Alberto Andrés Valdés González.
Degree: Mathematical Engineer.
Work position: Data Scientist.
Mail: anvaldes@uc.cl/alberto.valdes.gonzalez.96@gmail.com
Location: Santiago, Chile.

LoRA

The inception of LoRA dates back to early 2021, following the release of GPT-3. Microsoft, in collaboration with OpenAI, faced the challenge of making large models like GPT-3 commercially viable.

They discovered that one-shot prompting alone was insufficient for achieving optimal performance in production, especially for complex tasks like translating natural language to code. This led to an exploration of fine-tuning methods that were cost-effective yet efficient.

The invention of LoRA is based on the product's need to enable fast, efficient, and cost-effective fine-tuning of large language models to enable domain specificity, efficient task switching, or user switching at run time.

The LoRA research paper was published in October 2021 by a team of researchers from Microsoft.

Why do we need LoRA?

Let's understand the problem before we even understand what LoRa is and how it solves it.

Large language models like GPT-4, Claude 2, LLaMA 70b, etc., are great, but they are very generic and large. To adopt these large language models for specific domains like healthcare or banking or for specific tasks like converting text into code, we need something called fine-tuning.

Fine-tuning is the process of training a pre-trained model on a specific, smaller dataset to specialize its performance on a particular task or domain. As the models get larger (For example, GPT-3 has 175 billion parameters), full fine-tuning, which retrains all model parameters, becomes less feasible because of time, cost, and resources.

LoRA is a technique used for fine-tuning large models.

How LoRA Works?

At a high-level here is how LoRA works:

It keeps the original model unchanged and adds small, changeable parts to each layer of the model. This significantly reduces the trainable parameters of the model and reduces the GPU memory requirement for the training process, which is another significant challenge when it comes to fine-tuning or training large models.

For example, Full fine-tuning of the GPT-3 model will require us to train 175 billion parameters. Using LoRA, the trainable parameters for GPT-3 will be reduced roughly by 10,000 times and GPU memory requirements by three times.

In essence, LoRA solves these problems:

- Speed: Because less trainable parameters mean faster training.
- Compute Resources: Less trainable parameters mean less compute resources required for the training process, making it financially viable to fine-tune large models.
- Memory efficiency: Less trainable parameters mean we can cache them in memory, eliminating the need for disk reads, which are inefficient compared to reading from memory.

What is LoRA?

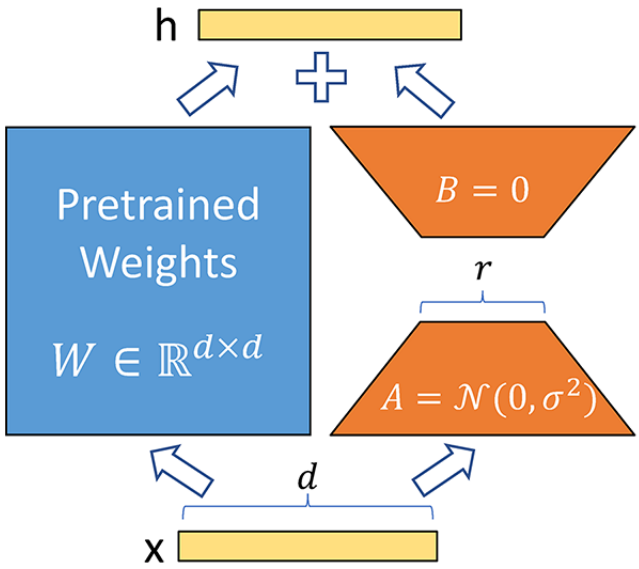
In very simple words, LoRA leverages the concept of lower-rank matrices to make the model training process extremely efficient and fast.

Large models have a lot of parameters. For example, GPT-3 has 175 billion parameters. These parameters are just numbers stored in matrices. Storing them requires a lot of storage.

Full fine-tuning means all the parameters will be trained, and this will require an extraordinary amount of compute resources that can easily cost in the millions of dollars for a model size like GPT.

Unlike traditional fine-tuning that requires adjusting the entire model, LoRA focuses on modifying a smaller subset of parameters (lower-rank matrices), thereby reducing computational and memory overhead.

LoRA is built on the understanding that large models inherently possess a low-dimensional structure. By leveraging low-rank matrices, LoRA adapts these models effectively. This method focuses on the core concept that significant model changes can be represented with fewer parameters, thus making the adaptation process more efficient.



Stable Difussion:

The LoRA technique is also widely adopted in image models like Stable Diffusion.

The idea is pretty much the same as language models. Instead of fully fine-tuning large models like Stable Diffusion, we only train lower-rank matrices on small datasets.

In the case of language models, the goal is domain specificity. For image models, the most obvious use case is to adopt a style or a consistent character when generating images.

These lower-rank matrices are known as adapters; they are very small in size, and there are thousands of them on the internet that you can download and put on top of your base Stable Diffusion model and generate style-specific images.

Some common uses of LoRA Stable Diffusion models are:

- Style specialization: Anime style, oil painting style, etc.
- Character specialization: Generate images of Mario or SpongeBob reliably.
- Quality Improvements: More detail, better faces, etc.

You can combine multiple LoRAs to get outputs reflecting multiple specializations.


Upgraded Model (SDXL)
X-Adapter
Pretrained Plugins for Stable Diffusion 1.5
(Condition or Style Examples)
LoRA:shukezouma
Instructpix2pix
Finetuned SD1.5
ControlNet-Depth
ControlNet-tile
T2IAdapter-Canny