Lecture 20 — Fine-tuning

GPT-3: In context learning.

But this was still doing "auto complete" aka "next token prediction"

Pre-trained models → Task-specific models.

How can we take a pretrained model and make it work for a task?

0. Pure prompting
1. Linear probing.
2. Full fine-tuning
3. LoRA.

Historical perspective

① How can we make better promptable models?

② How can we design better prompts?

→ How can we make a model follow instructions?

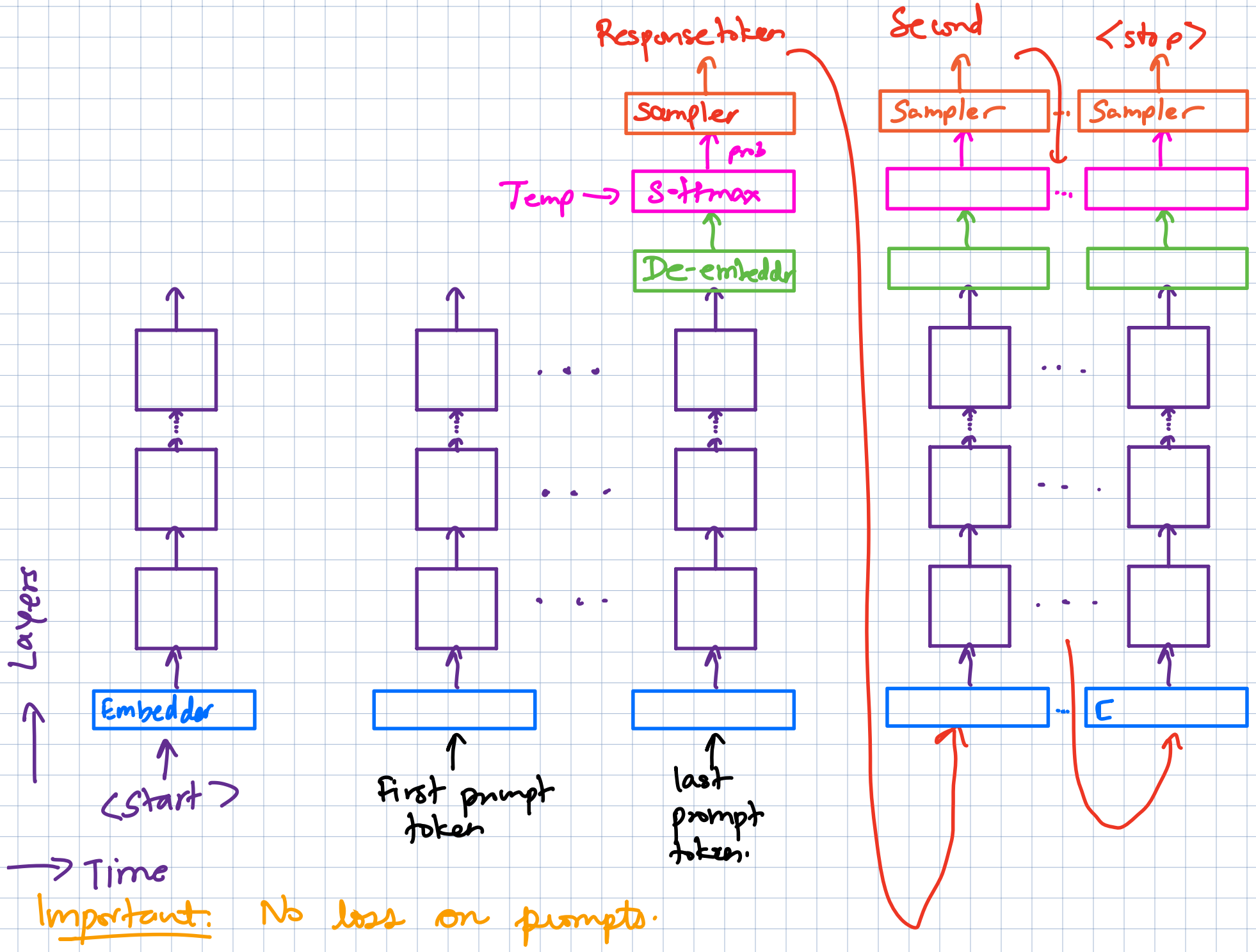
→ Rules – earlier AI models use rule based system. didn't work well.

→ Clever prompt — follow GPT-2
start writing the poem / document you want.
Possible, but hard for users

→ Train for it. (Instruct GPT)
→ provide reward when the task is done well
(i.e. instruction is followed)
→ Human reward is too expensive
→ Train model to act like human (RLHF)

→ Also supervised fine tuning
→ Train on Q & A pairs

→ How?
Feed in question, and once prompt ends, start generation
→ cross entropy loss



Important: No loss on prompts.

→ Took some time to figure out

→ Can also work with proper formatting as next token prediction.

Question Was it the data or the RL?

① SFT- alone. ALPACA story.

Feb 2023 Meta released LLAMA

How was data generated?

Instruction following example from Open AI model 52,000 demos

Hallucinations? —

② Human-written replication

Eleuther AI → Dolly

③ Vicuna trained on 70k ChatGPT conversations

→ SFT takes off

Training a new head.

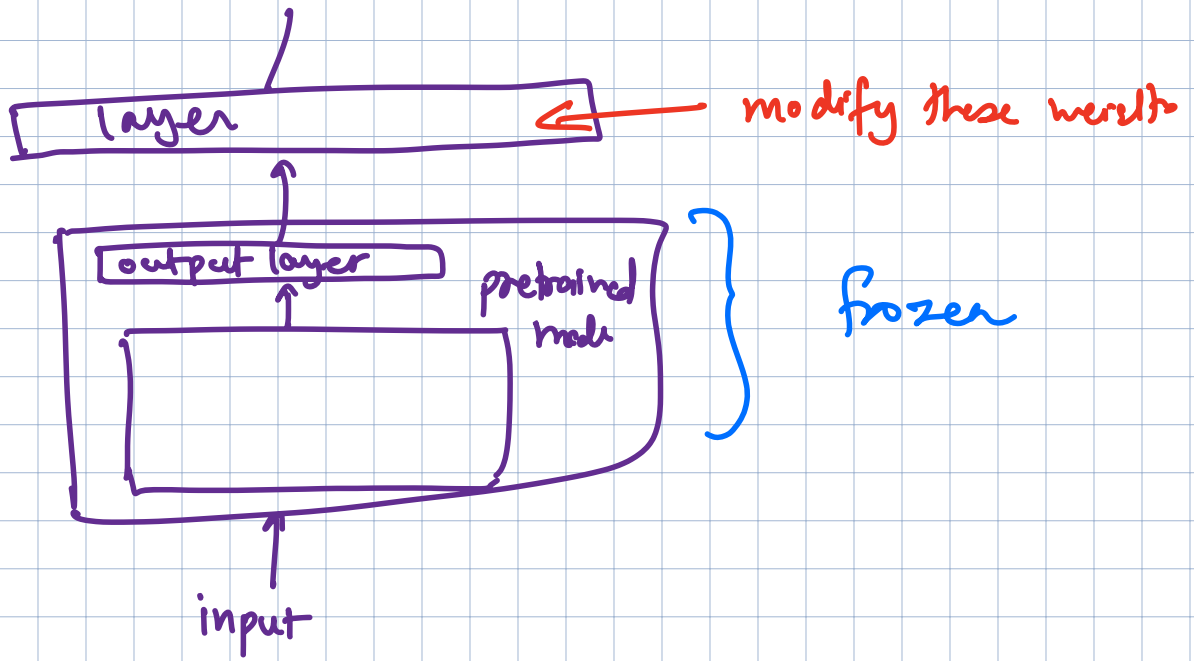
→ Sometimes during full fine tune
→ always linear probing

- Just train a new head / final linear layer
- Can sometimes work better than full fine tune, but not typically
- Much better in terms of compute.

→ Random initialization is bad.

→ Better: zero initialization

→ Better: Freeze weights and train just the head first.



Back to ICL

→ Why is ICL important?

→ New paradigm for ML

→ Experiments like Garg et al showed it works for synthetic tasks that we understand. eg. linear regression

→ not just in weights memorization

So what can we get with prompting + improved prompting?

→ Chat GPT ⇒ prompting became a craft

→ Can we have a disciplined search?

→ Idea: Split prompt into two parts

Pre-prompt + task details

↑ optimize this by running on examples.

Searching for good prompts

→ DSPy (Khattab et al) } Not gradient based

→ GEPA (Agarwal et al)

↳ Reflect on failures.

Can we optimize in a gradient sense?

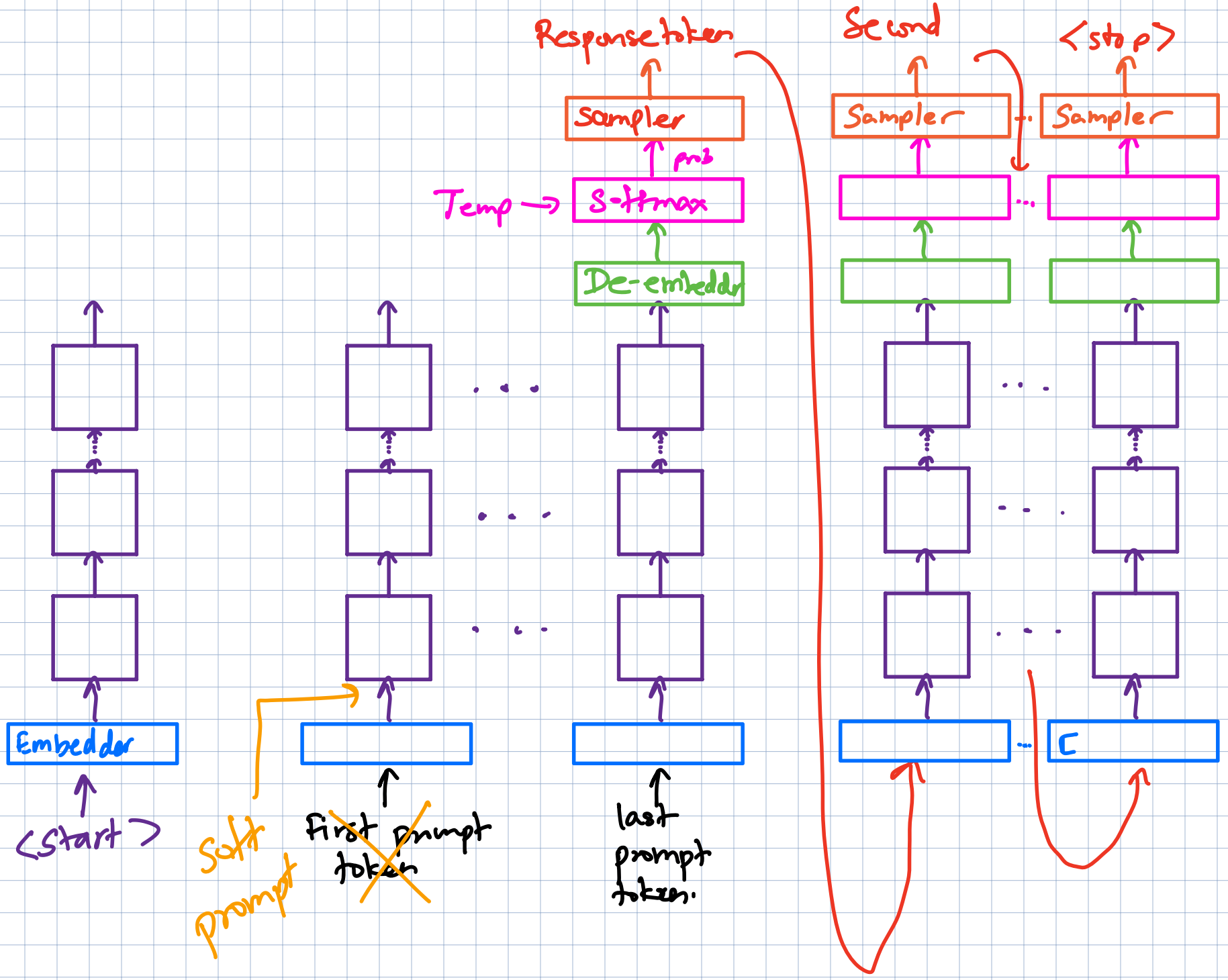
→ Idea: Fine-tune pre-prompt

Tokens are discrete → can't move them by a little bit :(

→ Use a person to edit → DSPy, GEPA automate this.

Soft prompting

Gradients stop at embedder. Ok. Work above it.


Response token
Second
<stop>
Sampler
Sampler
Sampler
pmf
Temp →
S-ttmax
De-embedder
Embedder
<Start>
Soft prompt
first prompt token
last prompt token.
C

How?

Create new special tokens <special>$_1$, <special>$_2$ ...

Say 100 such tokens

→ Make embedder columns for these

Much cheaper than full fine tune.

4096 Embedding

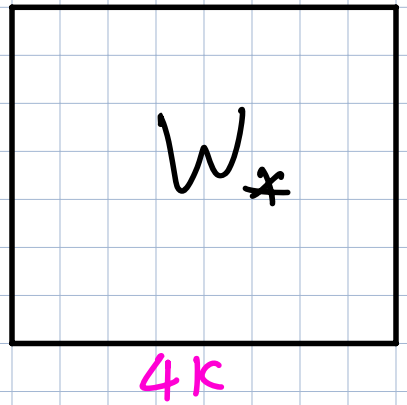
⇒ 4096 × 100 ≈ 40k params.

vs. 7B !

LoRA : Low Rank Adaptation.

Can we adapt weights of pretrained models with only few parameters?

MLPs or
e.g. K, V, Q matrices.
→ Not for embedding / deembedding / output layer

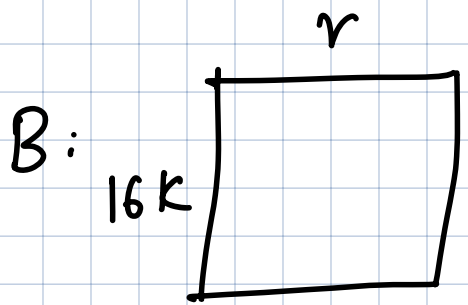


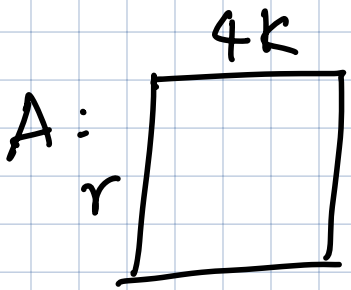
→ Total 64×10^6 parameters

Want: $W = W_* + \Delta W$

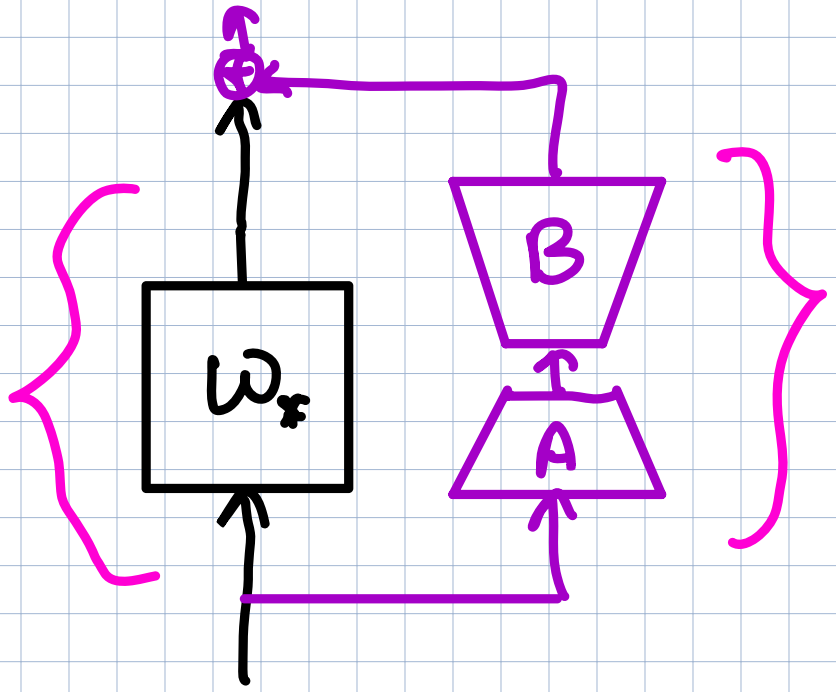
Low rank update → maybe only some directions are important

$\Delta W = B \cdot A$





Total parameters = $16r + 4r = 20r$ K



Initialization

0) All zero doesn't work

1) Both Xavier is bad

→ ΔW is too big

2) Make one zero

$B_0 = 0$, A_0 = non-zero

learning rate matters

→ Larger LR for B