients
    - 200 GB × 50 = **10 TB** Storage required (weights only)
    - Add optimizer states and checkpoints → even more

Motivating Example contd.

- **Slow Iteration & High Cost**
  - Each fine-tuning run takes hours to days and expensive GPU time
  - You cannot retrain every time when regulations change, new client is onboarded, data is updated
- **Catastrophic Forgetting**
  - Full weight updates may:
    - Overwrite pretrained knowledge
    - Reduce general capability
    - Make the model too narrow
  - Especially risky with small domain datasets

PEFT to the Rescue – updates only small percentage of parameters

Metric	**Full Fine-Tuning**	**PEFT (0.5%)**
Total Parameters Updated	100B	500M
% of Model Updated	100%	0.5%
Storage per Task	~200 GB	~1 GB
Training Memory Needed	~800 GB	Drastically reduced (only adapters trainable)
GPUs Required	8–16 high-end GPUs	1–4 GPUs (depending on setup)
Storage for 50 Tasks	~10 TB	~50 GB
Deployment Strategy	One full model per task	One shared base + small adapters
Risk of Forgetting	High	Lower (backbone frozen)
Scalability	Poor	High

Outline

- Motivation for fine-tuning LLMs
- Motivation for PEFT
- **Methods for PEFT**
- Hands-on with Jupyter Notebook
- Conclusion

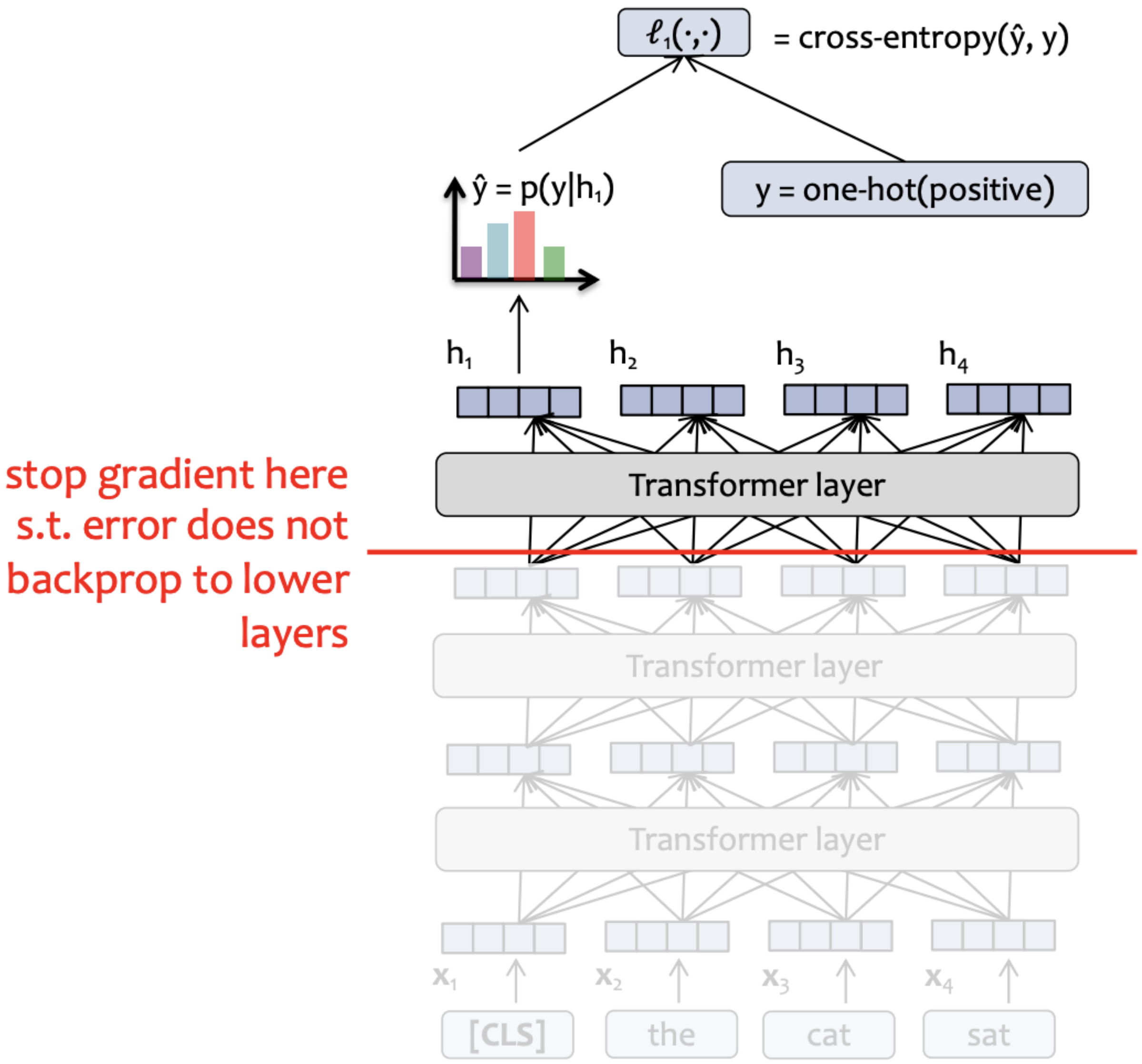
Finetune only last layer

Pros

- Very low compute and memory cost
- Simple to implement

Cons

- Limited adaptation capacity
- Cannot modify deeper representations



https://www.cs.cmu.edu/~mgormley/courses/10423-s25//slides/lecture10-peft.pdf

Adapters

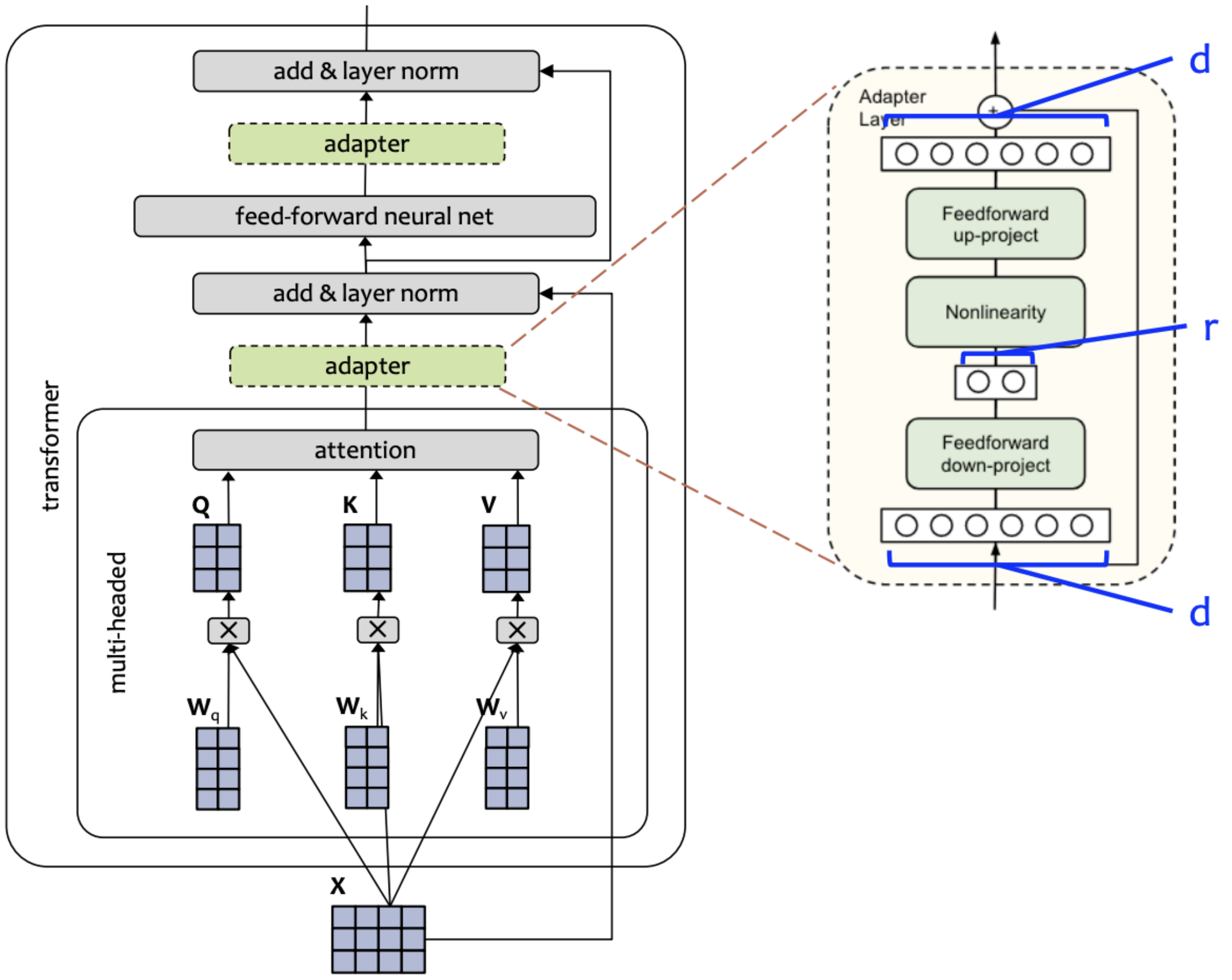
Insert small trainable modules between transformer layers while keeping the backbone unchanged.

Pros

- Good balance of efficiency and performance
- Separate small adapters per task

Cons

- Adds extra layers → slight inference overhead
- More architectural complexity



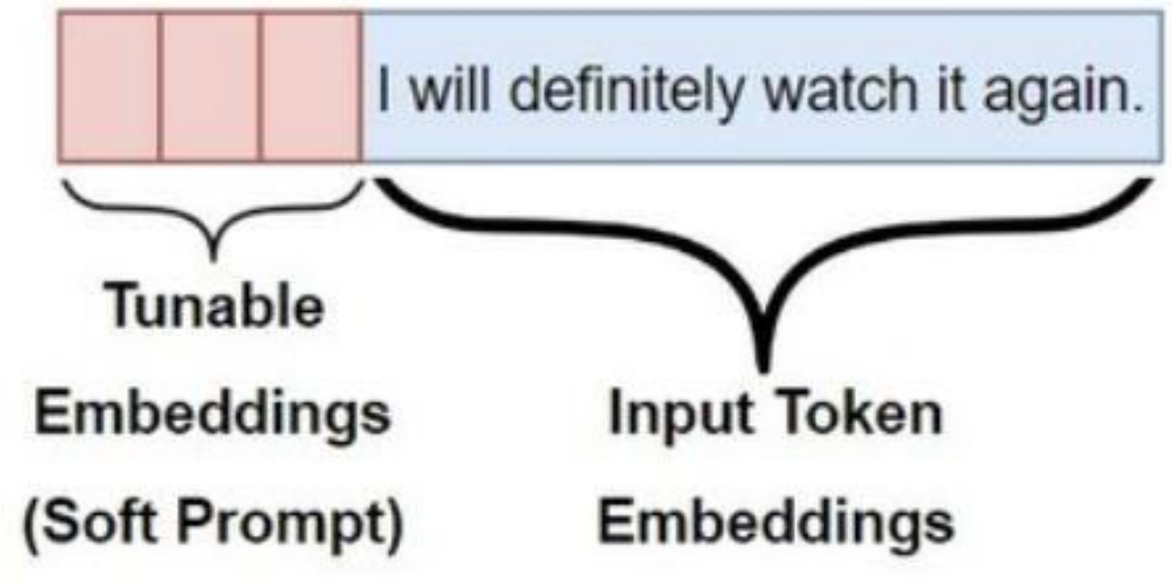
https://www.cs.cmu.edu/~mgormley/courses/10423-s25//slides/lecture10-peft.pdf

Prompt Tuning

Learn small trainable input embeddings (soft prompts) while keeping the main model frozen.

Pros
- Extremely lightweight
- No architecture changes

Cons
- Limited control for complex tasks
- Performance sensitive to prompt length



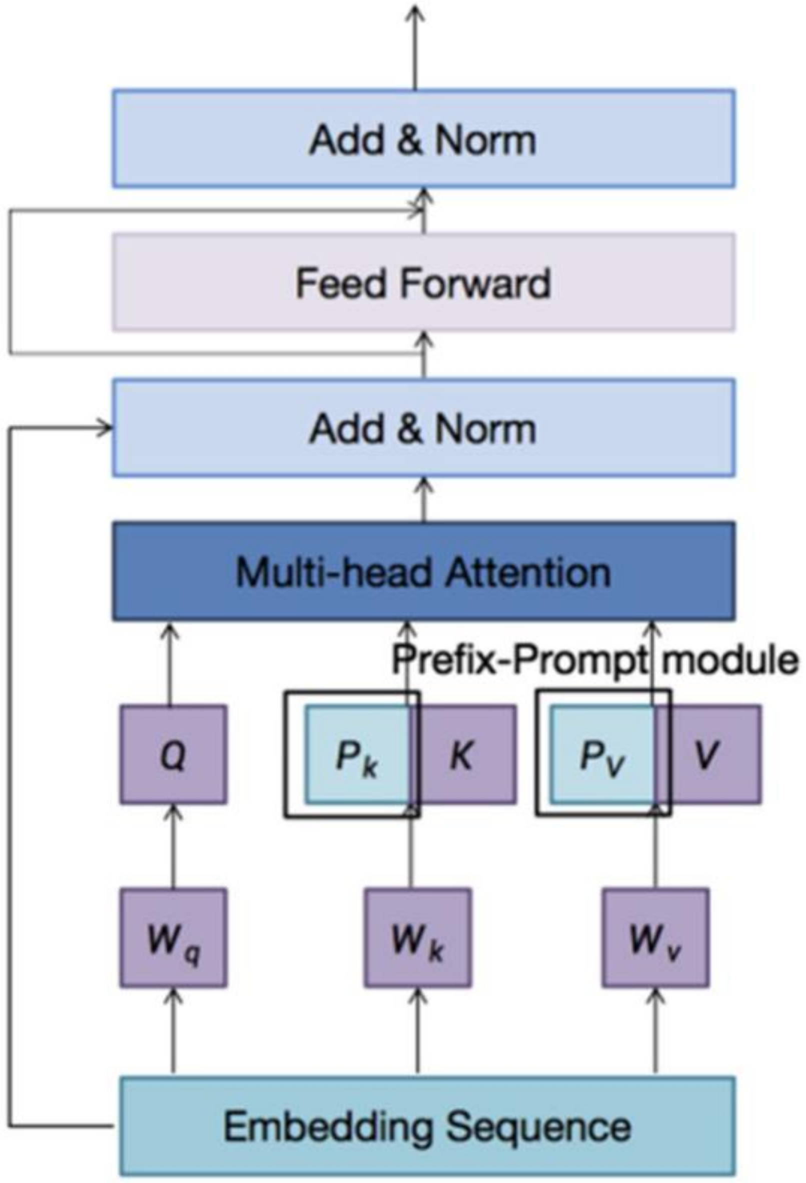
https://www.researchgate.net/figure/The-comparison-between-the-previous-T5-prompt-tuning-method-part-a-and-the-introduced_fig1_366062946

Prefix Tuning

Learn task-specific prefix vectors added to keys, values of each attention layer of a frozen model.

Pros
- Backbone fully frozen
- Fewer parameters than adapters

Cons
- Can be unstable to tune
- Lower expressivity than weight-based methods



https://www.mdpi.com/2076-3417/14/23/10772

Low Rank Adaptation (LoRA)

Add small low-rank trainable updates to selected weight matrices while freezing the original model.

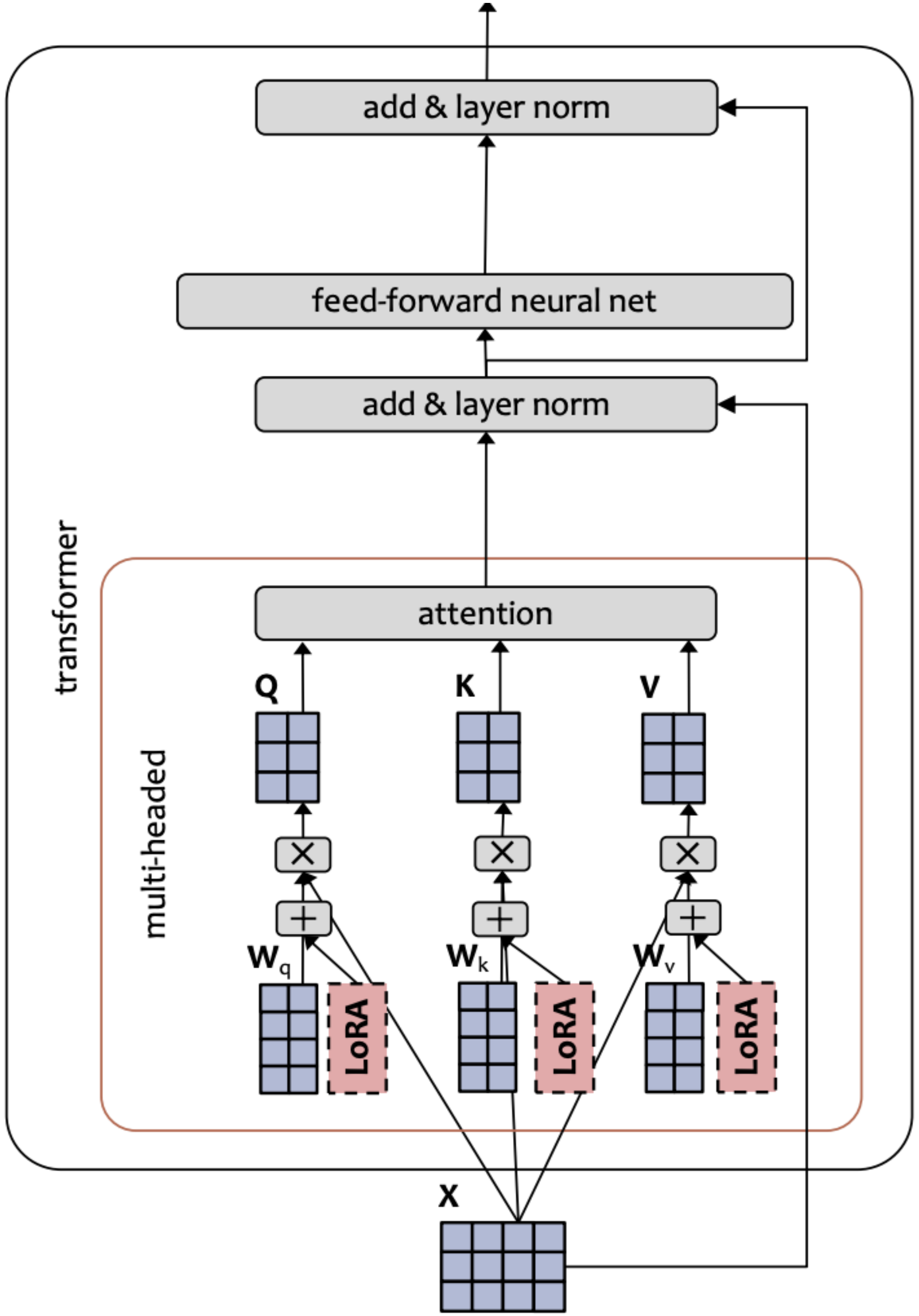
Pros

- Near full fine-tuning performance
- Very small number of trainable parameters

Cons

- Requires rank and module tuning
- Slight added complexity

LoRA could apply to every linear layer in the Transformer but the original paper applies it only to the attention weights

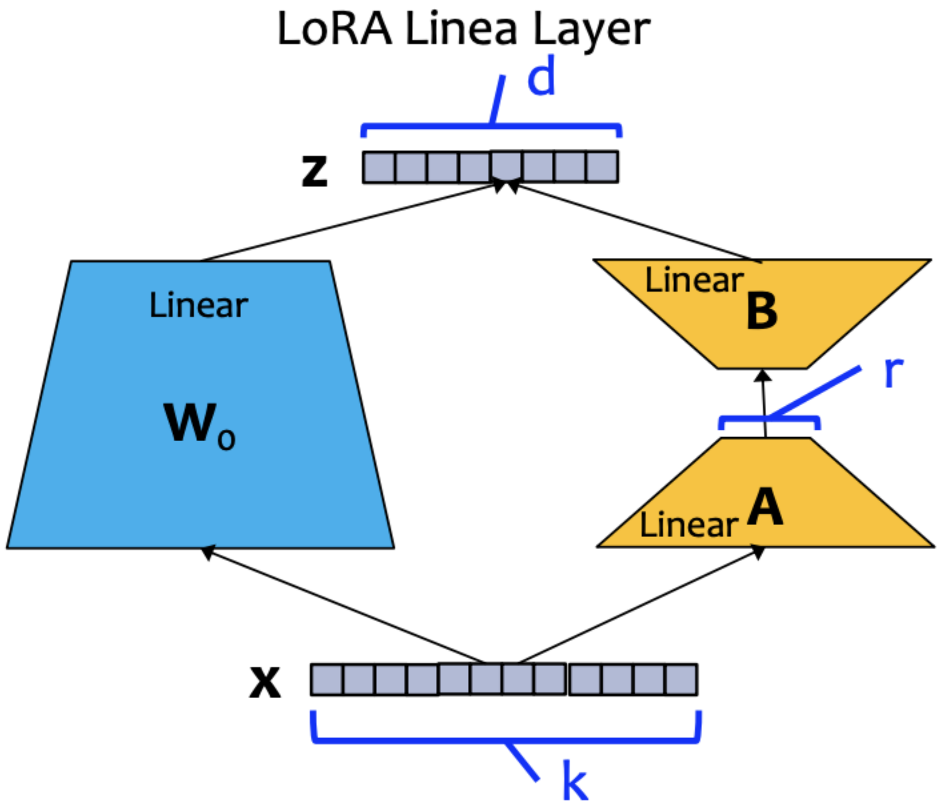


https://www.cs.cmu.edu/~mgormley/courses/10423-s25//slides/lecture10-peft.pdf

Low Rank Adaptation (LoRA)

Standard Layer: $z = W_0 x$

LoRA Layer: $z = (W_0 + BA)x, \quad r \ll \min(d, k)$



https://www.cs.cmu.edu/~mgormley/courses/10423-s25//slides/lecture10-peft.pdf

Outline

- Motivation for fine-tuning LLMs
- Motivation for PEFT
- Methods for PEFT
- **Hands-on with Jupyter Notebook**
- Conclusion

[Code Walkthrough with Jupyter Notebook]

Outline

- Motivation for fine-tuning LLMs
- Motivation for PEFT
- Methods for PEFT
- Hands-on with Jupyter Notebook
- **Conclusion**

Conclusion

- Fine-tuning LLMs is important for better performance on specialized domains
- Full fine-tuning LLMs in infeasible, so PEFT comes to the rescue
- PEFT could be implemented in multiple ways like Finetuning only last layer, Adapters, Prompt Tuning, Prefix Tuning, LoRA, QLoRA, etc.

References

- Code: https://colab.research.google.com/drive/1cMbEENo1KJhD13LO3hS6Eh5ry3az8yvv?usp=sharing
- https://llmsystem.github.io/llmsystem2024spring/assets/files/Group2-Presentation-cf8028bc58193a5e6e6d7b05709ef1a9.pdf
- https://www.cs.cmu.edu/~mgormley/courses/10423-s25//slides/lecture10-peft.pdf
- https://cseweb.ucsd.edu/classes/wi24/cse234-a/slides/CSE234-GuestLecture-SumanthHegde.pdf
- https://www.youtube.com/watch?v=7a6tZrnfVVE

Thank you!! Questions??