

Low Rank Adaptation (LoRA)

A methodology for more resource efficient transfer learning.

George Rooney

Me

- Machine Learning Engineer at Fable, focusing on generative ML.
- Working in the NLP space for 7 years.
- Not a real picture of me.



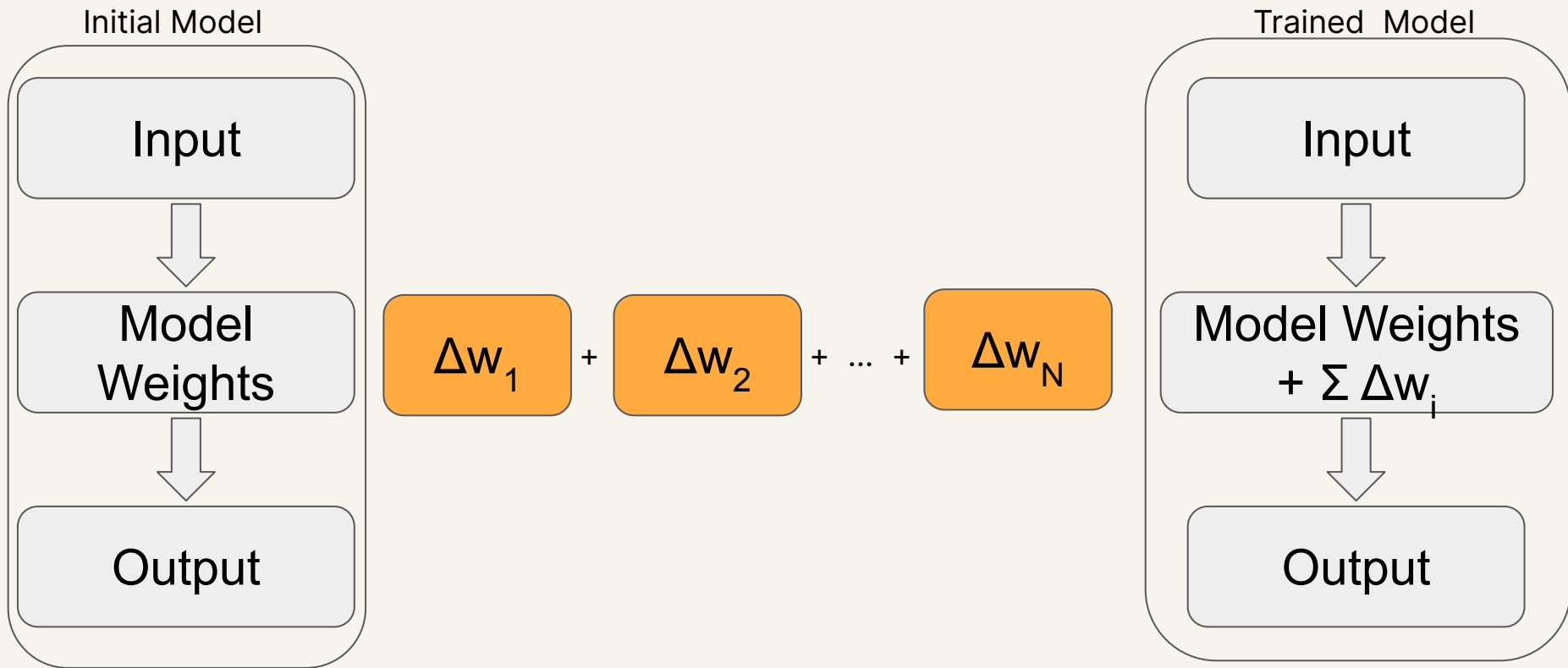
Who is LoRA for?

Anyone transfer learning from an existing model who wants to reduce their GPU compute requirements by two-thirds.

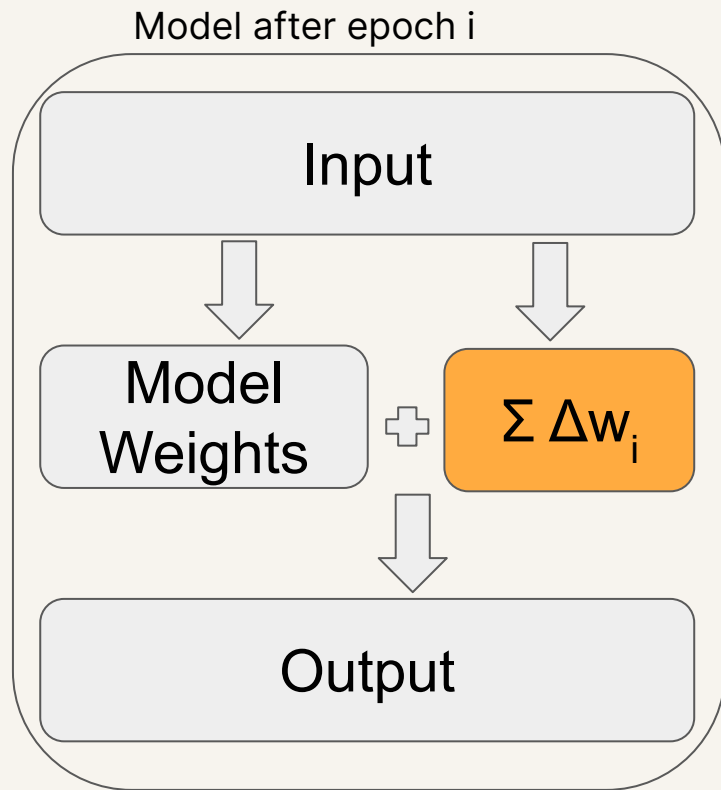
Anyone with multiple production models of the same base architecture.

Anyone who has had to explain to a PM the napkin math for why your organization can't train their own ChatGPT.

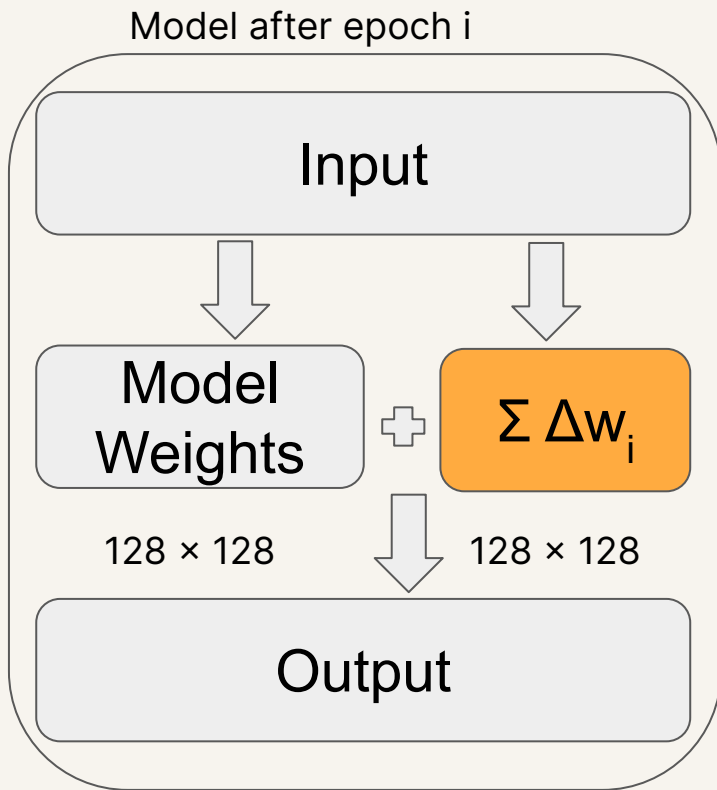
Transfer Learning Refresher



Reframing the Weight Updates



Now With (Some) Numbers

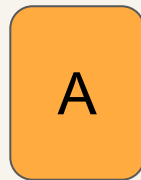


Matrix Decomposition Trick

- The rank of a matrix can be thought of as a measure of the amount of information that matrix contains.
- A matrix can be decomposed into the product of smaller matrices where the smallest dimension represents the rank of the original matrix.
- Research has demonstrated that for most LLM models we can get a reasonable approximation of a matrix with a very small rank, and transfer learning routines with ranks as small as 2.


$$\sum \Delta w_i$$

$$128 \times 128 = 16,384 \text{ Parameters}$$

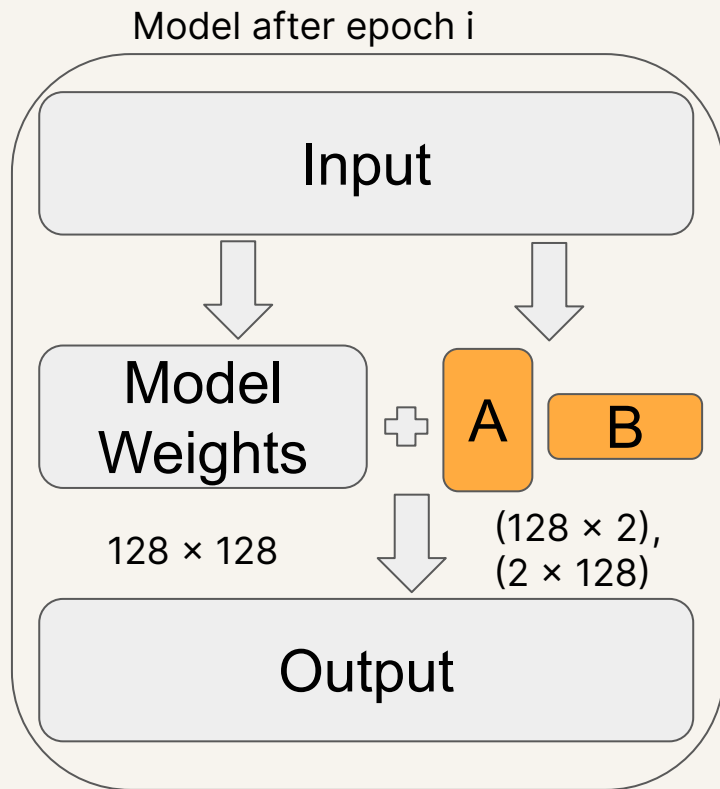

$$A$$


$$B$$

$$128 \times 2 + 2 \times 128 = 512 \text{ parameters}$$

Reality

- In LLM applications these LoRA networks generally represent <1 % of model weights.
- We hold the entire model in GPU memory for the forward pass, but only need gradient and momentum information for an extra 1% of the model. This corresponds to a nearly 2/3 reduction in memory requirements.



Inference

- During inference the low rank matrix approximations can be expanded and added in to model weights for 0 latency.
- If you have multiple models (one per user, department, task) that share the same base model, you can just swap out the weights for the approximated matrices which reduces model load time by over 99%.

