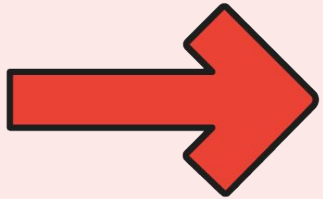
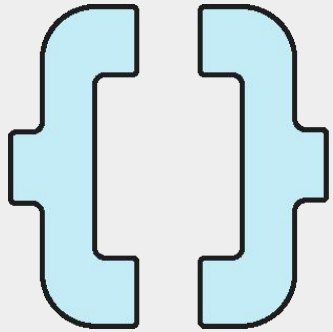
Google Developer Group
Chandigarh University

Gemini Study Jams

Session 2 - Supervised PEFT using LoRA



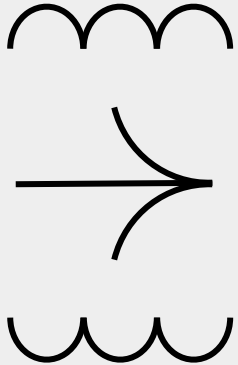
What will we do today?



1. The main loop of the session will be:

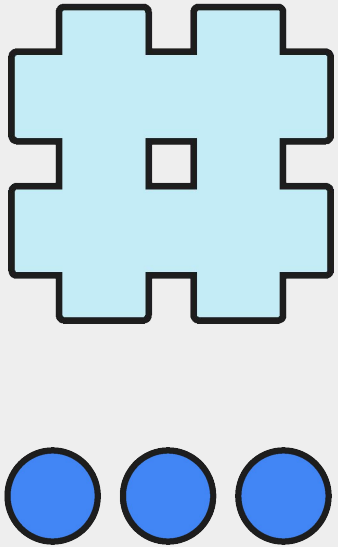
   Code -> Theory -> Code

2. We will work through the given code and stop and understand the theory behind it at **Exhibitions**.
3. It is **recommended** that you have the file “Session 2 - PEFT with LoRA.ipynb” open.



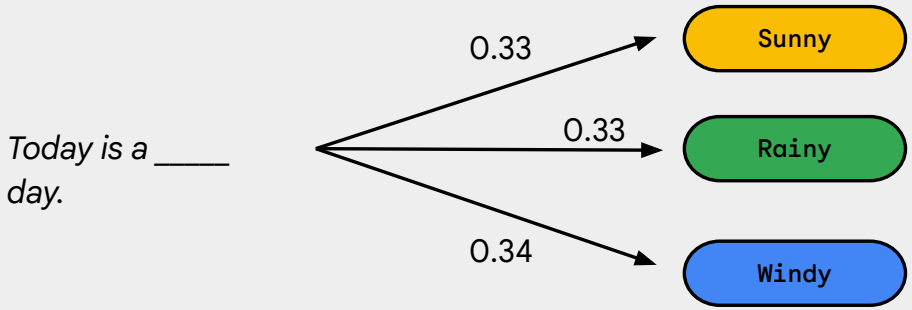
E1: LoRA decreases complexity!

1. During full fine-tuning, **the model is initialized to pre-trained weights Φ and updated to Φ + ΔΦ** by repeatedly following the gradient to maximize the conditional language modeling objective.
2. One of the main drawbacks for full fine-tuning is that **for each downstream task, we learn a different set of parameters** ΔΦ whose dimension |ΔΦ| equals |Φ|.
3. This means that for each downstream task, for a model like GPT3, **|Φ| ≈ 175 Billion**, storing and deploying many independent instances of fine-tuned models can be challenging, if at all feasible.



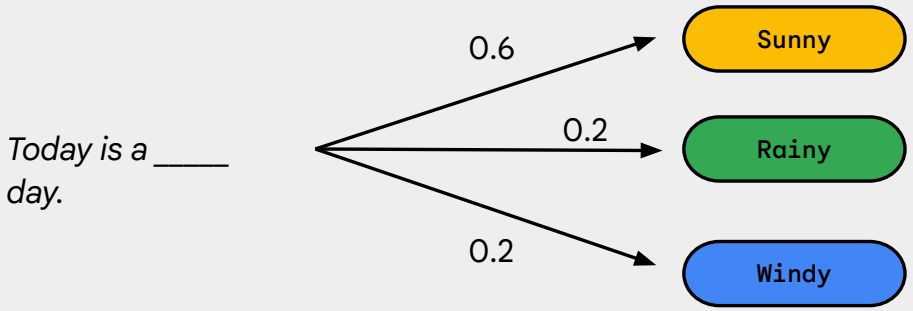
Overly Simplified Visualization : A Sunny Model

Pre-Trained Model



Φ = [0.33, 0.33, 0.34]

Fine Tuned Model



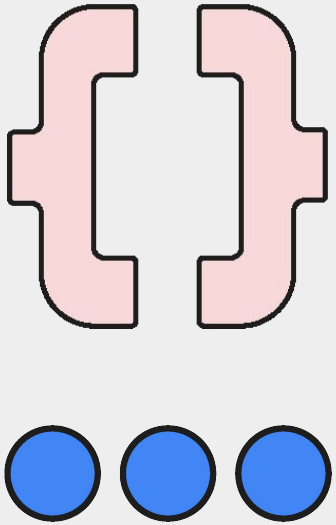
Φ = [0.6, 0.2, 0.2]

ΔΦ = [0.27, -0.13, -0.14]

Now imagine 175 billion parameter updates for GPT3 - unique to each task!

E1: LoRA decreases complexity!

1. In LoRA, the task-specific parameter increment $\Delta\Phi = \Delta\Phi(\Theta)$ is **further encoded by a much smaller-sized set of parameters** Θ with $|\Theta| << |\Phi|$.
2. When the pre-trained model is GPT-3 175B, **the number of trainable parameters $|\Theta|$ can be as small as 0.01% of $|\Phi|$**.



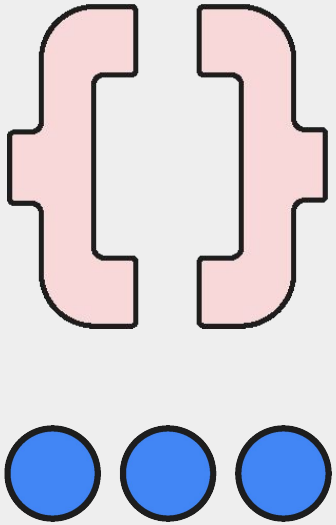
Now back to Code!

E2: How does LoRA work?

1. Consider a single Transformer layer, single attention head for clarity.
2. You start with an input sequence:

   $\mathbf{X \in R}$ where X has dimensions **T * d** where

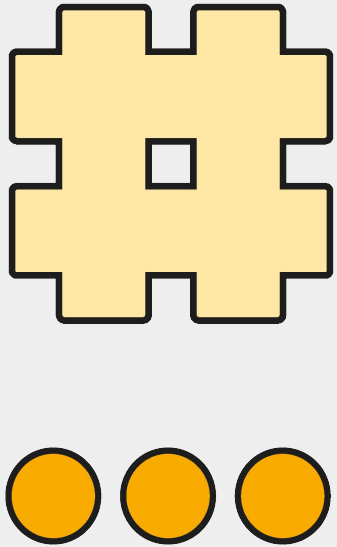
   - **T:** sequence length
   - **d:** embedding dimension
3. Each token is a row vector in X.



E2: How does LoRA work?

1. The attention mechanism does **not** operate directly on X.
2. Instead, it projects X into three different representation spaces:
- **Query (Q)** — “What am I looking for?”
- **Key (K)** — “What do I contain?”
- **Value (V)** — “What information do I give if selected?”
3. And then attention is computed using these three.
4. Mathematically:

Q = XW(Q) | V = XW(V) | K = XW(K)



E2: How does LoRA work?

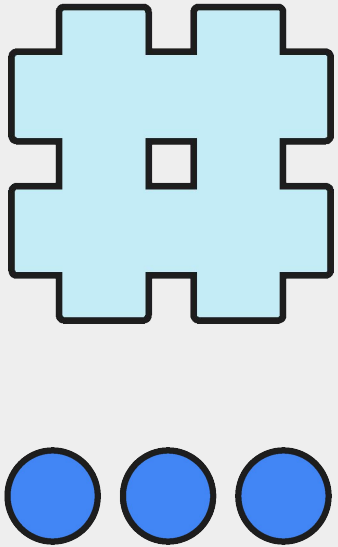
1. During Fine Tuning, the projection matrices are learned as follows:

   $Q = X(W(Q) + \Delta W(Q))$

2. LoRA **does not change** the attention mechanism.
3. It changes how the projection matrices are *learned*. LoRA enforces:

   $\Delta W(Q) = BA$

   Where A has dims **r * d** and B has dimension **d * r** where **r << d.** The original weight matrix W(Q) is frozen, and only A and B are trained.



E3: LoRA Evaluation

(a) Language modeling

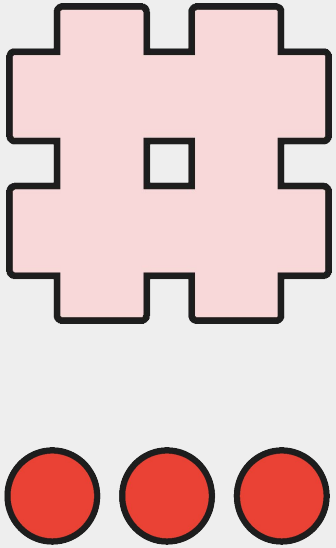
- Perplexity (PPL)
- Negative log-likelihood (NLL)

(b) Classification / structured prediction

- Accuracy
- Precision / Recall / F1
- ROC-AUC

(c) Generation tasks

- BLEU, ROUGE, METEOR (summarization, translation)
- Exact Match (EM), F1 (QA)
- Human evaluation (helpfulness, factuality, style adherence)



E4: Considerations

Choose target layers carefully (Q, V usually most effective)

Select rank r wisely: low r captures most gains; high r has diminishing returns

Tune scaling α to control update magnitude and stability

Initialize properly (base weights frozen; LoRA starts as no-op)

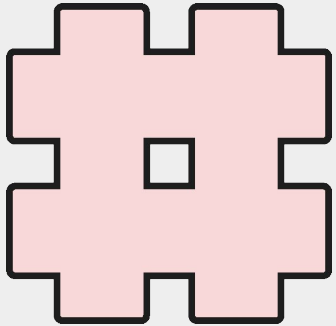
Use higher learning rates; avoid heavy regularization

Best for domain/style adaptation; limited for learning new capabilities

Check in-domain vs out-of-domain generalization

Decide whether to merge adapters at inference

Monitor memory, speed, and perplexity together



Quiz Time!

Doubts?

Ask away!

Google Developer Group
Chandigarh University

Thank You!

Fine Tuning Basics & Supervised Fine Tuning - End