14.04.2024 15:31 Easily Train a Specialized LLM: PEFT, LoRA, QLoRA, LLaMA-Adapter, and More

Easily Train a Specialized LLM: PEFT, LoRA, QLoRA, LLaMA-Adapter, and More

Training a specialized LLM over your own data is easier than you think...

**NOV 27, 2023**



**77**  **11**  **8** [**Share**](javascript:void(0))



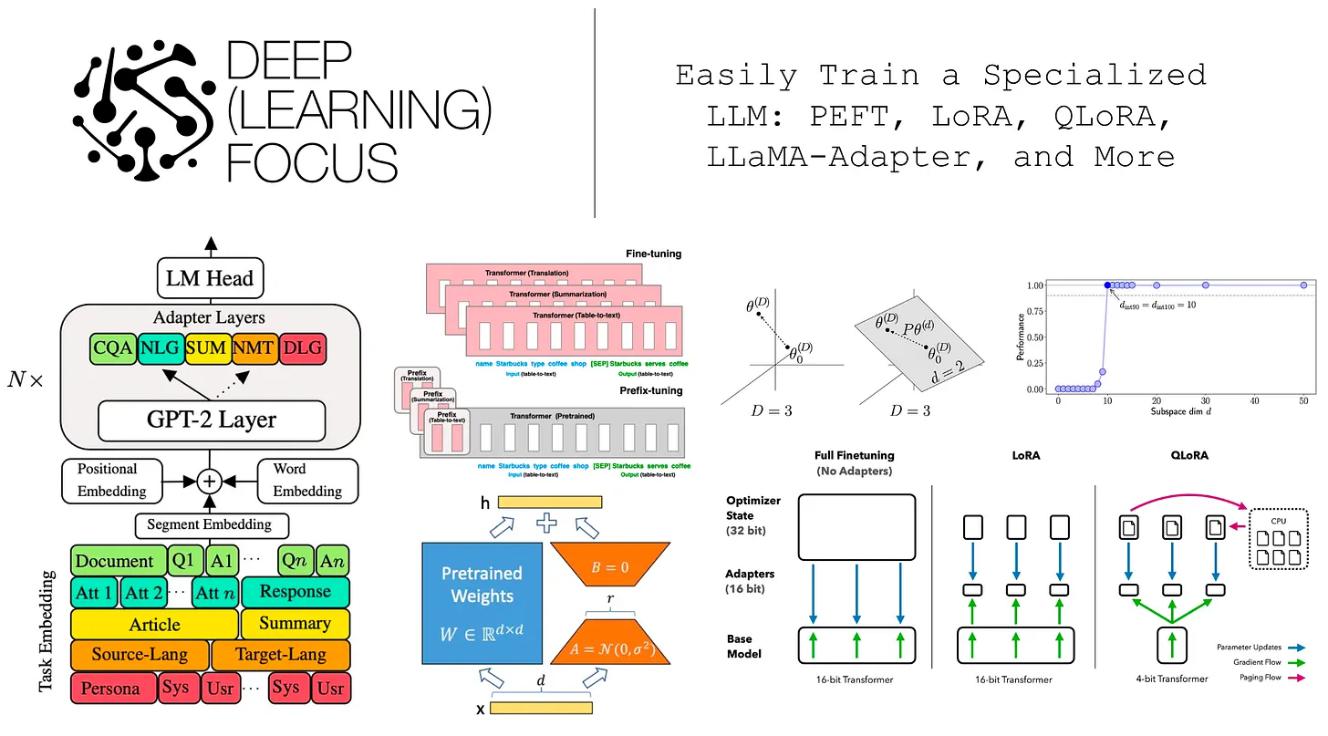
This newsletter is presented by [Rebuy](https://www.rebuyengine.com/), the commerce AI company.

Join subscribers from Microsoft, Tesla, Google, Meta, and more that use Deep (Learning) Focus to better understand AI research!

If you like the newsletter, feel free to [get in touch with me](https://cameronrwolfe.me/) or follow me on [Medium](https://medium.com/@wolfecameron), [X](https://twitter.com/cwolferesearch), and [LinkedIn](https://www.linkedin.com/in/cameron-r-wolfe-ph-d-04744a238/). I try my best to produce useful/informative content.

https://cameronrwolfe.substack.com/p/easily-train-a-specialized-llm-peft 1/46

14.04.2024 15:31 Easily Train a Specialized LLM: PEFT, LoRA, QLoRA, LLaMA-Adapter, and More



[(from [1, 3, 6, 15, 19])](https://substackcdn.com/image/fetch/f_auto,q_auto:good,fl_progressive:steep/https%3A%2F%2Fsubstack-post-media.s3.amazonaws.com%2Fpublic%2Fimages%2Fd8867fda-7270-4990-9710-0c7cf2e13878_2354x1304.png)

[Due to the surge of interest in large language models (LLMs), AI practitioners are](https://substackcdn.com/image/fetch/f_auto,q_auto:good,fl_progressive:steep/https%3A%2F%2Fsubstack-post-media.s3.amazonaws.com%2Fpublic%2Fimages%2Fd8867fda-7270-4990-9710-0c7cf2e13878_2354x1304.png) [commonly asked questions such as: ***How can we train a specialized LLM over our own***](https://substackcdn.com/image/fetch/f_auto,q_auto:good,fl_progressive:steep/https%3A%2F%2Fsubstack-post-media.s3.amazonaws.com%2Fpublic%2Fimages%2Fd8867fda-7270-4990-9710-0c7cf2e13878_2354x1304.png)[***data?*** However, answering this question is far from simple. Recent advances in](https://substackcdn.com/image/fetch/f_auto,q_auto:good,fl_progressive:steep/https%3A%2F%2Fsubstack-post-media.s3.amazonaws.com%2Fpublic%2Fimages%2Fd8867fda-7270-4990-9710-0c7cf2e13878_2354x1304.png)[generative AI are powered by massive models with many parameters, and training](https://substackcdn.com/image/fetch/f_auto,q_auto:good,fl_progressive:steep/https%3A%2F%2Fsubstack-post-media.s3.amazonaws.com%2Fpublic%2Fimages%2Fd8867fda-7270-4990-9710-0c7cf2e13878_2354x1304.png) [such an LLM requires expensive hardware (i.e., many expensive GPUs with a lot of](https://substackcdn.com/image/fetch/f_auto,q_auto:good,fl_progressive:steep/https%3A%2F%2Fsubstack-post-media.s3.amazonaws.com%2Fpublic%2Fimages%2Fd8867fda-7270-4990-9710-0c7cf2e13878_2354x1304.png) [memory) and fancy training techniques (e.g., fully-sharded data parallel training).](https://substackcdn.com/image/fetch/f_auto,q_auto:good,fl_progressive:steep/https%3A%2F%2Fsubstack-post-media.s3.amazonaws.com%2Fpublic%2Fimages%2Fd8867fda-7270-4990-9710-0c7cf2e13878_2354x1304.png) [Luckily, these models are usually trained in two phases—***pretraining*** and ***fnetuning***](https://substackcdn.com/image/fetch/f_auto,q_auto:good,fl_progressive:steep/https%3A%2F%2Fsubstack-post-media.s3.amazonaws.com%2Fpublic%2Fimages%2Fd8867fda-7270-4990-9710-0c7cf2e13878_2354x1304.png) —where the former phase is (much) more expensive. Given that high-quality pretrained LLMs are readily available online, most AI practitioners can simply download a pretrained model and focus upon adapting this model (via fnetuning) to their desired task.

***“Fine-tuning enormous language models is prohibitively expensive in terms of the hardware required and the storage/switching cost for hosting independent instances for diferent tasks.”*** - from [1]



Nonetheless, the size of the model does not change during fnetuning! As a result, fnetuning an LLM—***though cheaper than pretraining***—is not easy. We still need

https://cameronrwolfe.substack.com/p/easily-train-a-specialized-llm-peft 2/46

14.04.2024 15:31 Easily Train a Specialized LLM: PEFT, LoRA, QLoRA, LLaMA-Adapter, and More

training techniques and hardware than can handle such a model. Plus, and every fnetuning run creates an entirely separate “copy” of the LLM that we must store, maintain, and deploy—***this can quickly become both complicated and expensive***!

**How do we fx this?** Within this overview, we will learn about a popular solution to the issues outlined above—***parameter-efcient fnetuning***. Instead of training the full model end-to-end, parameter-efcient fnetuning leaves pretrained model weights fxed and only adapts a small number of task-specifc parameters during fnetuning. Such an approach drastically reduces memory overhead, simplifes the storage/deployment process, and allows us to fnetune LLMs with more accessible hardware. Although the overview will include a many techniques (e.g., prefx tuning and adapter layers), our focus will be upon Low-Rank Adaptation (LoRA) [1], a simple and widely-used approach for efciently fnetuning LLMs.

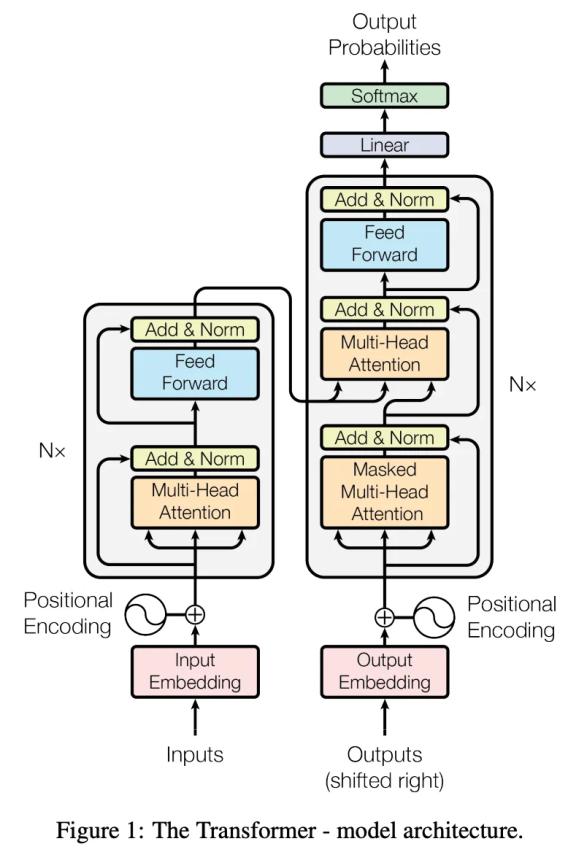
Background Information

Before diving into LoRA and its (many) variants, we need to go over some background information that’s necessary for understanding the rest of the overview. Given that this writeup is already quite long, we’ll keep this section brief and provide links to further reading for those who are less familiar.

The Structure of a Language Model

https://cameronrwolfe.substack.com/p/easily-train-a-specialized-llm-peft 3/46

14.04.2024 15:31 Easily Train a Specialized LLM: PEFT, LoRA, QLoRA, LLaMA-Adapter, and More



[(from [25])](https://substackcdn.com/image/fetch/f_auto,q_auto:good,fl_progressive:steep/https%3A%2F%2Fsubstack-post-media.s3.amazonaws.com%2Fpublic%2Fimages%2F7bfba492-5e13-4883-bb7a-fa85c3ec810e_778x1168.png)

[The frst step in understanding language models is developing a solid grasp of the](https://substackcdn.com/image/fetch/f_auto,q_auto:good,fl_progressive:steep/https%3A%2F%2Fsubstack-post-media.s3.amazonaws.com%2Fpublic%2Fimages%2F7bfba492-5e13-4883-bb7a-fa85c3ec810e_778x1168.png) [architecture upon which these models are based—***the transformer architecture*** [25];](https://substackcdn.com/image/fetch/f_auto,q_auto:good,fl_progressive:steep/https%3A%2F%2Fsubstack-post-media.s3.amazonaws.com%2Fpublic%2Fimages%2F7bfba492-5e13-4883-bb7a-fa85c3ec810e_778x1168.png) [see above. The transformer architecture was originally proposed for Seq2Seq tasks](https://substackcdn.com/image/fetch/f_auto,q_auto:good,fl_progressive:steep/https%3A%2F%2Fsubstack-post-media.s3.amazonaws.com%2Fpublic%2Fimages%2F7bfba492-5e13-4883-bb7a-fa85c3ec810e_778x1168.png) [(e.g., summarization, translation, conditional generation, etc.) and contains both an](https://substackcdn.com/image/fetch/f_auto,q_auto:good,fl_progressive:steep/https%3A%2F%2Fsubstack-post-media.s3.amazonaws.com%2Fpublic%2Fimages%2F7bfba492-5e13-4883-bb7a-fa85c3ec810e_778x1168.png) [encoder and a decoder component.](https://substackcdn.com/image/fetch/f_auto,q_auto:good,fl_progressive:steep/https%3A%2F%2Fsubstack-post-media.s3.amazonaws.com%2Fpublic%2Fimages%2F7bfba492-5e13-4883-bb7a-fa85c3ec810e_778x1168.png)

[***Encoder***: each block has bidirectional self-attention and a feed-forward layer.](https://substackcdn.com/image/fetch/f_auto,q_auto:good,fl_progressive:steep/https%3A%2F%2Fsubstack-post-media.s3.amazonaws.com%2Fpublic%2Fimages%2F7bfba492-5e13-4883-bb7a-fa85c3ec810e_778x1168.png)



[***Decoder***: each block has causal self-attention, cross attention, and a feed-](https://substackcdn.com/image/fetch/f_auto,q_auto:good,fl_progressive:steep/https%3A%2F%2Fsubstack-post-media.s3.amazonaws.com%2Fpublic%2Fimages%2F7bfba492-5e13-4883-bb7a-fa85c3ec810e_778x1168.png)forward layer.

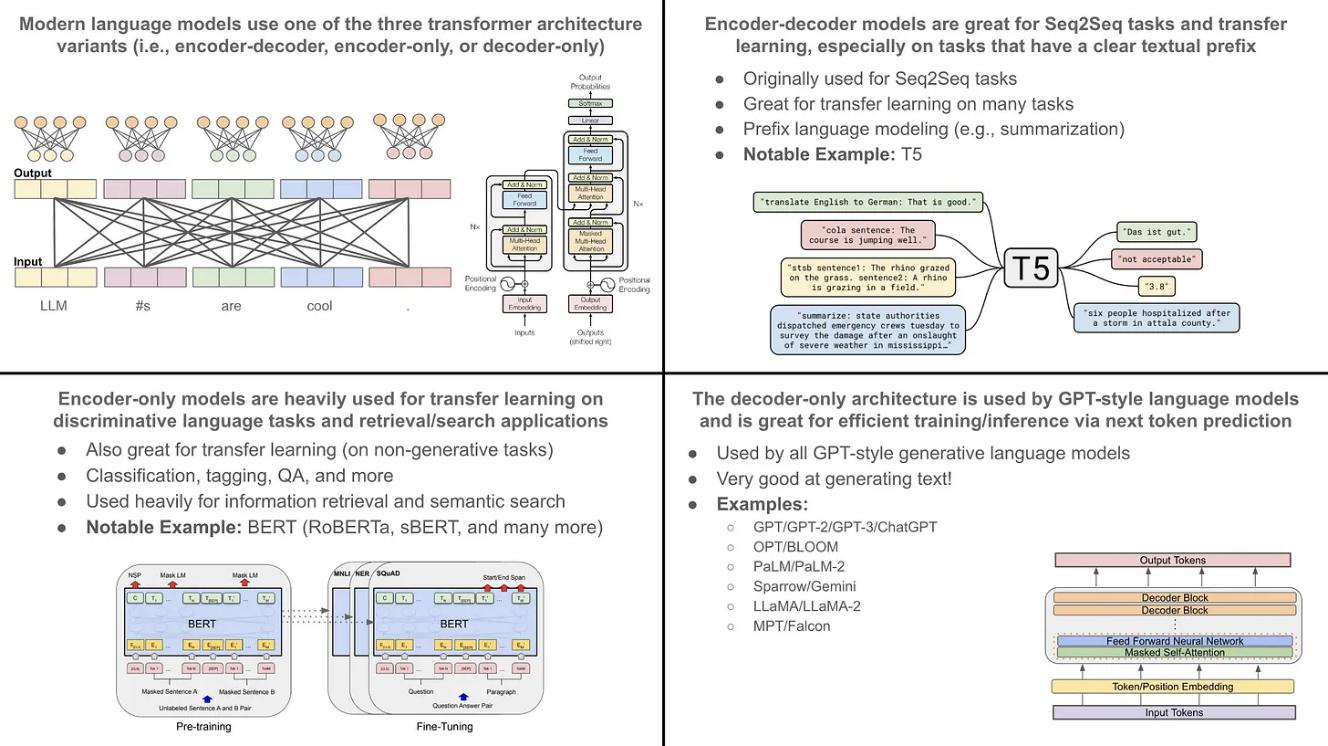


**Transformer variants.** The originally-proposed transformer architecture is usually referred to as an encoder-decoder transformer. However, two other popular variants of the transformer architecture exist: encoder-only and decoder-only transformers. These architectures are exactly as their name indicates—***they use only the encoder or decoder portion of the transformer architecture***. Notably, the decoder-only architecture removes not only the encoder, but also all of the cross attention

https://cameronrwolfe.substack.com/p/easily-train-a-specialized-llm-peft 4/46

14.04.2024 15:31 Easily Train a Specialized LLM: PEFT, LoRA, QLoRA, LLaMA-Adapter, and More

layers in the decoder, as there is no longer any encoder to which the cross attention layers can attend. A summary of these diferent architectural variants, along with popular examples of each, is provided in the table below.



Outline of the three major transformer architecture variants and their

primary use cases (from [9, 10, 25])

Although [GPT-style](https://cameronrwolfe.substack.com/p/language-models-gpt-and-gpt-2) generative LLMs [14] (i.e., large decoder-only transformers) are very popular today, many types of useful language models exist. Encoder-only models (e.g., BERT [10]) are used for discriminative language understanding tasks (e.g., classifcation or retrieval) 1, while encoder-decoder models (e.g., T5 [10]) are great for performing conditional generation (e.g., summarization, translation, text-to-SQL, etc.). Learn more about these models and their popular use cases below.

Language Understanding with BERT [[link](https://cameronrwolfe.substack.com/p/language-understanding-with-bert)]



T5: Text-to-Text Transformers [[Part One](https://cameronrwolfe.substack.com/p/t5-text-to-text-transformers-part)] [[Part Two](https://cameronrwolfe.substack.com/p/t5-text-to-text-transformers-part-354)]



**Transformer layer components.** Within each transformer layer, two primary transformations are used: self-attention and a feed-forward transformation. Additionally, diferent styles of self-attention—***either bidirectional or masked***—are

https://cameronrwolfe.substack.com/p/easily-train-a-specialized-llm-peft 5/46

14.04.2024 15:31 Easily Train a Specialized LLM: PEFT, LoRA, QLoRA, LLaMA-Adapter, and More

used within the decoder and encoder portions of the architecture. To learn more about these operations, check out the links below.

***Bidirectional Self-Attention*** [[link](https://x.com/cwolferesearch/status/1641932082283700226?s=20)]: used in the encoder



***Masked (Decoder) Self-Attention*** [[link](https://x.com/cwolferesearch/status/1644773244786941952?s=20)]: used in the decoder



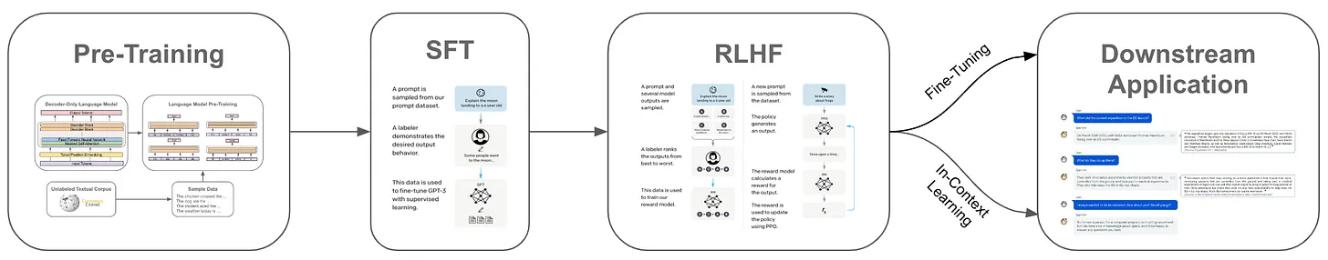
***Feed-Forward Layers*** [[link](https://cameronrwolfe.substack.com/i/94634004/feed-forward-neural-networks)]: used in both the encoder and the decoder



Additionally, encoder-decoder models have an extra [cross attention](https://sebastianraschka.com/blog/2023/self-attention-from-scratch.html#cross-attention) module within each block of the decoder following the masked self-attention.

**Putting it together.** Despite learning the concepts above, understanding how these pieces combined together within an LLM can be difcult. Check out the link below for more details on the structure and mechanics of language models.

How are LLMs trained?



The LLM training pipeline (from [12, 13])

Modern language models are trained in several steps. For example, encoder-only and encoder-decoder models are trained via a transfer learning approach that includes self-supervised pretraining and fnetuning on a downstream task, while generative (GPT-style) LLMs follow the multi-step training procedure shown in the fgure above. Within this discussion, we will mostly focus upon the training procedure of generative LLMs, which are the primary topic of this overview.

**Pretraining.** All language models are pretrained using some form of a self-supervised learning objective 2. For example, generative language models are

https://cameronrwolfe.substack.com/p/easily-train-a-specialized-llm-peft 6/46

14.04.2024 15:31 Easily Train a Specialized LLM: PEFT, LoRA, QLoRA, LLaMA-Adapter, and More

usually pretrained using a next token prediction objective, while encoder-only and encoder-decoder models commonly use a Cloze task. Read more below.

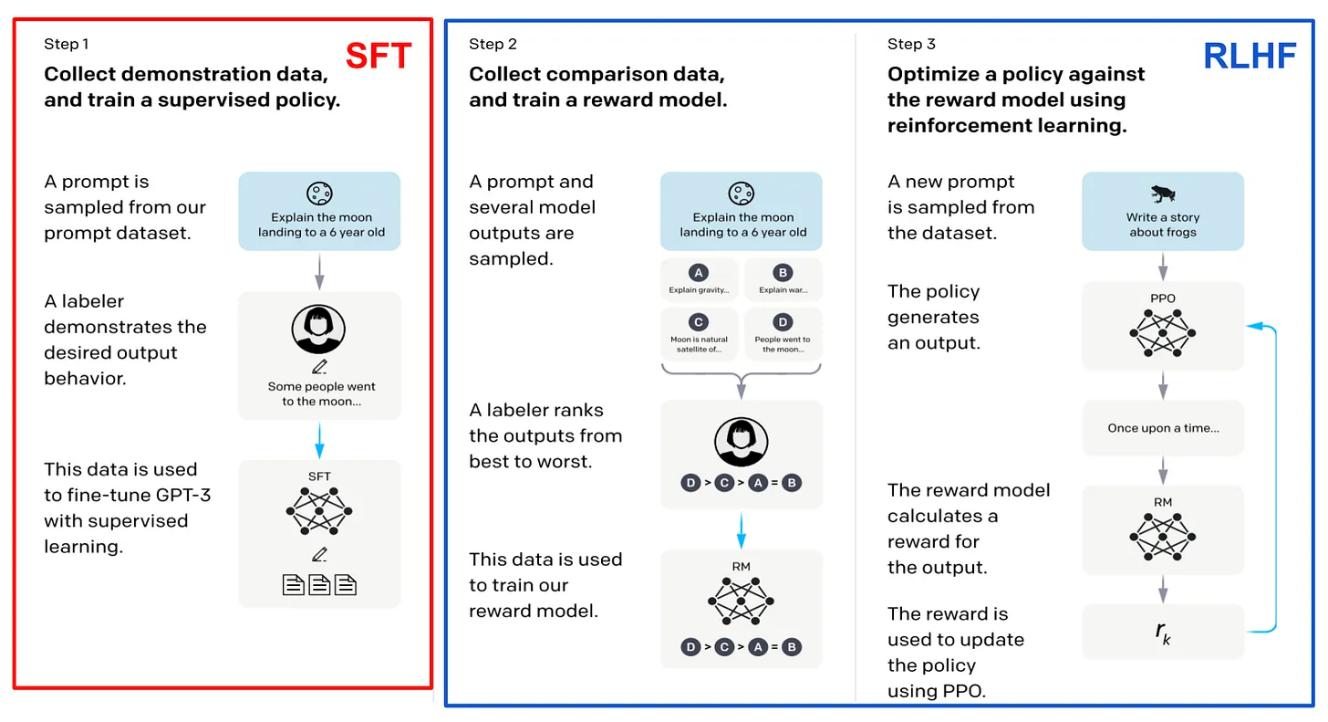
Pretraining with Next Token Prediction [[link](https://cameronrwolfe.substack.com/p/language-model-training-and-inference)]



Pretraining with Cloze (Masked Language Modeling) [[link](https://cameronrwolfe.substack.com/i/76273144/training-bert)]



Self-supervised learning techniques do not rely on manual human annotation—***the “labels” used for supervision are already present in the data itself***. For example, nexttoken prediction predicts the next word/token in a sequence of tokens sampled from a textual corpus (e.g., a book), while Cloze tasks mask and predict tokens in a sequence. We can collect massive datasets of unlabeled text (e.g., by scraping the internet) to use for self-supervised pretraining. Due to the scale of data available, the pretraining process is quite computationally expensive. So, we perform pretraining once and repeatedly use this same [foundation model](https://crfm.stanford.edu/) as a starting point for training a specialized model on many diferent tasks and applications.



(from [12])

**Alignment.** The alignment process, which is only applicable to generative LLMs, refers to the process of fnetuning a language model such that its output aligns

https://cameronrwolfe.substack.com/p/easily-train-a-specialized-llm-peft 7/46

14.04.2024 15:31 Easily Train a Specialized LLM: PEFT, LoRA, QLoRA, LLaMA-Adapter, and More

with the expectations of human users. To perform alignment, we frst defne several criteria (e.g., helpfulness, harmlessness, the ability to follow instructions, etc.) that we want to instill within the model. Then, we can align the model based on these criteria using two fnetuning techniques in sequence (shown above):

Supervised Finetuning (SFT) [[link](https://cameronrwolfe.substack.com/p/understanding-and-using-supervised)]



Reinforcement Learning from Human Feedback (RLHF) [[link](https://cameronrwolfe.substack.com/p/the-story-of-rlhf-origins-motivations)]



SFT trains the language model over a set of high-quality reference outputs using a next token prediction objective, and the LLM learns to mimic the style and format of responses within this dataset. In contrast, RLHF collects feedback (from humans) on the LLM’s output and uses this feedback as a training signal.

**Finetuning.** Afer pretraining and alignment (i.e., an optional step for generative LLMs), we have a generic foundation model that can be used as a starting point for solving many tasks. To solve a task, however, we must adapt the language model to this task, usually via fnetuning (i.e., further training on data from a task). The combination of pretraining followed by fnetuning is commonly referred to as ***transfer learning***, and this approach has numerous benefts:

Pretraining, although expensive, is only performed once and can be shared as a starting point for fnetuning on several diferent tasks.



In most cases, we can download a pretrained model online and only focus on performing fnetuning, which is cheap compared to pretraining.



Finetuning a pretrained model usually performs better than training a model from scratch, despite requiring less fnetuning data and training time.

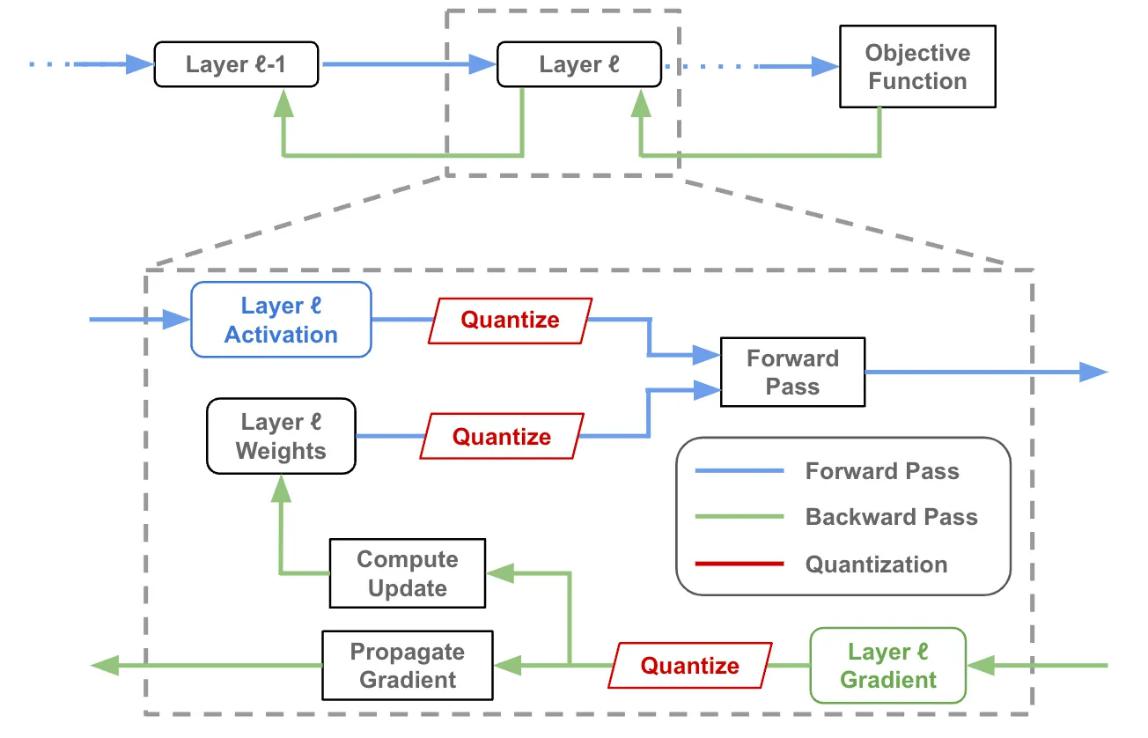


Although fnetuning is computationally cheap relative to pretraining or training from scratch, it can still be quite expensive, especially for the massive generative LLMs that have recently become popular. Within this overview, we will focus upon this fnetuning process and how we can make it more efcient from a compute and memory perspective, ***allowing practitioners to train specialized LLMs with fewer/smaller GPUs and less investment of time and money***.

https://cameronrwolfe.substack.com/p/easily-train-a-specialized-llm-peft 8/46

14.04.2024 15:31 Easily Train a Specialized LLM: PEFT, LoRA, QLoRA, LLaMA-Adapter, and More

Model Quantization



Quantization in the forward and backward pass of neural network

The idea of quantization in deep learning refers to quantizing (i.e., converting to a format that uses fewer bits) a model’s activations and weights—***during the forward and/or backward pass***—such that we can use low precision arithmetic duringtraining and/or inference. Assuming we have access to an accelerator that supports arithmetic at lower precisions, we can actually save costs by performing the model’s forward and backward pass using lower precision. Plus, we usually do not sacrifce performance when performing such quantization—***we get these efciency benefts basically for free***! As we will see, quantization techniques are commonlycombined with LoRA to save costs during both training and inference.

The literature on quantization for deep learning is vast. However, proposed techniques can be (roughly) categorized into three primary areas:

***Quantized Training*** [[link](https://cameronrwolfe.substack.com/p/quantized-training-with-deep-networks-82ea7f516dc6)]: perform quantization during training to make the training process more efcient.



https://cameronrwolfe.substack.com/p/easily-train-a-specialized-llm-peft 9/46

14.04.2024 15:31 Easily Train a Specialized LLM: PEFT, LoRA, QLoRA, LLaMA-Adapter, and More

***Post-Training Quantization*** [[link](https://lightning.ai/docs/pytorch/stable/advanced/post_training_quantization.html)]: quantize a model’s weights afer training to make inference more efcient.

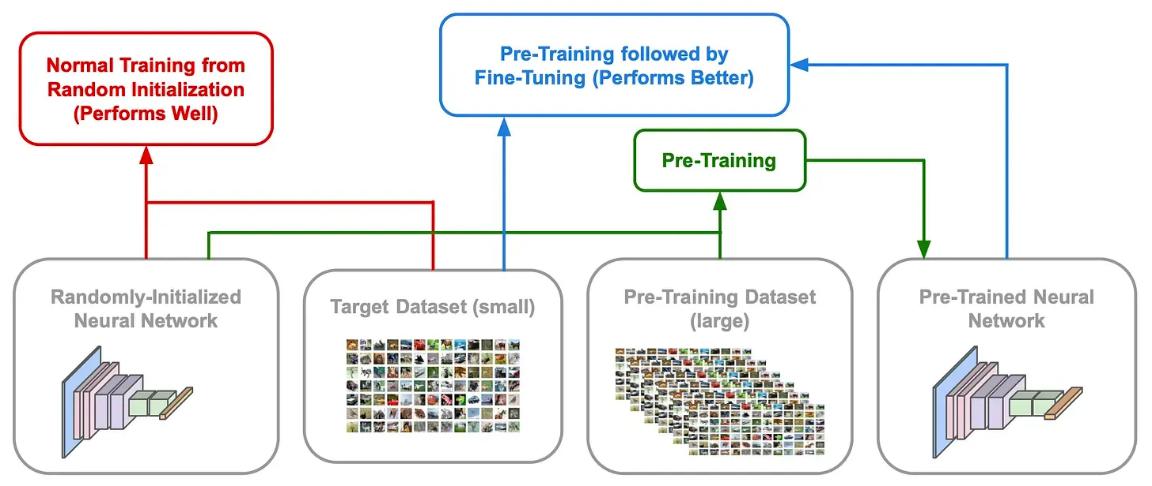


***Quantization-Aware Training*** [[link](https://towardsdatascience.com/inside-quantization-aware-training-4f91c8837ead)]: train in a manner that makes the model more amenable to quantization, thus avoiding performance degradations due to post-training quantization 3.



**Side note.** One of my favorite applications of quantization is automatic mixed-precision (AMP) training. With AMP, we can train arbitrary neural networks using lower precision in certain portions of the network architecture. This approach can cut training time in half while achieving comparable performance without any code changes. To try this out, check out the [AMP package](https://developer.nvidia.com/automatic-mixed-precision) from NVIDIA, which has easy-to-use integrations with common packages like PyTorch, or the [mixed-precision training options](https://huggingface.co/docs/transformers/v4.18.0/en/performance) within HuggingFace.

Adaptation of Foundation Models



Schematic depiction of training from scratch, pretraining, and transfer

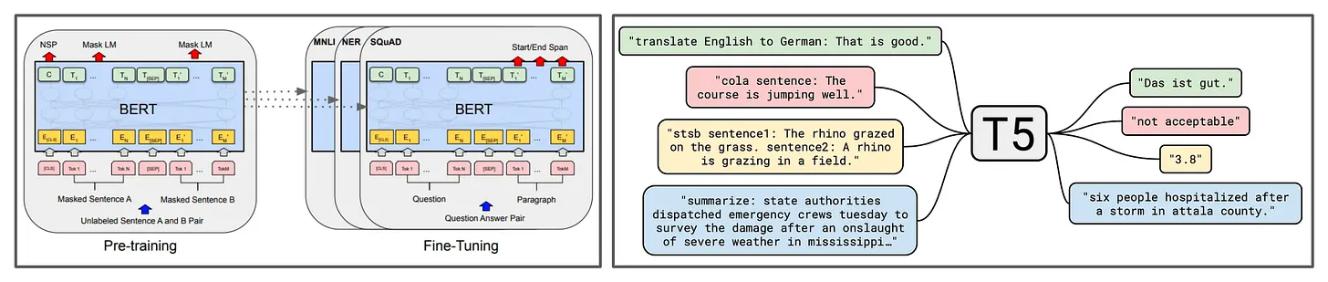
learning (i.e., pretraining + finetuning)

Self-supervised pretraining has been heavily leveraged by language models even before the advent of the GPT-style LLMs that are so popular today. Put simply, self-supervised learning allows us to meaningfully pretrain language models over large amounts of unlabeled text. The resulting model can then be fnetuned—***or trained further***—to accomplish some downstream task; see above. Given that pretraining

https://cameronrwolfe.substack.com/p/easily-train-a-specialized-llm-peft 10/46

14.04.2024 15:31 Easily Train a Specialized LLM: PEFT, LoRA, QLoRA, LLaMA-Adapter, and More

endows the model with a certain amount of linguistic understanding, fnetuning is more efcient than training from scratch—***the model converges more quickly, performs better, and requires less data to perform well***.



(from [9, 10])

For example, BERT and T5 [9, 10] are pretrained using a Cloze objective 4 and fnetuned to solve a variety of downstream tasks; see above. Generative LLMs follow a similar approach, but pretraining is performed with a [next token](https://cameronrwolfe.substack.com/i/136638774/understanding-next-token-prediction) [prediction objective](https://cameronrwolfe.substack.com/i/136638774/understanding-next-token-prediction), which is more conducive to generating text. Then, the model can be fnetuned using a variety of diferent techniques (e.g., [SFT](https://cameronrwolfe.substack.com/p/understanding-and-using-supervised), [RLHF](https://cameronrwolfe.substack.com/p/the-story-of-rlhf-origins-motivations), [task-specifc fnetuning](https://cameronrwolfe.substack.com/i/85568430/language-models-are-unsupervised-multitask-learners-gpt), etc.) to fulfll its role in a desired application.

**Adapting foundation models.** Despite the large variety of language models that exist, self-supervised pretraining is a common characteristic between most of them. ***Why?*** Pretraining can be quite expensive due to the amount of unlabeled data on which we want to train 5. However, the pretraining process only needs to be performed once and can be shared (either publicly or within an organization) aferwards. We can fnetune this single pretrained checkpoint any number of times to accomplish a variety of diferent downstream tasks.

***“Many applications in natural language processing rely on adapting one large-scale, pre-trained language model to multiple downstream applications.”*** - from [1]



For generative LLMs, the pretraining process is [especially expensive](https://www.mosaicml.com/blog/gpt-3-quality-for-500k), but it plays a massive role in the model’s downstream performance. In order for generative LLMs to perform well, we need to pretrain them over a large, high-quality corpus of data. Luckily, however, we usually don’t need to pay for the (massive) cost of this

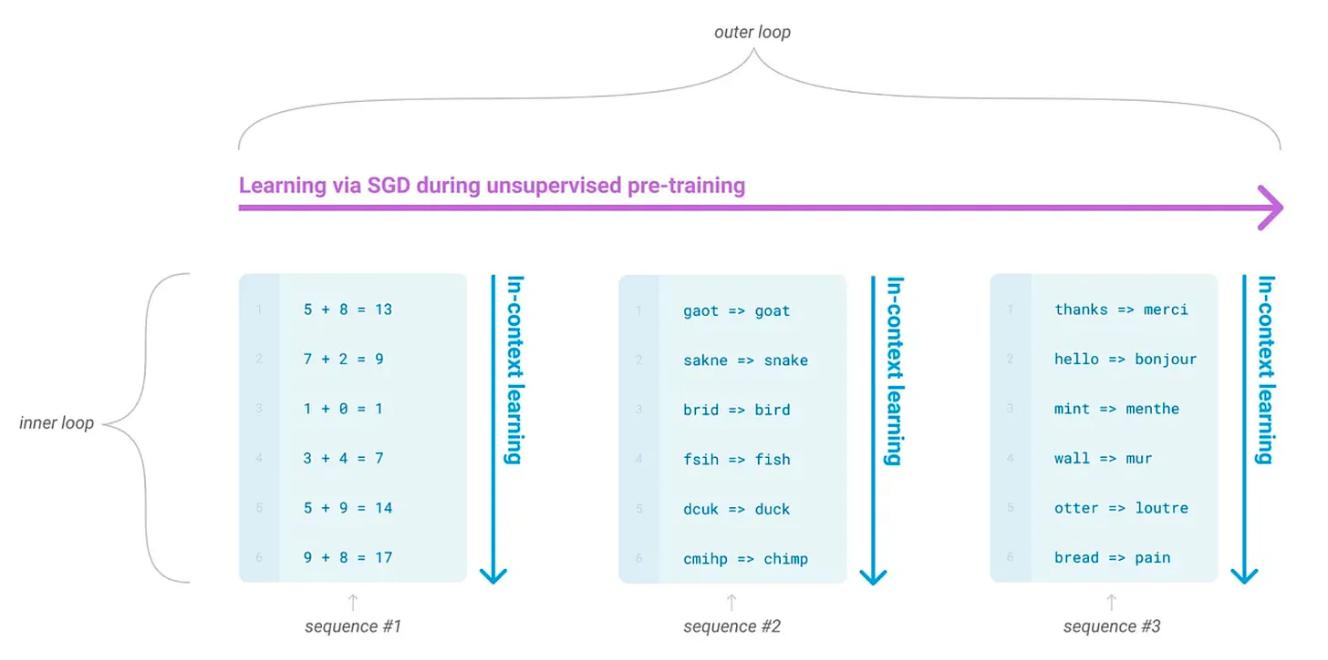
https://cameronrwolfe.substack.com/p/easily-train-a-specialized-llm-peft 11/46

14.04.2024 15:31 Easily Train a Specialized LLM: PEFT, LoRA, QLoRA, LLaMA-Adapter, and More

pretraining process—***a variety of pretrained (base) LLMs are openly available online***; e.g., [LLaMA](https://cameronrwolfe.substack.com/p/llama-llms-for-everyone), [LLaMA-2](https://cameronrwolfe.substack.com/p/llama-2-from-the-ground-up), [MPT](https://cameronrwolfe.substack.com/p/democratizing-ai-mosaicmls-impact), [Falcon](https://cameronrwolfe.substack.com/p/falcon-the-pinnacle-of-open-source), and [Mistral](https://mistral.ai/news/announcing-mistral-7b/). Using fnetuning or in-context learning, these models can be repurposed to solve a variety of diferent tasks. We will now take a look at several such approaches and consider how these models can be most efciently adapted to solve a task.

In-Context Learning

Pretrained LLMs have rudimentary abilities to solve problems via prompting, but the alignment process improves their instruction following capabilities, making the model more capable of solving tasks via in-context learning; see below.



In-context learning refers to solving a variety of problems with a single model by writing task-specifc prompts (i.e., textual inputs to the language model), rather than fnetuning the model on each task. For more information on the prompting process and diferent techniques that exist, check out the articles below:

Practical Prompt Engineering [[link](https://cameronrwolfe.substack.com/p/practical-prompt-engineering-part)]



Advanced Prompt Engineering [[link](https://cameronrwolfe.substack.com/p/advanced-prompt-engineering)]



https://cameronrwolfe.substack.com/p/easily-train-a-specialized-llm-peft 12/46

14.04.2024 15:31 Easily Train a Specialized LLM: PEFT, LoRA, QLoRA, LLaMA-Adapter, and More

Because no fnetuning is required and one model can be used for all tasks, in-context learning is (by far) the easiest way to adapt an LLM to solve a downstream task. However, this approach lags behind the performance of fnetuning, making fnetuning a common approach for creating specialized LLMs in practice.

Full Finetuning

If in- context learning does not perform well enough for our needs, our next option is to fnetune the model on our dataset. The most naive approach is full fnetuning, where we train the model end-to-end and update all of its parameters over new data. This approach works well, but it has several downsides:

The fnetuned model contains as many parameters as the original model, which becomes more burdensome for large models (e.g., LLMs).



We must store all of the model’s parameters each time we either retrain the model or train it on a new/diferent task.



Training the full model is both compute and memory intensive.



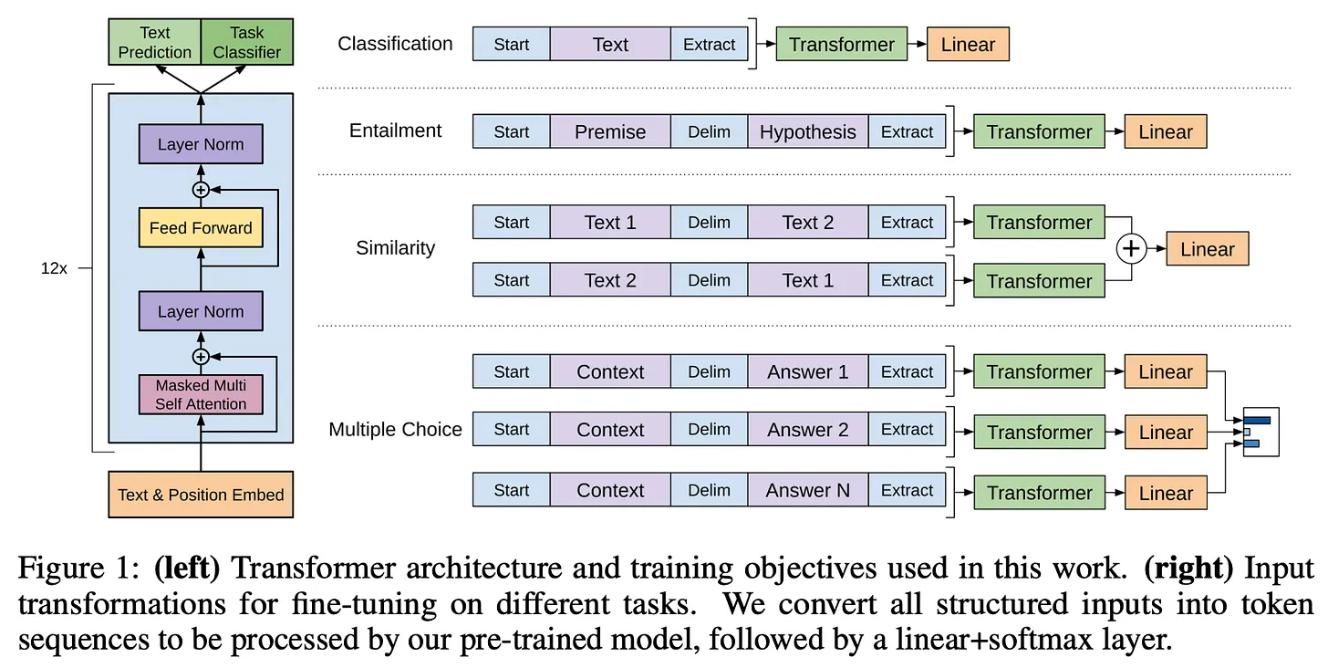
End-to-end training might require more hyperparameter tuning or data to avoid overftting and achieve the best possible results.



Full fnetuning becomes burdensome if we ***i)*** want to frequently retrain the model or ***ii)*** are fnetuning the same model on many diferent tasks. In these cases, we end up with several “copies” of an already large model. Storing and deploying many independent instances of a large model can be challenging; see below.

https://cameronrwolfe.substack.com/p/easily-train-a-specialized-llm-peft 13/46

14.04.2024 15:31 Easily Train a Specialized LLM: PEFT, LoRA, QLoRA, LLaMA-Adapter, and More



[(from [14])](https://substackcdn.com/image/fetch/f_auto,q_auto:good,fl_progressive:steep/https%3A%2F%2Fbucketeer-e05bbc84-baa3-437e-9518-adb32be77984.s3.amazonaws.com%2Fpublic%2Fimages%2F60d46502-4340-48d7-8db6-057993f82060_1622x816.png)

[Beyond the burden of storing and deploying multiple fnetuned models, training](https://substackcdn.com/image/fetch/f_auto,q_auto:good,fl_progressive:steep/https%3A%2F%2Fbucketeer-e05bbc84-baa3-437e-9518-adb32be77984.s3.amazonaws.com%2Fpublic%2Fimages%2F60d46502-4340-48d7-8db6-057993f82060_1622x816.png) [large models end-to-end is difcult in itself—***the memory overhead and amount of***](https://substackcdn.com/image/fetch/f_auto,q_auto:good,fl_progressive:steep/https%3A%2F%2Fbucketeer-e05bbc84-baa3-437e-9518-adb32be77984.s3.amazonaws.com%2Fpublic%2Fimages%2F60d46502-4340-48d7-8db6-057993f82060_1622x816.png)[***computation required*** 6 ***is signifcant***. To learn more, take a look at the link below,](https://substackcdn.com/image/fetch/f_auto,q_auto:good,fl_progressive:steep/https%3A%2F%2Fbucketeer-e05bbc84-baa3-437e-9518-adb32be77984.s3.amazonaws.com%2Fpublic%2Fimages%2F60d46502-4340-48d7-8db6-057993f82060_1622x816.png)[which details the fnetuning process for the LLaMA-2 70B model! As we will see,](https://substackcdn.com/image/fetch/f_auto,q_auto:good,fl_progressive:steep/https%3A%2F%2Fbucketeer-e05bbc84-baa3-437e-9518-adb32be77984.s3.amazonaws.com%2Fpublic%2Fimages%2F60d46502-4340-48d7-8db6-057993f82060_1622x816.png) [fnetuning large models end-to-end is not cheap and/or easy by any means.](https://substackcdn.com/image/fetch/f_auto,q_auto:good,fl_progressive:steep/https%3A%2F%2Fbucketeer-e05bbc84-baa3-437e-9518-adb32be77984.s3.amazonaws.com%2Fpublic%2Fimages%2F60d46502-4340-48d7-8db6-057993f82060_1622x816.png)

Adapter Layers and Pre x Tuning

***“Since the inception of transfer learning, dozens of works have sought to make model adaptation more parameter- and compute-efcient”*** - from [1]

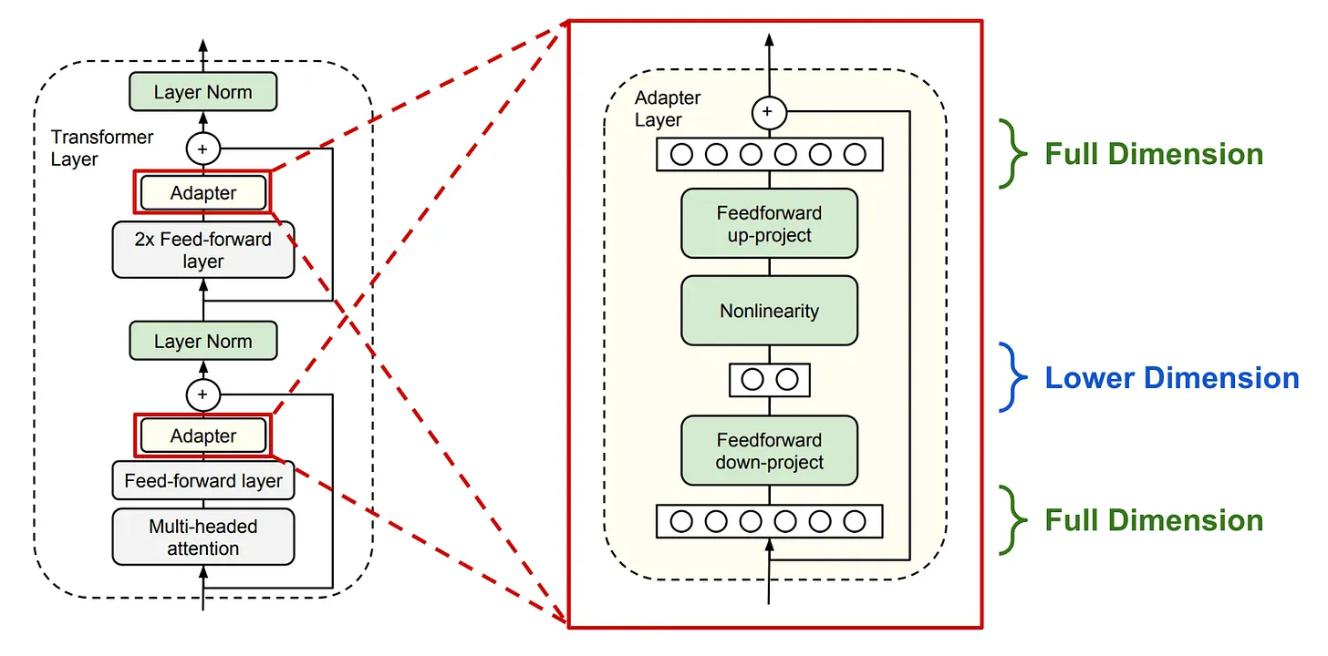
******

Given these limitations, we might wonder whether we could fnetune an LLM in a manner that is more compute/data efcient but maintains the performance of end-to-end fnetuning. A variety of research has been done in this area that boils down to one, core idea—***only adapting a small portion of the model’s parameters (or added parameters) during fnetuning***. Each fnetuned model only has a small number of task-specifc parameters that should be stored/loaded, thus lessoning the compute/memory overhead of fnetuning and simplifying the deployment process.

https://cameronrwolfe.substack.com/p/easily-train-a-specialized-llm-peft 14/46

14.04.2024 15:31 Easily Train a Specialized LLM: PEFT, LoRA, QLoRA, LLaMA-Adapter, and More

Prior to the proposal of LoRA [1], two main parameter-efcient fnetuning techniques were used: ***adapter layers*** and ***prefx tuning***.



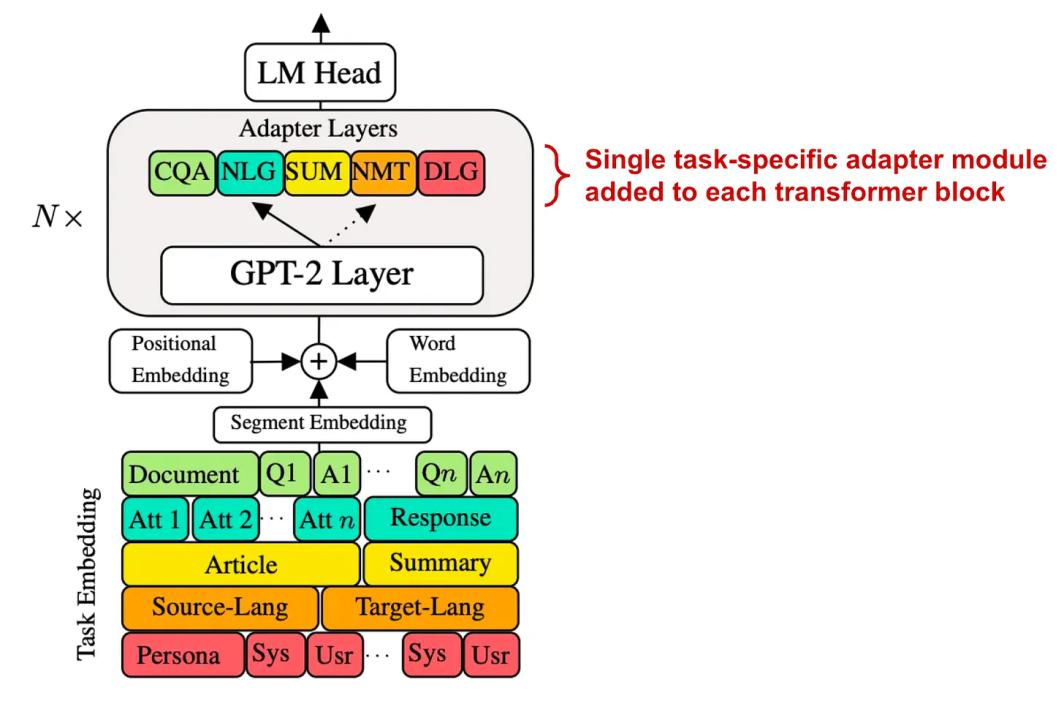
Adapter layers add two bottleneck-style feedforward modules into each

transformer block (from [2])

**Adapter layers** were originally proposed in [2], where authors inserted two extra adapter blocks into each transformer block. Each adapter block is a bottleneck-style feedforward module that ***i)*** decreases the dimensionality of the input via a (trainable) linear layer, ***ii)*** applies a non-linearity, and ***iii)*** restores the original dimensionality of the input via a fnal (trainable) linear layer. Put simply, the adapter blocks are extra trainable modules inserted into the existing transformer block—***in [2], adapter blocks are inserted afer both attention and feedforward layers***— that have a small number of parameters 7 and can be fnetuned while keeping the weights of the pretrained model fxed. As such, each fnetuned version of the model only has a small number of task-specifc parameters associated with it.

https://cameronrwolfe.substack.com/p/easily-train-a-specialized-llm-peft 15/46

14.04.2024 15:31 Easily Train a Specialized LLM: PEFT, LoRA, QLoRA, LLaMA-Adapter, and More



[(from [3])](https://substackcdn.com/image/fetch/f_auto,q_auto:good,fl_progressive:steep/https%3A%2F%2Fsubstack-post-media.s3.amazonaws.com%2Fpublic%2Fimages%2Fa4567103-c992-47d9-b824-a3eb19d65ca5_1580x1044.png)

[Several variants of adapter layers have been proposed that are more efcient and](https://substackcdn.com/image/fetch/f_auto,q_auto:good,fl_progressive:steep/https%3A%2F%2Fsubstack-post-media.s3.amazonaws.com%2Fpublic%2Fimages%2Fa4567103-c992-47d9-b824-a3eb19d65ca5_1580x1044.png) [even go beyond language models [4, 5]. For example, authors in [3] simplify the](https://substackcdn.com/image/fetch/f_auto,q_auto:good,fl_progressive:steep/https%3A%2F%2Fsubstack-post-media.s3.amazonaws.com%2Fpublic%2Fimages%2Fa4567103-c992-47d9-b824-a3eb19d65ca5_1580x1044.png) structure of adapter layers such that only a single task-specifc adapter is added to each transformer block, as well as an extra [LayerNorm](https://pytorch.org/docs/stable/generated/torch.nn.LayerNorm.html) module; see above.

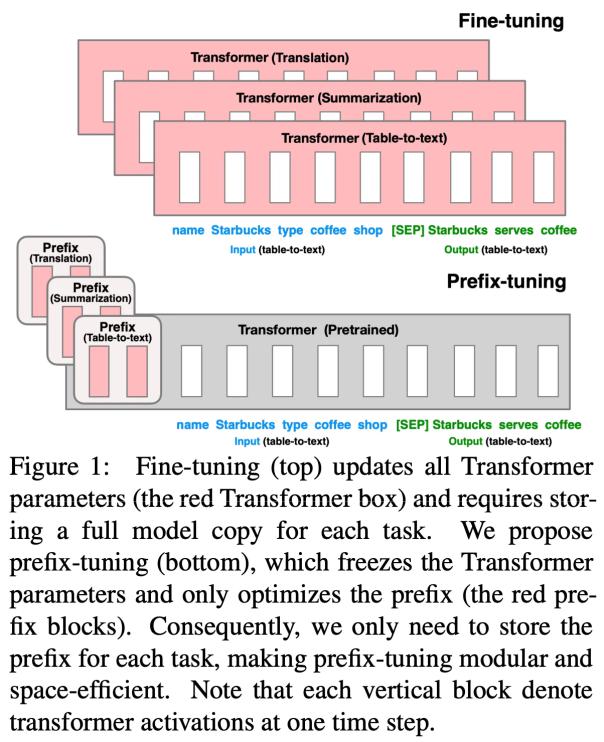
***“Prefx tuning keeps language model parameters frozen, but optimizes a small continuous task vector. Prefx tuning draws inspiration from prompting, allowing subsequent tokens to attend to this prefx as if it were virtual tokens.”*** - from [6]



**Prefx tuning.** Another parameter-efcient fnetuning alternative is prefx tuning, which keeps the language model’s parameters frozen and only fnetunes a few (trainable) token vectors that are added to the beginning of the model’s input sequence. These token vectors can be interchanged for diferent tasks; see below.

https://cameronrwolfe.substack.com/p/easily-train-a-specialized-llm-peft 16/46

14.04.2024 15:31 Easily Train a Specialized LLM: PEFT, LoRA, QLoRA, LLaMA-Adapter, and More

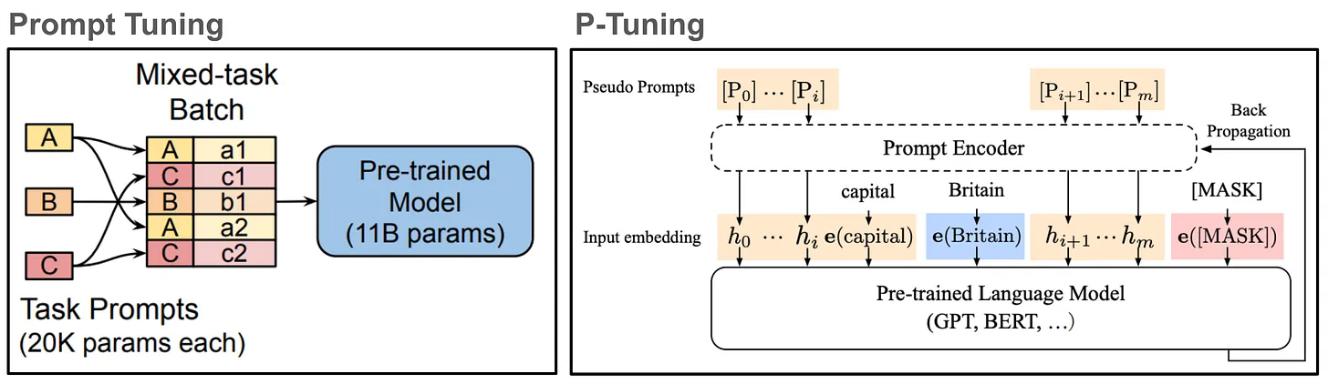


[(from [6])](https://substackcdn.com/image/fetch/f_auto,q_auto:good,fl_progressive:steep/https%3A%2F%2Fsubstack-post-media.s3.amazonaws.com%2Fpublic%2Fimages%2F5381a048-df74-4871-9842-2d99caf8786b_1204x1480.png)

[In other words, prefx tuning adds a few extra token vectors to the model’s input.](https://substackcdn.com/image/fetch/f_auto,q_auto:good,fl_progressive:steep/https%3A%2F%2Fsubstack-post-media.s3.amazonaws.com%2Fpublic%2Fimages%2F5381a048-df74-4871-9842-2d99caf8786b_1204x1480.png) [However, these added vectors do not correspond to a specifc word or token—***we***](https://substackcdn.com/image/fetch/f_auto,q_auto:good,fl_progressive:steep/https%3A%2F%2Fsubstack-post-media.s3.amazonaws.com%2Fpublic%2Fimages%2F5381a048-df74-4871-9842-2d99caf8786b_1204x1480.png)[***train the entries of these vectors just like normal model parameters***. This idea was frst](https://substackcdn.com/image/fetch/f_auto,q_auto:good,fl_progressive:steep/https%3A%2F%2Fsubstack-post-media.s3.amazonaws.com%2Fpublic%2Fimages%2F5381a048-df74-4871-9842-2d99caf8786b_1204x1480.png)proposed in [6], where we see that authors freeze all model parameters and train only a small set of prefx token vectors added to the model’s input layer for each task. These trainable prefx tokens can be shared across a task. Beyond prefx tuning as it was originally proposed, several works have extended this idea. For example, authors in [7] explore the creation of trainable “sof prompts” that condition an LLM’s output on certain tasks, while [8] proposes the concatenation of trainable (continuous) prompts with discrete (normal) prompts; see below.

https://cameronrwolfe.substack.com/p/easily-train-a-specialized-llm-peft 17/46

14.04.2024 15:31 Easily Train a Specialized LLM: PEFT, LoRA, QLoRA, LLaMA-Adapter, and More



[(from [7, 8])](https://substackcdn.com/image/fetch/f_auto,q_auto:good,fl_progressive:steep/https%3A%2F%2Fsubstack-post-media.s3.amazonaws.com%2Fpublic%2Fimages%2F359126b3-0b45-4695-98b4-6842cafee418_2324x674.png)

[**What’s the problem?** Both prefx tuning and adapter layers reduce the compute](https://substackcdn.com/image/fetch/f_auto,q_auto:good,fl_progressive:steep/https%3A%2F%2Fsubstack-post-media.s3.amazonaws.com%2Fpublic%2Fimages%2F359126b3-0b45-4695-98b4-6842cafee418_2324x674.png) [requirements, memory overhead, and number of trainable parameters associated](https://substackcdn.com/image/fetch/f_auto,q_auto:good,fl_progressive:steep/https%3A%2F%2Fsubstack-post-media.s3.amazonaws.com%2Fpublic%2Fimages%2F359126b3-0b45-4695-98b4-6842cafee418_2324x674.png) [with fnetuning an LLM. Despite giving us a parameter-efcient and performant](https://substackcdn.com/image/fetch/f_auto,q_auto:good,fl_progressive:steep/https%3A%2F%2Fsubstack-post-media.s3.amazonaws.com%2Fpublic%2Fimages%2F359126b3-0b45-4695-98b4-6842cafee418_2324x674.png) [alternative to full fnetuning, however, these approaches do not come without](https://substackcdn.com/image/fetch/f_auto,q_auto:good,fl_progressive:steep/https%3A%2F%2Fsubstack-post-media.s3.amazonaws.com%2Fpublic%2Fimages%2F359126b3-0b45-4695-98b4-6842cafee418_2324x674.png) limitations. Adapter layers add layers to the underlying model that—***despite having very few parameters***—must be processed sequentially, resulting in extra inference latency and slower training. Furthermore, prefx tuning is ofentimes difcult to optimize (i.e., the fnetuning process is less stable), reduces the available context window, and does not monotonically improve performance with respect to the number of trainable parameters 8. ***Low-Rank Adaptation (LoRA) aims to eliminate such issues while maintaining performance that is comparable to full fnetuning***.

***“LoRA performs on-par or better than fnetuning in model quality on RoBERTa, DeBERTa, GPT-2, and GPT-3, despite having fewer trainable parameters, a higher training throughput, and, unlike adapters, no additional inference latency.”*** - from [1]



Finetuning LLMs More Ef ciently

Given our discussion of fnetuning techniques so far, it should be clear that we need a fnetuning approach that is not just parameter efcient, but also:

1. ***Compute efcient***: the training process should be fast and cheap.
2. ***Memory efcient***: we should not need massive GPUs to fnetune an LLM.

https://cameronrwolfe.substack.com/p/easily-train-a-specialized-llm-peft 18/46

14.04.2024 15:31 Easily Train a Specialized LLM: PEFT, LoRA, QLoRA, LLaMA-Adapter, and More

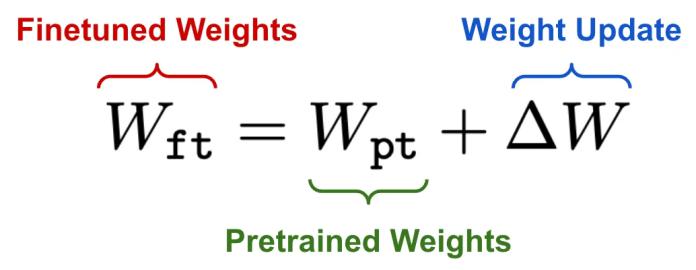
1. ***Easy to deploy***: we should not have to deploy several copies of an LLM for each task that we want to solve.

As we will see, Low-Rank Adaptation (LoRA) [1] checks all of these boxes! With LoRA, we lower the barrier to entry for fnetuning specialized LLMs, achieve performance that is comparable to end-to-end fnetuning, can easily switch between specifc versions of a model, and have no increase in inference latency. Due to its practical utility, LoRA has also been explored heavily within the research community, leading to a plethora of variants and extensions.

LoRA: Low-Rank Adaptation of Large Language Models [1]



When we fnetune a language model, we modify the underlying parameters of the model. To make this idea more concrete, we can formulate the parameter update derived from fnetuning as shown in the equation below.

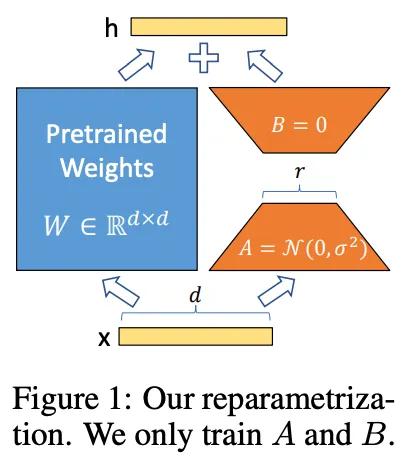


Finetuning updates the pretrained model’s weights

The core idea behind LoRA is to model this update to the model’s parameters with a low-rank decomposition 9, implemented in practice as a pair of [linear projections](https://www.cs.bu.edu/fac/snyder/cs132-book/L08MatrixofLinearTranformation.html). LoRA leaves the pretrained layers of the LLM fxed and injects a trainable rank decomposition matrix into each layer of the model; see below.

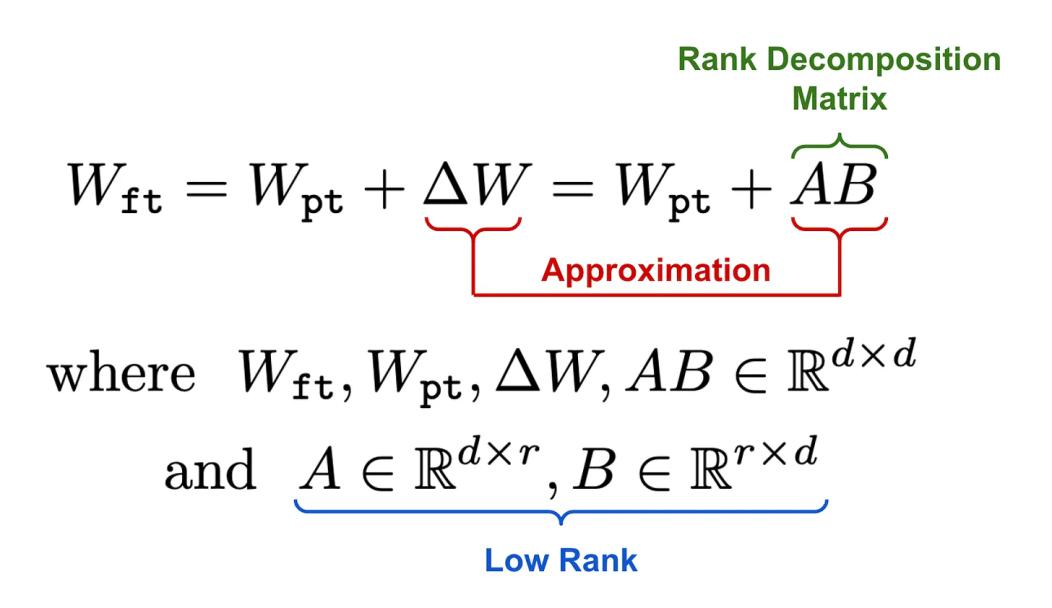
https://cameronrwolfe.substack.com/p/easily-train-a-specialized-llm-peft 19/46

14.04.2024 15:31 Easily Train a Specialized LLM: PEFT, LoRA, QLoRA, LLaMA-Adapter, and More



[(from [1])](https://substackcdn.com/image/fetch/f_auto,q_auto:good,fl_progressive:steep/https%3A%2F%2Fsubstack-post-media.s3.amazonaws.com%2Fpublic%2Fimages%2Fcdccfb29-1e68-4032-a5b7-a37927e6df10_414x462.png)

[**Rank decomposition matrix.** Put simply, the rank decomposition matrix is just two](https://substackcdn.com/image/fetch/f_auto,q_auto:good,fl_progressive:steep/https%3A%2F%2Fsubstack-post-media.s3.amazonaws.com%2Fpublic%2Fimages%2Fcdccfb29-1e68-4032-a5b7-a37927e6df10_414x462.png) [linear projections that reduce and restore the dimensionality of the input. The](https://substackcdn.com/image/fetch/f_auto,q_auto:good,fl_progressive:steep/https%3A%2F%2Fsubstack-post-media.s3.amazonaws.com%2Fpublic%2Fimages%2Fcdccfb29-1e68-4032-a5b7-a37927e6df10_414x462.png) [output of these two linear projections is added to the output derived from the](https://substackcdn.com/image/fetch/f_auto,q_auto:good,fl_progressive:steep/https%3A%2F%2Fsubstack-post-media.s3.amazonaws.com%2Fpublic%2Fimages%2Fcdccfb29-1e68-4032-a5b7-a37927e6df10_414x462.png) [model’s pretrained weights. The updated layer formed by the addition of these two](https://substackcdn.com/image/fetch/f_auto,q_auto:good,fl_progressive:steep/https%3A%2F%2Fsubstack-post-media.s3.amazonaws.com%2Fpublic%2Fimages%2Fcdccfb29-1e68-4032-a5b7-a37927e6df10_414x462.png) parallel transformations is formulated as shown below. As we can see, adding LoRA to a layer directly learns the update to the underlying layer’s weights.



LoRA trains a low-rank approximation to the weight matrix update

derived during finetuning

https://cameronrwolfe.substack.com/p/easily-train-a-specialized-llm-peft 20/46

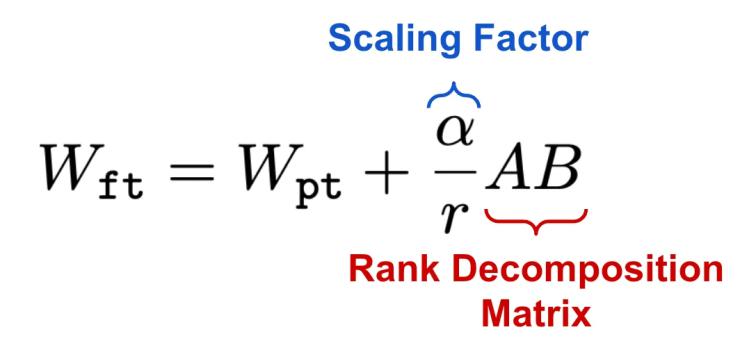
14.04.2024 15:31 Easily Train a Specialized LLM: PEFT, LoRA, QLoRA, LLaMA-Adapter, and More

The matrix product AB has the same dimension as a full fnetuning update. Decomposing the update as a product of two smaller matrices ensures that the update is [low rank](https://www.mathsisfun.com/algebra/matrix-rank.html) and signifcantly reduces the number of parameters that we have to train. Instead of directly fnetuning the parameters in the pretrained LLM’s layers, LoRA only optimizes the rank decomposition matrix, yielding a result that approximates the update derived from full fnetuning. We initialize A with random, small values, while B is initialized as zero, ensuring that we begin the fnetuning process with the model’s original, pretrained weights.

***“We roughly recover the expressiveness of full fne-tuning by setting the LoRA rank r to the rank of the pre-trained weight matrices.”*** - from [1]



Increasing r improves LoRA’s approximation of the full fnetuning update, but incredibly small values of r sufce in practice, allowing us to signifcantly reduce compute and memory costs with minimal impact on performance. For example, we can use LoRA to fnetune GPT-3 using only 0.01% of total parameters and still achieve performance comparable to that of full fnetuning.



Scaling factor in LoRA

**Scaling factor.** Once the low-rank update to the weight matrix is derived, we scale it by a factor α prior to adding it to the model’s pretrained weights. Such an approach yields the adaptation rule shown above. The default value of the scaling factor is one, meaning that the pretrained weights and the low-rank weight update are weighted equally when computing the model’s forward pass. However, the value of α can be changed to balance the importance of the pretrained model and new

https://cameronrwolfe.substack.com/p/easily-train-a-specialized-llm-peft 21/46

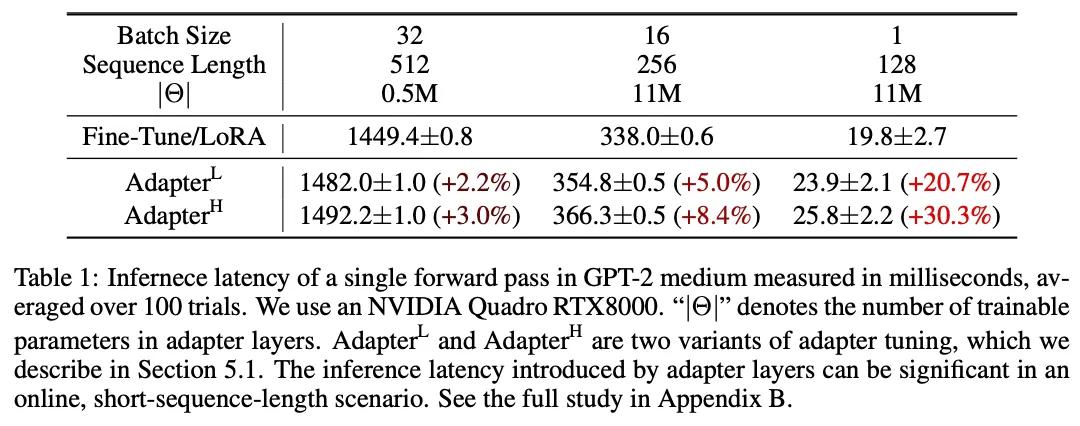
14.04.2024 15:31 Easily Train a Specialized LLM: PEFT, LoRA, QLoRA, LLaMA-Adapter, and More

task-specifc adaptation. [Recent empirical analysis](https://lightning.ai/pages/community/lora-insights/) indicates that larger values of α are necessary for LoRA with a higher rank (i.e., larger r —> la α).

**Comparison to adapter layers.** At frst glance, the approach used by LoRA might seem similar to adapter layers. However, there are two notable diferences:

1. There is no non-linearity between the two linear projections.
2. The rank decomposition matrix is injected into an existing layer of the model, instead of being sequentially added as an extra layer.

The biggest impact of these changes is the fact that LoRA has no added inference latency compared to the original pretrained model; see below. When deploying a fnetuned LoRA model into production, we can directly compute and store the updated weight matrix derived from LoRA. As such, the structure of the model is identical to the pretrained model—***the weights are just diferent***.



By storing both the model’s pretrained weights and LoRA modules derived from fnetuning on several diferent tasks, we can “switch out” LoRA modules by:

1. Subtracting the LoRA update for one task from the model’s weights.
2. Adding the LoRA update for another task to the model’s weights.

In comparison, switching between models that are fnetuned end-to-end on diferent tasks requires loading all model parameters in and out of memory,

https://cameronrwolfe.substack.com/p/easily-train-a-specialized-llm-peft 22/46

14.04.2024 15:31 Easily Train a Specialized LLM: PEFT, LoRA, QLoRA, LLaMA-Adapter, and More

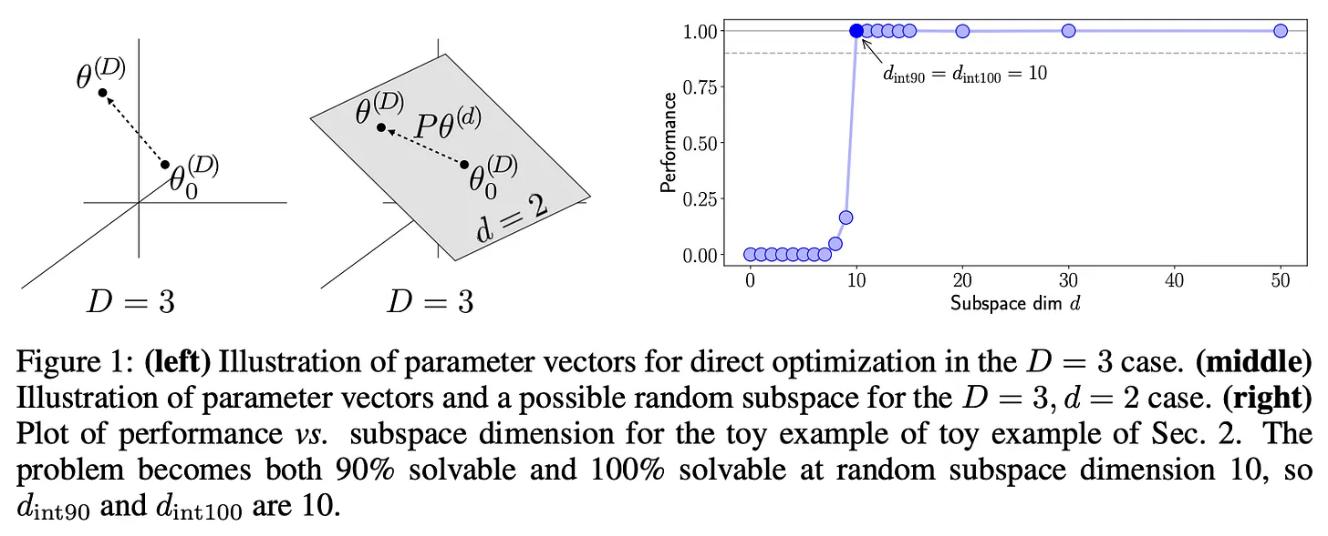
creating a signifcant I/O bottleneck. ***LoRA’s efcient parameterization of the weight update derived from fnetuning makes switching between tasks efcient and easy***.

**Why does this work?** LoRA structures the weight update derived from fnetuning

with a low rank decomposition that contains very few trainable parameters. With

this in mind, we might wonder: ***Why does the model perform well despite dedicating so***

***few parameters to fnetuning? Wouldn’t we beneft from more trainable parameters?***



(from [15])

To answer this question, we can look at prior research [15], which shows that large models (e.g., LLMs) tend to have a low intrinsic dimension. Although this idea sounds complicated, it just means that the weight matrices of very large models tend to be low rank. In other words, ***not all of these parameters are necessary***! We can achieve comparable performance by decomposing these weight matrices into a representation that has way fewer trainable parameters; see above.

***“Aghajanyan et al. show that the learned over-parametrized models in fact reside on a low intrinsic dimension. We hypothesize that the change in weights during model adaptation also has a low intrinsic rank.”*** - from [1]

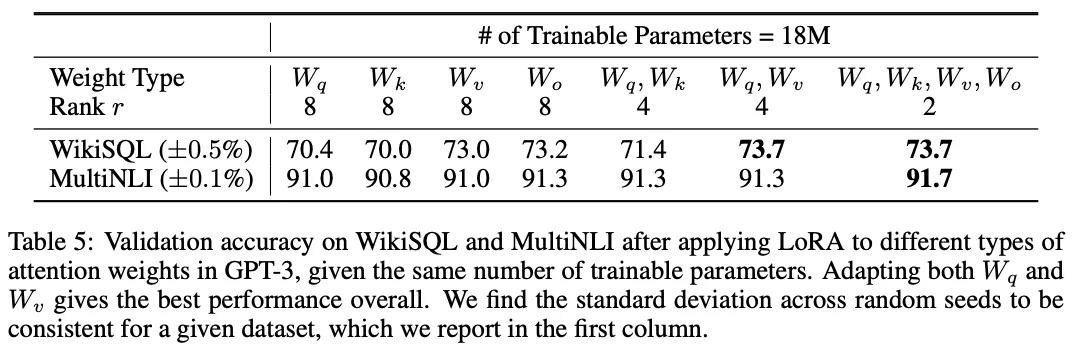


Given that the parameters of large models have low intrinsic dimension, we can reasonably infer that the same is true of models that are fnetuned—***the weight***

https://cameronrwolfe.substack.com/p/easily-train-a-specialized-llm-peft 23/46

14.04.2024 15:31 Easily Train a Specialized LLM: PEFT, LoRA, QLoRA, LLaMA-Adapter, and More

***update derived from fnetuning should also have a low intrinsic dimension***. As a result, techniques like LoRA that approximate the fnetuning update with a low rank decomposition should be able to learn efciently and efectively, producing a model with impressive performance despite having few trainable parameters.



(from [1])

**LoRA for LLMs.** The general idea proposed by LoRA can be applied to any type of dense layer for a neural network (i.e., more than just transformers!). When applying LoRA to LLMs, however, authors in [1] only use LoRA to adapt attention layer weights. Feed-forward modules and pretrained weights are kept fxed. We only update the rank decomposition matrix inserted into each attention layer. In particular, LoRA is used in [1] to update the query and value matrices of the attention layer, which is found in experiments to yield the best results; see above.

***“Compared to GPT-3 175B fne-tuned with Adam, LoRA can reduce the number of trainable parameters by 10,000 times and the GPU memory requirement by 3 times. LoRA performs on-par or better than fnetuning in model quality on RoBERTa, DeBERTa, GPT-2, and GPT-3, despite having fewer trainable parameters, a higher training throughput, and, unlike adapters, no additional inference latency.”*** - from [1]



However, [subsequent empirical analysis](https://magazine.sebastianraschka.com/i/138081202/enable-lora-for-more-layers) revealed that better results can be achieved by applying LoRA to all weight matrices in the transformer. Although these results may depend upon the application, we see that adapting all weight matrices with LoRA tends to yield competitive performance.

https://cameronrwolfe.substack.com/p/easily-train-a-specialized-llm-peft 24/46

14.04.2024 15:31 Easily Train a Specialized LLM: PEFT, LoRA, QLoRA, LLaMA-Adapter, and More

**The Benefts of LoRA** are plentiful as we can probably tell. However, some of the most notable benefts of this approach include the following:

A single pretrained model can be shared by several (much smaller) LoRA modules that adapt it to solve diferent tasks, which simplifes the deployment and hosting process.



LoRA modules can be “baked in” to the weights of a pretrained model to avoid extra inference latency, and we can quickly switch between diferent LoRA modules to solve diferent tasks.



When fnetuning an LLM with LoRA, we only have to maintain the optimizer state for a very small number of parameters 10, which signifcantly reduces memory overhead and allows fnetuning to be performed with more modest hardware (i.e., smaller/fewer GPUs with less memory).



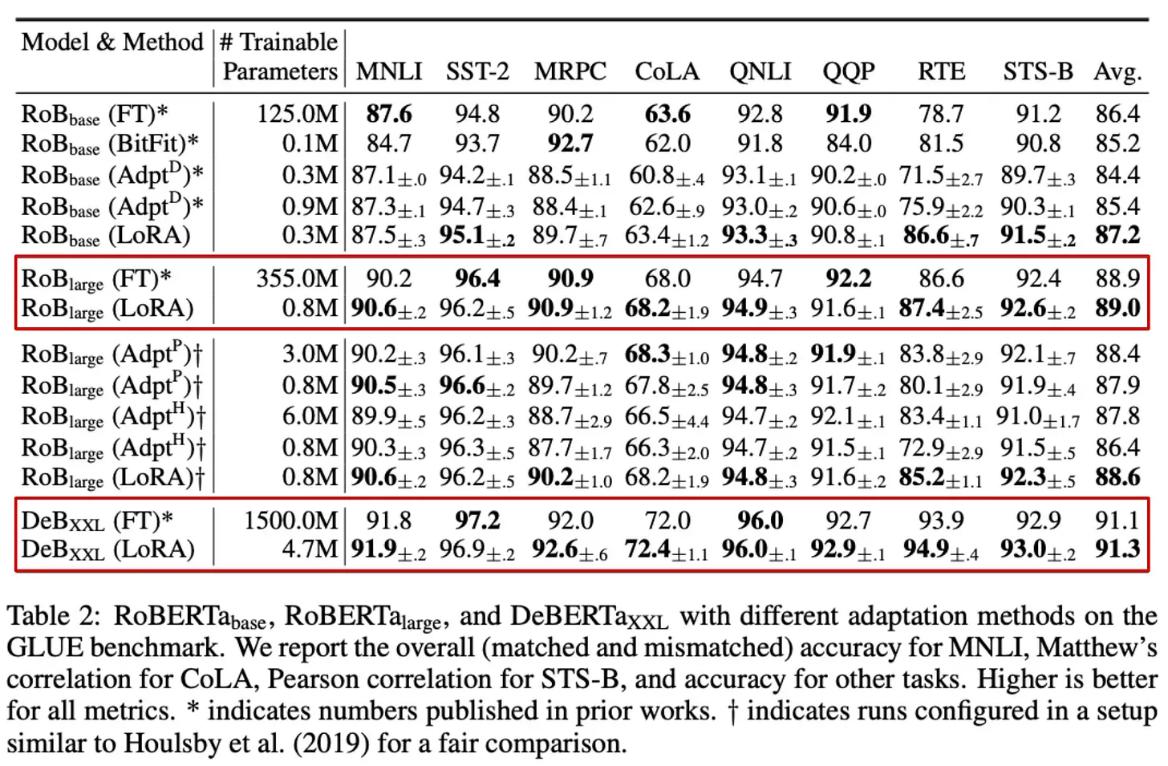
Finetuning with LoRA is signifcantly faster than end-to-end fnetuning (i.e., roughly 25% faster in the case of GPT-3).



LoRA signifcantly reduces the barrier to entry for fnetuning LLMs. Training is fast, we don’t need tons of fancy GPUs, each task has only a small number of task-specifc parameters associated with it, and—***as we will see soon***—there are a variety of resources and repos available online to get started with using LoRA.

https://cameronrwolfe.substack.com/p/easily-train-a-specialized-llm-peft 25/46

14.04.2024 15:31 Easily Train a Specialized LLM: PEFT, LoRA, QLoRA, LLaMA-Adapter, and More

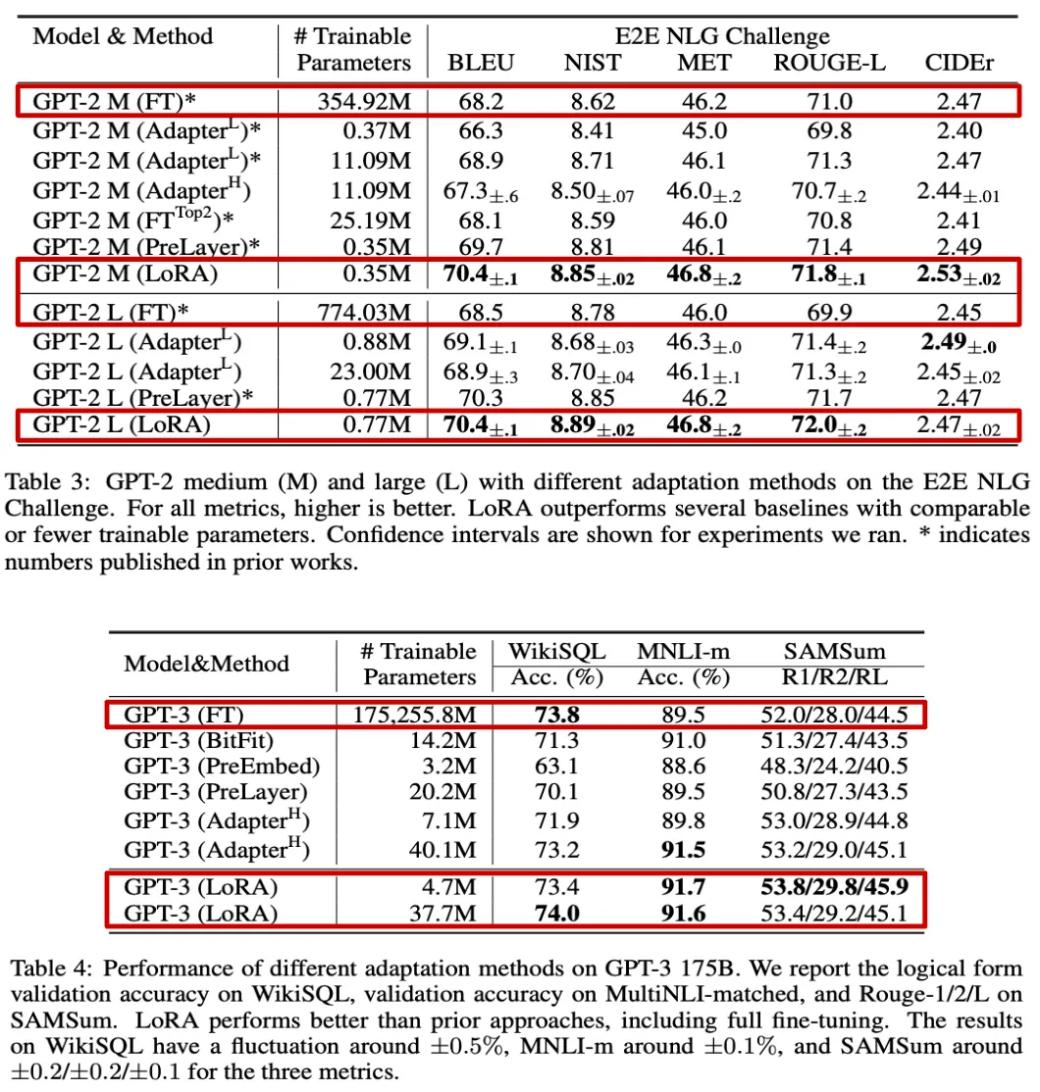


[(from [2])](https://substackcdn.com/image/fetch/f_auto,q_auto:good,fl_progressive:steep/https%3A%2F%2Fsubstack-post-media.s3.amazonaws.com%2Fpublic%2Fimages%2F0ae43026-6b42-43e8-b0c2-b24a7b20ac37_1946x1276.png)

[**Experimental results.** In [1], LoRA is tested with diferent types of LLMs, including](https://substackcdn.com/image/fetch/f_auto,q_auto:good,fl_progressive:steep/https%3A%2F%2Fsubstack-post-media.s3.amazonaws.com%2Fpublic%2Fimages%2F0ae43026-6b42-43e8-b0c2-b24a7b20ac37_1946x1276.png) [encoder-only (RoBERTa [16] and DeBERTa [17]) and decoder-only (GPT-2 [18] and](https://substackcdn.com/image/fetch/f_auto,q_auto:good,fl_progressive:steep/https%3A%2F%2Fsubstack-post-media.s3.amazonaws.com%2Fpublic%2Fimages%2F0ae43026-6b42-43e8-b0c2-b24a7b20ac37_1946x1276.png) [GPT-3 [11]) language models. In experiments with encoder-only architectures, we](https://substackcdn.com/image/fetch/f_auto,q_auto:good,fl_progressive:steep/https%3A%2F%2Fsubstack-post-media.s3.amazonaws.com%2Fpublic%2Fimages%2F0ae43026-6b42-43e8-b0c2-b24a7b20ac37_1946x1276.png) [see that LoRA—***for both RoBERTa and DeBERTa***—is capable of producing results on](https://substackcdn.com/image/fetch/f_auto,q_auto:good,fl_progressive:steep/https%3A%2F%2Fsubstack-post-media.s3.amazonaws.com%2Fpublic%2Fimages%2F0ae43026-6b42-43e8-b0c2-b24a7b20ac37_1946x1276.png) [par with or better than end-to-end fnetuning; see above.](https://substackcdn.com/image/fetch/f_auto,q_auto:good,fl_progressive:steep/https%3A%2F%2Fsubstack-post-media.s3.amazonaws.com%2Fpublic%2Fimages%2F0ae43026-6b42-43e8-b0c2-b24a7b20ac37_1946x1276.png)

https://cameronrwolfe.substack.com/p/easily-train-a-specialized-llm-peft 26/46

14.04.2024 15:31 Easily Train a Specialized LLM: PEFT, LoRA, QLoRA, LLaMA-Adapter, and More

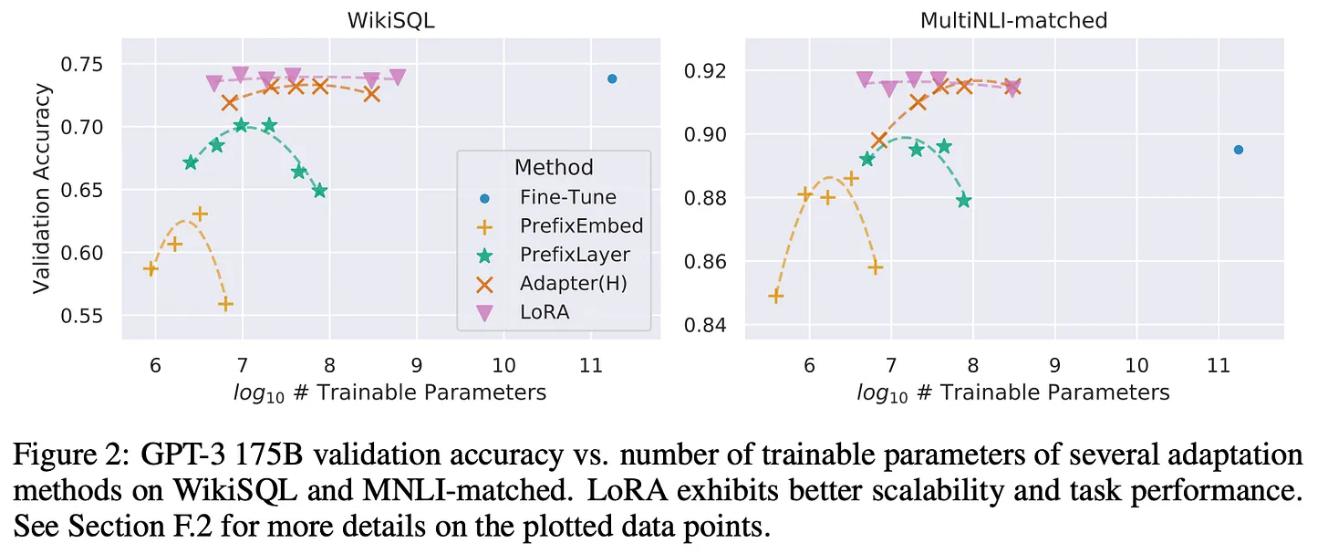


[(from [2])](https://substackcdn.com/image/fetch/f_auto,q_auto:good,fl_progressive:steep/https%3A%2F%2Fsubstack-post-media.s3.amazonaws.com%2Fpublic%2Fimages%2Ff6db182e-8c5f-4321-ac73-169d36e97320_1060x1114.png)

[When experiments are performed with generative LLMs, we see that LoRA handles](https://substackcdn.com/image/fetch/f_auto,q_auto:good,fl_progressive:steep/https%3A%2F%2Fsubstack-post-media.s3.amazonaws.com%2Fpublic%2Fimages%2Ff6db182e-8c5f-4321-ac73-169d36e97320_1060x1114.png) [these workloads well and is efective even with much larger models; see above.](https://substackcdn.com/image/fetch/f_auto,q_auto:good,fl_progressive:steep/https%3A%2F%2Fsubstack-post-media.s3.amazonaws.com%2Fpublic%2Fimages%2Ff6db182e-8c5f-4321-ac73-169d36e97320_1060x1114.png) [Notably, we see that LoRA matches or exceeds the performance of end-to-end](https://substackcdn.com/image/fetch/f_auto,q_auto:good,fl_progressive:steep/https%3A%2F%2Fsubstack-post-media.s3.amazonaws.com%2Fpublic%2Fimages%2Ff6db182e-8c5f-4321-ac73-169d36e97320_1060x1114.png) [fnetuning on every dataset that is tested. Furthermore, LoRA’s performance is](https://substackcdn.com/image/fetch/f_auto,q_auto:good,fl_progressive:steep/https%3A%2F%2Fsubstack-post-media.s3.amazonaws.com%2Fpublic%2Fimages%2Ff6db182e-8c5f-4321-ac73-169d36e97320_1060x1114.png) [incredibly stable with respect to the number of trainable parameters that are used,](https://substackcdn.com/image/fetch/f_auto,q_auto:good,fl_progressive:steep/https%3A%2F%2Fsubstack-post-media.s3.amazonaws.com%2Fpublic%2Fimages%2Ff6db182e-8c5f-4321-ac73-169d36e97320_1060x1114.png) [especially when compared to techniques like prefx tuning; see below.](https://substackcdn.com/image/fetch/f_auto,q_auto:good,fl_progressive:steep/https%3A%2F%2Fsubstack-post-media.s3.amazonaws.com%2Fpublic%2Fimages%2Ff6db182e-8c5f-4321-ac73-169d36e97320_1060x1114.png)

https://cameronrwolfe.substack.com/p/easily-train-a-specialized-llm-peft 27/46

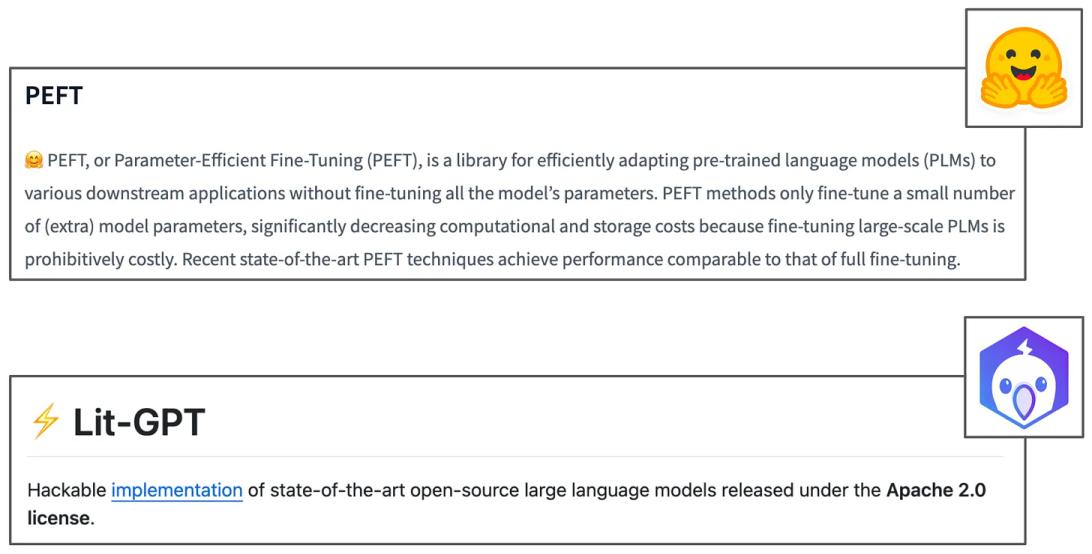
14.04.2024 15:31 Easily Train a Specialized LLM: PEFT, LoRA, QLoRA, LLaMA-Adapter, and More



[(from [1])](https://substackcdn.com/image/fetch/f_auto,q_auto:good,fl_progressive:steep/https%3A%2F%2Fsubstack-post-media.s3.amazonaws.com%2Fpublic%2Fimages%2Fec2329b8-b295-4f6a-bda1-51dfe56a40a9_1630x680.png)

[Put simply, LoRA can achieve impressive performance—***comparable to or beyond***](https://substackcdn.com/image/fetch/f_auto,q_auto:good,fl_progressive:steep/https%3A%2F%2Fsubstack-post-media.s3.amazonaws.com%2Fpublic%2Fimages%2Fec2329b8-b295-4f6a-bda1-51dfe56a40a9_1630x680.png)[***that of full fnetuning***—with very few trainable parameters, which minimizes I/O](https://substackcdn.com/image/fetch/f_auto,q_auto:good,fl_progressive:steep/https%3A%2F%2Fsubstack-post-media.s3.amazonaws.com%2Fpublic%2Fimages%2Fec2329b8-b295-4f6a-bda1-51dfe56a40a9_1630x680.png)bottlenecks, reduces memory usage, and speeds up the fnetuning process.

Using LoRA in Practice



Best libraries for finetuning with LoRA

The reason that LoRA has become so popular is because it is such a useful tool for AI practitioners. With LoRA, fnetuning LLMs for our desired application is much easier than before! We don’t need tons of massive GPUs and the fnetuning process

https://cameronrwolfe.substack.com/p/easily-train-a-specialized-llm-peft 28/46

14.04.2024 15:31 Easily Train a Specialized LLM: PEFT, LoRA, QLoRA, LLaMA-Adapter, and More

is efcient, which makes it possible for (almost) anyone to train a specialized LLM on their own data. Plus, numerous efcient implementations of LoRA are already available online with tons of useful features; e.g., [gradient accumulation](https://lightning.ai/blog/gradient-accumulation/) to reduce memory usage, [mixed-precision training](https://lightning.ai/docs/pytorch/1.5.9/advanced/mixed_precision.html) to speedup fnetuning, and easy integration with accelerators (GPUs/TPUs). Notable libraries that can be used to fnetune LLMs with LoRA (shown above) include:

PEFT by HuggingFace [[link](https://huggingface.co/docs/peft/index)]



Lit-GPT by [Lightning AI](https://lightning.ai/) [[link](https://github.com/Lightning-AI/lit-gpt)]



In this section, we will briefy overview how we can use Lit-GPT to fnetune an LLM with LoRA and provide some helpful tips for using LoRA in practice.

**Finetuning with LoRA.** The Lit-GPT library contains a variety of useful scripts that can be used to fnetune [open-source LLMs](https://cameronrwolfe.substack.com/p/the-history-of-open-source-llms-better) using LoRA; see [here](https://github.com/Lightning-AI/lit-gpt/blob/main/tutorials/finetune_lora.md) for a full guide. Afer cloning the Lit-GPT repository and installing dependencies, the frst step is to download a pretrained model to fnetune with LoRA. To download [LLaMA-2](https://cameronrwolfe.substack.com/p/llama-2-from-the-ground-up) (this requires being [granted access to LLaMA-2](https://huggingface.co/blog/llama2) via HuggingFace frst), we just ***i)*** download the model from HuggingFace and ***ii)*** convert this into the format needed for Lit-GPT. We can do this via the scripts shown below.



Download the pretrained LLaMA-2 model (HuggingFace version)

Afer we’ve downloaded the pretrained model, we need a dataset to use for fnetuning. Examples of popular [instruction tuning](https://blog.research.google/2021/10/introducing-flan-more-generalizable.html) datasets that are commonly used for LLM fnetuning include:

https://cameronrwolfe.substack.com/p/easily-train-a-specialized-llm-peft 29/46

14.04.2024 15:31 Easily Train a Specialized LLM: PEFT, LoRA, QLoRA, LLaMA-Adapter, and More

Alpaca [[link](https://github.com/tatsu-lab/stanford_alpaca)]



Dolly [[link](https://huggingface.co/datasets/databricks/databricks-dolly-15k)]



LongForm [[link](https://huggingface.co/datasets/akoksal/LongForm)]



LIMA [[link](https://huggingface.co/datasets/GAIR/lima)]



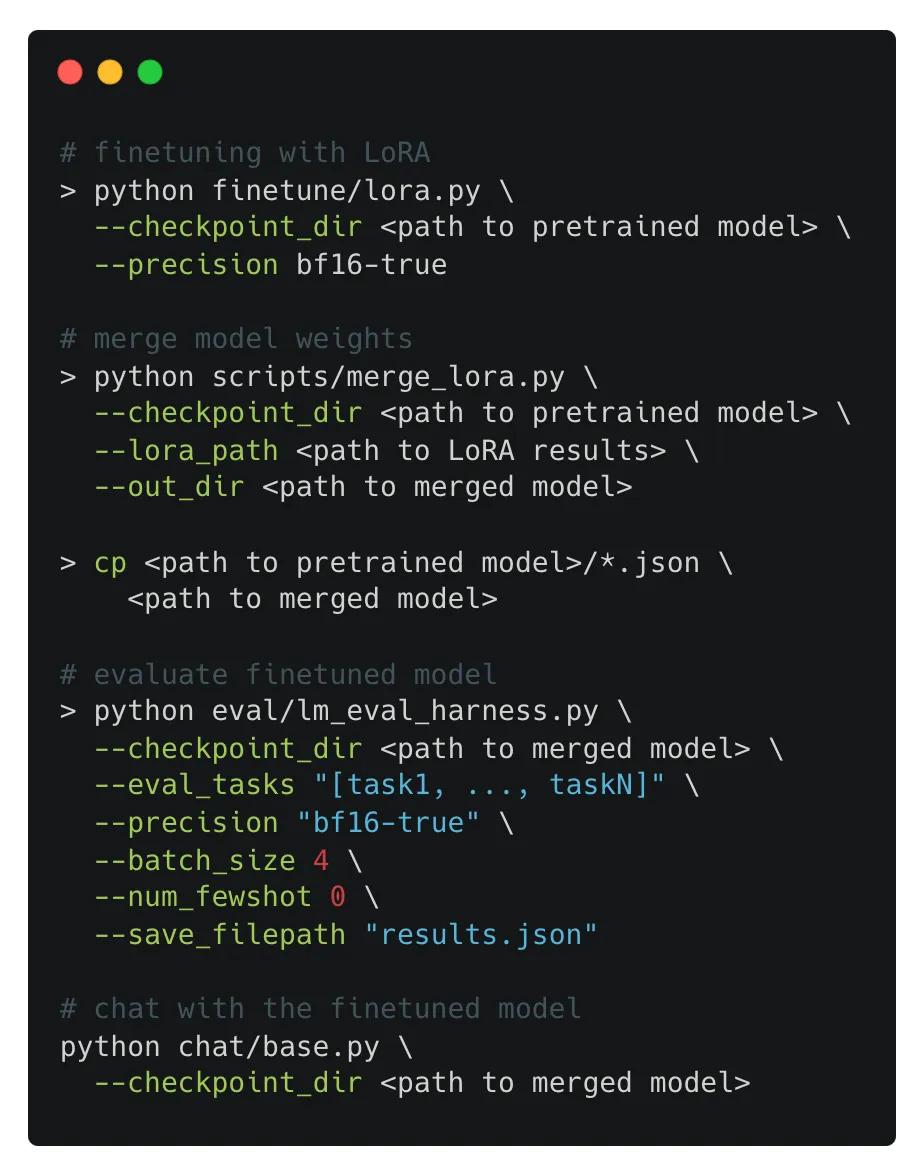
RedPajama [[link](https://huggingface.co/datasets/togethercomputer/RedPajama-Data-V2)]



To download (and properly format) any of these datasets, we can simply use the [helper scripts](https://github.com/Lightning-AI/lit-gpt/blob/main/tutorials/prepare_dataset.md) within Lit-GPT, which also support [creating our own](https://github.com/Lightning-AI/lit-gpt/blob/main/tutorials/finetune_lora.md#tune-on-your-dataset) (custom) fnetuning dataset. From here, all that’s lef is to run the fnetuning script, merge the model’s weights 11, and evaluate the resulting model, either on a set of specifed tasks or by just chatting with the model to assess quality; see below.

https://cameronrwolfe.substack.com/p/easily-train-a-specialized-llm-peft 30/46

14.04.2024 15:31 Easily Train a Specialized LLM: PEFT, LoRA, QLoRA, LLaMA-Adapter, and More

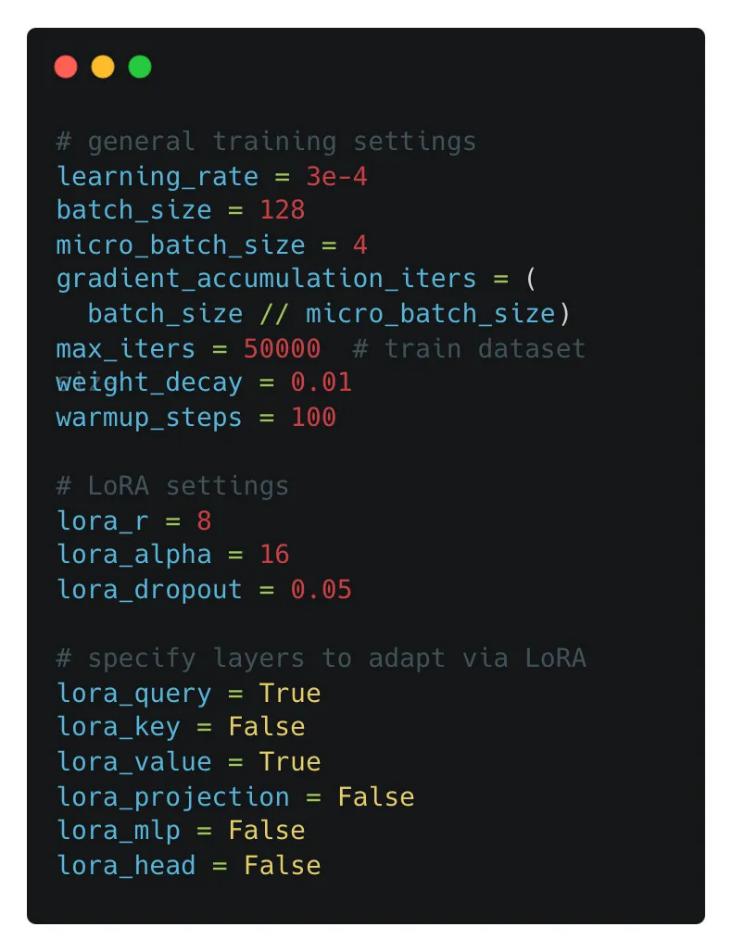


[Finetuning and evaluating a model with LoRA using Lit-GPT](https://substackcdn.com/image/fetch/f_auto,q_auto:good,fl_progressive:steep/https%3A%2F%2Fsubstack-post-media.s3.amazonaws.com%2Fpublic%2Fimages%2Fbca36552-6615-4a0b-89f1-162a0c6d2909_924x1176.png)

[The fnetuning script within Lit-GPT has several default confgurations that are](https://substackcdn.com/image/fetch/f_auto,q_auto:good,fl_progressive:steep/https%3A%2F%2Fsubstack-post-media.s3.amazonaws.com%2Fpublic%2Fimages%2Fbca36552-6615-4a0b-89f1-162a0c6d2909_924x1176.png) [used for LoRA; see below. We can edit these options in the finetune/lora.py fle](https://substackcdn.com/image/fetch/f_auto,q_auto:good,fl_progressive:steep/https%3A%2F%2Fsubstack-post-media.s3.amazonaws.com%2Fpublic%2Fimages%2Fbca36552-6615-4a0b-89f1-162a0c6d2909_924x1176.png) [prior to performing a fnetuning run. For example, we might want to change the](https://substackcdn.com/image/fetch/f_auto,q_auto:good,fl_progressive:steep/https%3A%2F%2Fsubstack-post-media.s3.amazonaws.com%2Fpublic%2Fimages%2Fbca36552-6615-4a0b-89f1-162a0c6d2909_924x1176.png) [value of r that is used, or apply LoRA to all layers within the transformer.](https://substackcdn.com/image/fetch/f_auto,q_auto:good,fl_progressive:steep/https%3A%2F%2Fsubstack-post-media.s3.amazonaws.com%2Fpublic%2Fimages%2Fbca36552-6615-4a0b-89f1-162a0c6d2909_924x1176.png)

https://cameronrwolfe.substack.com/p/easily-train-a-specialized-llm-peft 31/46

14.04.2024 15:31 Easily Train a Specialized LLM: PEFT, LoRA, QLoRA, LLaMA-Adapter, and More



[Settings for LoRA finetuning](https://substackcdn.com/image/fetch/f_auto,q_auto:good,fl_progressive:steep/https%3A%2F%2Fsubstack-post-media.s3.amazonaws.com%2Fpublic%2Fimages%2Fc68bb437-504c-48fe-b8fd-18b5ec32169e_790x1028.png)

[**Is LoRA just for LLMs?** Finally, it’s important to realize that LoRA is a general](https://substackcdn.com/image/fetch/f_auto,q_auto:good,fl_progressive:steep/https%3A%2F%2Fsubstack-post-media.s3.amazonaws.com%2Fpublic%2Fimages%2Fc68bb437-504c-48fe-b8fd-18b5ec32169e_790x1028.png) [technique that can be used for any type of dense neural network layer—***we can***](https://substackcdn.com/image/fetch/f_auto,q_auto:good,fl_progressive:steep/https%3A%2F%2Fsubstack-post-media.s3.amazonaws.com%2Fpublic%2Fimages%2Fc68bb437-504c-48fe-b8fd-18b5ec32169e_790x1028.png) ***fnetune more than just LLMs with LoRA***. For example, the link below shows an example of using LoRA to fnetune an image classifcation model.

Furthermore, we should notice that LoRA is orthogonal to most existing (parameter-efcient) fnetuning techniques, meaning that we can use both at the same time! ***Why is this the case?*** LoRA does not directly modify the pretrained model’s weight matrices, but rather learns a low-rank update to these matrices that can (optionally) be fused with the pretrained weights to avoid inference latency. ***This is an inline adaptation technique that adds no additional layers to the model***. As aresult, we can perform end-to-end fnetuning in addition to LoRA, as well as apply techniques like prefx tuning and adapter layers on top of LoRA.

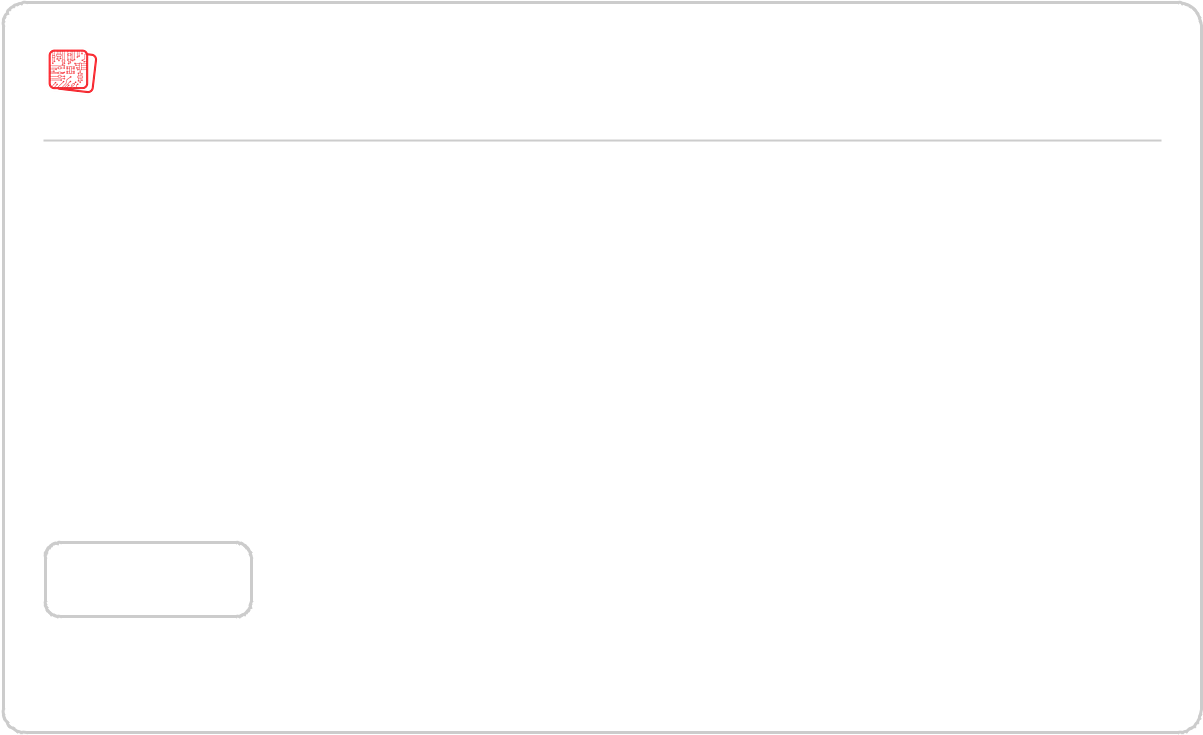
https://cameronrwolfe.substack.com/p/easily-train-a-specialized-llm-peft 32/46

14.04.2024 15:31 Easily Train a Specialized LLM: PEFT, LoRA, QLoRA, LLaMA-Adapter, and More



***“The principles outlined here apply to any dense layers in deep learning models, though we only focus on certain weights in Transformer language models in our experiments as the motivating use case.”*** - from [1]

**Further reading.** Within this section, we have only scratched the surface of how to use LoRA efectively in practice. Although this serves as a good starting point, there are so many details/fndings that one could gather from running experiments with LoRA and learning the best practices for this technique.



[Ahead of AI](https://magazine.sebastianraschka.com/p/practical-tips-for-finetuning-llms?utm_source=substack&utm_campaign=post_embed&utm_medium=web)

[Practical Tips for Finetuning LLMs Using LoRA (Low-Rank](https://magazine.sebastianraschka.com/p/practical-tips-for-finetuning-llms?utm_source=substack&utm_campaign=post_embed&utm_medium=web) [Adaptation)](https://magazine.sebastianraschka.com/p/practical-tips-for-finetuning-llms?utm_source=substack&utm_campaign=post_embed&utm_medium=web)

[Low-rank adaptation (LoRA) is among the most widely used and effective](https://magazine.sebastianraschka.com/p/practical-tips-for-finetuning-llms?utm_source=substack&utm_campaign=post_embed&utm_medium=web) [techniques for efficiently training custom LLMs. For those interested in open-source LLMs, it's an essential technique worth familiarizing oneself with. Last](https://magazine.sebastianraschka.com/p/practical-tips-for-finetuning-llms?utm_source=substack&utm_campaign=post_embed&utm_medium=web) [month, I shared an article with several LoRA experiments…](https://magazine.sebastianraschka.com/p/practical-tips-for-finetuning-llms?utm_source=substack&utm_campaign=post_embed&utm_medium=web)

[**Read more**](https://magazine.sebastianraschka.com/p/practical-tips-for-finetuning-llms?utm_source=substack&utm_campaign=post_embed&utm_medium=web)

[5 months ago · 119 likes · 16 comments · Sebastian Raschka, PhD](https://magazine.sebastianraschka.com/p/practical-tips-for-finetuning-llms?utm_source=substack&utm_campaign=post_embed&utm_medium=web)

To learn more about these details, I highly recommend the series of overviews written on LoRA by [Sebastian Raschka](https://twitter.com/rasbt) and the Lightning AI team:

Practical Tips for Finetuning LLMs Using LoRA (Low-Rank Adaptation) [[link](https://magazine.sebastianraschka.com/p/practical-tips-for-finetuning-llms)]



Finetuning LLMs with LoRA and QLoRA: Insights from Hundreds of Experiments [[link](https://lightning.ai/pages/community/lora-insights/)]



Finetuning Falcon LLMs More Efciently With LoRA and Adapters [[link](https://lightning.ai/pages/community/finetuning-falcon-efficiently/)] Parameter-Efcient LLM Finetuning With Low-Rank Adaptation (LoRA) [[link](https://lightning.ai/pages/community/tutorial/lora-llm/)]



https://cameronrwolfe.substack.com/p/easily-train-a-specialized-llm-peft 33/46

14.04.2024 15:31 Easily Train a Specialized LLM: PEFT, LoRA, QLoRA, LLaMA-Adapter, and More

Within these articles (especially the “***Practical Tips***” and “***Insights from Hundreds of Experiments***” articles), we will fnd a variety of practical takeaways that help us tobetter leverage LoRA for language model adaptation. Some of the best takeaways gathered from this extensive empirical analysis of LoRA include:

The choice of optimizer (i.e., [SGD](https://pytorch.org/docs/stable/generated/torch.optim.SGD.html) or [AdamW](https://pytorch.org/docs/stable/generated/torch.optim.AdamW.html)) for LoRA does not make a huge diference in performance or memory overhead (assuming r is small).



Performing multiple epochs of training over the fnetuning dataset is ofentimes not benefcial (i.e., degrades performance).



Applying LoRA across all weight matrices in the transformer is better than just applying LoRA to the query and value matrices, as proposed in [1].



Setting α to 2X the value of r yields competitive results. Larger values of r call for larger values of α, and r is a hyperparameter that must be tuned.



Going beyond LoRA, the same group of researchers has written a variety of practical overviews of other parameter-efcient fnetuning techniques (e.g., prefx tuning and adapter layers) as well:

Understanding Parameter-Efcient LLM Finetuning: Prompt Tuning And Prefx Tuning [[link](https://magazine.sebastianraschka.com/p/understanding-parameter-efficient)]



Understanding Parameter-Efcient Finetuning of Large Language Models:



From Prefx Tuning to LLaMA-Adapters [[link](https://lightning.ai/pages/community/article/understanding-llama-adapters/)]

Finetuning Large Language Models [[link](https://magazine.sebastianraschka.com/p/finetuning-large-language-models)]

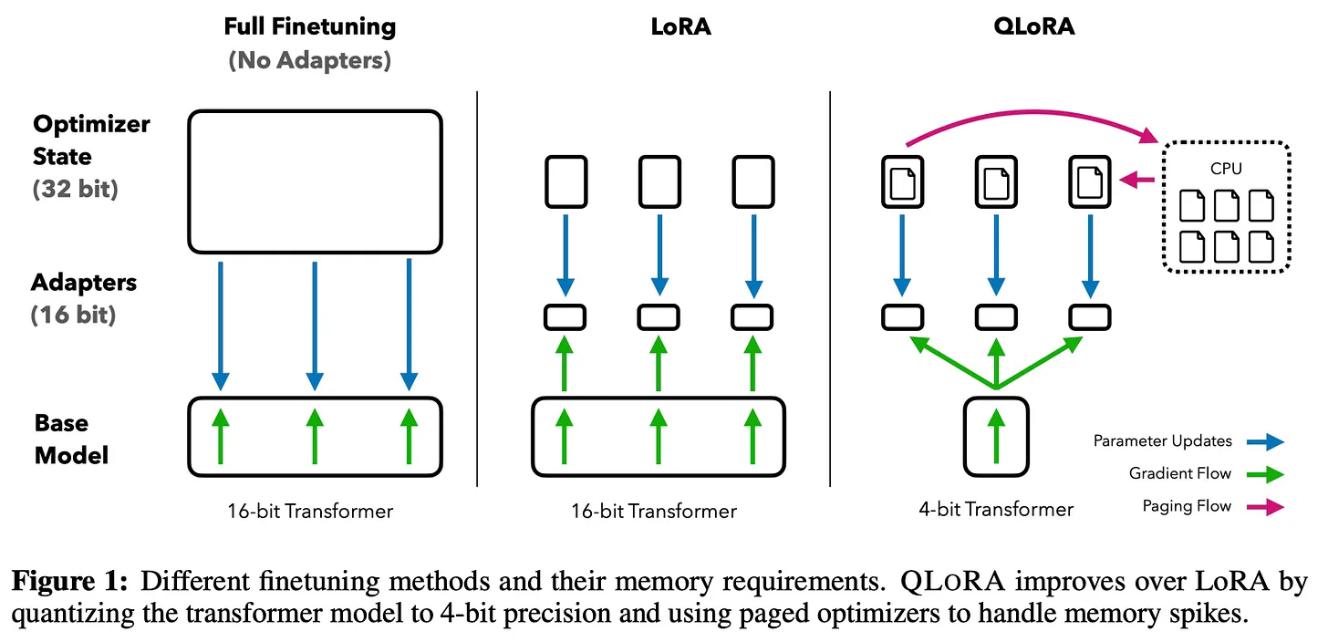


LoRA Variants (there are a ton…)

Due to its practical utility, the proposal of LoRA catalyzed the development of an entire feld of research devoted to parameter-efcient fnetuning—***and LoRA in particular***—that is quite active. Within this section, we will (attempt to) overviewmost of the notable LoRA variants that have been recently proposed.

https://cameronrwolfe.substack.com/p/easily-train-a-specialized-llm-peft 34/46

14.04.2024 15:31 Easily Train a Specialized LLM: PEFT, LoRA, QLoRA, LLaMA-Adapter, and More



[(from [19])](https://substackcdn.com/image/fetch/f_auto,q_auto:good,fl_progressive:steep/https%3A%2F%2Fsubstack-post-media.s3.amazonaws.com%2Fpublic%2Fimages%2Fed362d46-f789-4685-b4b7-87755f6fdcd2_1622x784.png)

**QLoRA [19]** (shown above) is arguably the most popular LoRA variant. At a high level, QLoRA uses [model quantization](https://cameronrwolfe.substack.com/p/quantized-training-with-deep-networks-82ea7f516dc6) to reduce memory usage during fnetuning with LoRA, while maintaining a (roughly) equal level of performance. More specifcally, QLoRA uses 4-bit quantization on the pretrained model weights and trains LoRA modules on top of this. Additionally, authors in [19] propose several novel quantization techniques for further reducing memory usage:

***4-Bit NormalFloat (NF4) Format***: a new (quantized) data type that works better for weights that follow a normal distribution.



***Double Quantization***: reduces memory footprint by quantizing both model weights and their corresponding quantization constants.



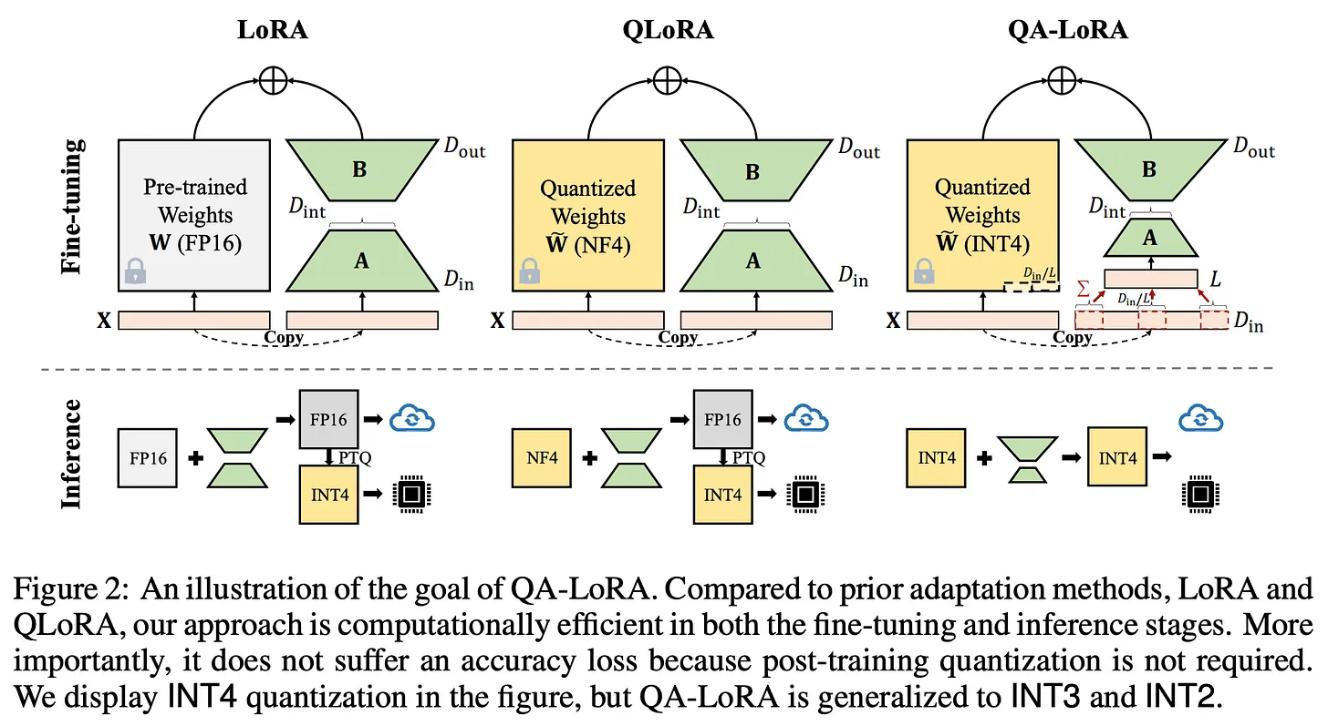
***Paged Optimizers***: prevents memory spikes due to gradient checkpointing that cause out-of-memory errors when processing long sequences or training a large model 12.



In practice, ***QLoRA saves memory at the cost of slightly-reduced training speed***. For example, we see [here](https://lightning.ai/pages/community/lora-insights/) that replacing LoRA with QLoRA to fnetune LLaMA-2-7B reduces memory usage by 33% but increases wall-clock training time by 39%.

https://cameronrwolfe.substack.com/p/easily-train-a-specialized-llm-peft 35/46

14.04.2024 15:31 Easily Train a Specialized LLM: PEFT, LoRA, QLoRA, LLaMA-Adapter, and More



(from [20])

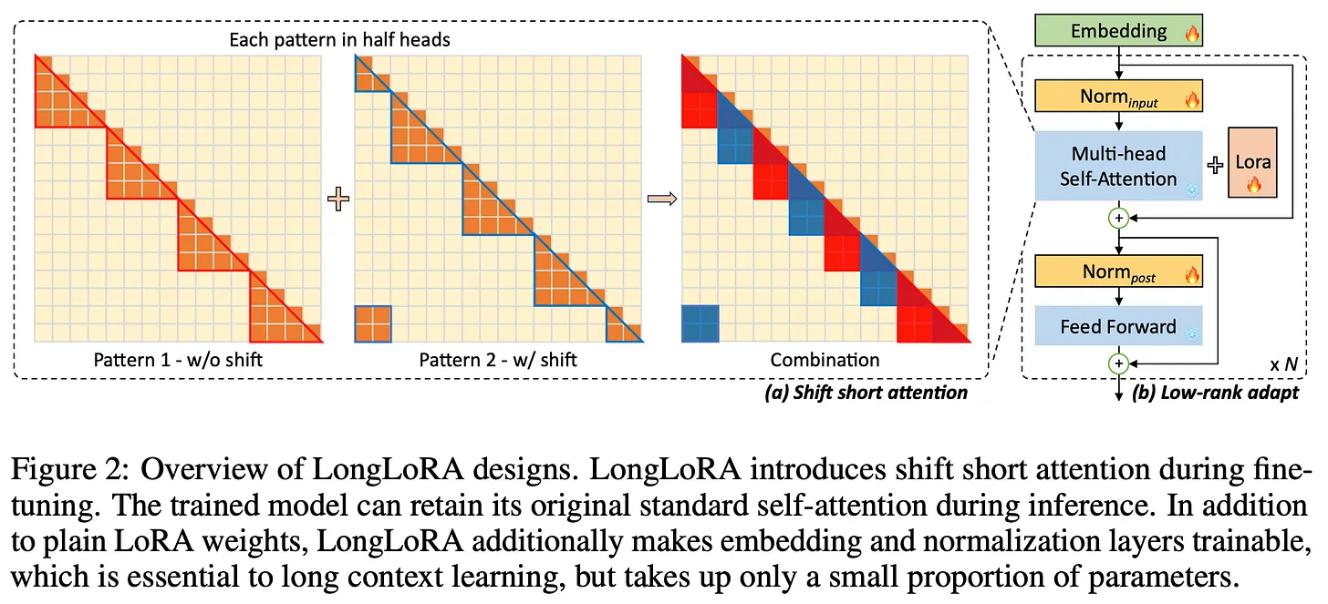
**QA-LoRA [20]** (shown above) is an extension of LoRA/QLoRA that further reduces the computational burden of training and deploying LLMs. Two major techniques are used to reduce the compute/memory overhead of fnetuning LLMs:

1. ***Parameter-Efcient Finetuning (PEFT)***: fnetune pretrained LLMs with a small number of trainable parameters (e.g., LoRA is one form of PEFT).
2. ***Quantization***: convert trained weights of an LLM into low-bit representations.

QA-LoRA integrates these two ideas in a simple and performant manner. To do this, we could perform post-training quantization on a model fnetuned with LoRA, but this approach has been shown to work poorly. Instead, QA-LoRA improves both training and inference efciency by proposing a group-wise quantization scheme that separately quantizes diferent groups of weights in the model. Because such quantization is applied during training, there is no need for post-training quantization—***QA-LoRA fnetunes in a quantization-aware manner***! Beyond QA-LoRA, LofQ [23] studies a similar idea of applying quantization and LoRA fnetuning on a pretrained model simultaneously.

https://cameronrwolfe.substack.com/p/easily-train-a-specialized-llm-peft 36/46

14.04.2024 15:31 Easily Train a Specialized LLM: PEFT, LoRA, QLoRA, LLaMA-Adapter, and More



(from [21])

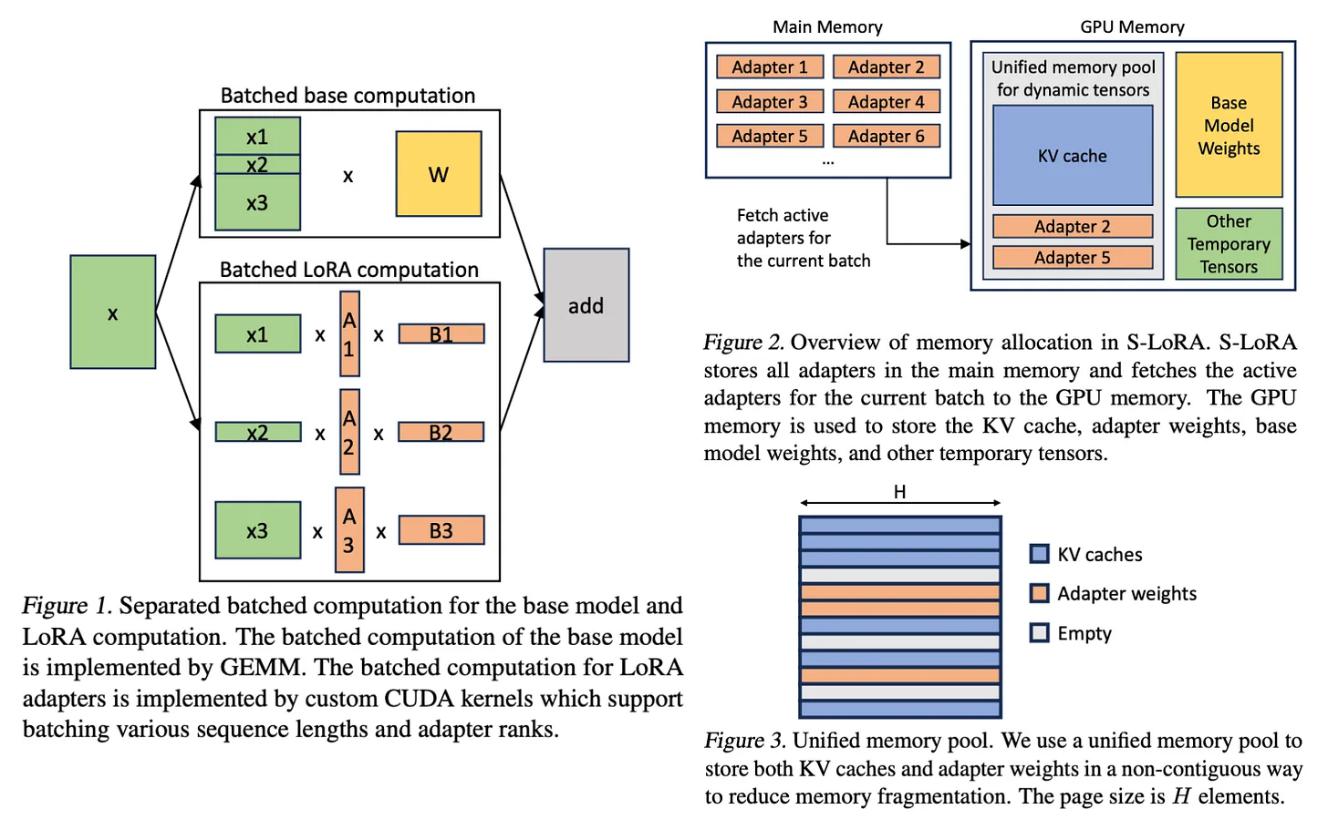
**LongLoRA [21]** attempts to cheaply adapt LLMs to longer context lengths using a parameter-efcient (LoRA-based) fnetuning scheme; see above. Training LLMs with long context lengths is expensive because the cost of self-attention is quadratic with respect to the length of the input sequence. However, we can avoid some of this cost by ***i)*** starting with a pretrained LLM and ***ii)*** expanding its context length via fnetuning. LongLoRA does exactly this, making the extension of a pretrained LLM’s context length via fnetuning cheaper by:

1. Using sparse local attention instead of dense global attention (optional at inference time).
2. Using LoRA (authors fnd that this works well for context extension).

Put simply, LongLoRA is just an efcient fnetuning technique that we can use to adapt a pretrained LLM to support longer context lengths.

https://cameronrwolfe.substack.com/p/easily-train-a-specialized-llm-peft 37/46

14.04.2024 15:31 Easily Train a Specialized LLM: PEFT, LoRA, QLoRA, LLaMA-Adapter, and More



[(from [22])](https://substackcdn.com/image/fetch/f_auto,q_auto:good,fl_progressive:steep/https%3A%2F%2Fsubstack-post-media.s3.amazonaws.com%2Fpublic%2Fimages%2Fbe989d13-c559-4542-b5d3-3bb3d8aba6d0_1732x1084.png)

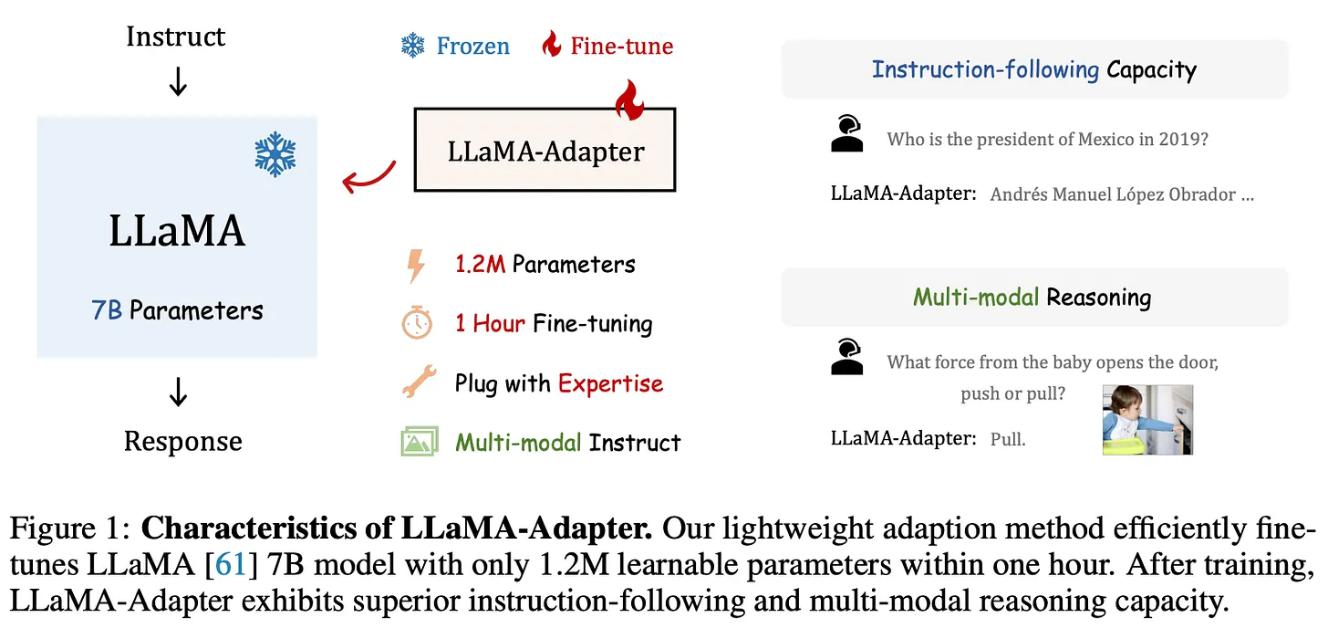
[**S-LoRA [22]** (shown above) aims to solve the problem of deploying multiple LoRA](https://substackcdn.com/image/fetch/f_auto,q_auto:good,fl_progressive:steep/https%3A%2F%2Fsubstack-post-media.s3.amazonaws.com%2Fpublic%2Fimages%2Fbe989d13-c559-4542-b5d3-3bb3d8aba6d0_1732x1084.png) [modules that are used to adapt the same pretrained model to a variety of diferent](https://substackcdn.com/image/fetch/f_auto,q_auto:good,fl_progressive:steep/https%3A%2F%2Fsubstack-post-media.s3.amazonaws.com%2Fpublic%2Fimages%2Fbe989d13-c559-4542-b5d3-3bb3d8aba6d0_1732x1084.png) [tasks. Put simply, S-LoRA is a scalable deployment strategy that:](https://substackcdn.com/image/fetch/f_auto,q_auto:good,fl_progressive:steep/https%3A%2F%2Fsubstack-post-media.s3.amazonaws.com%2Fpublic%2Fimages%2Fbe989d13-c559-4542-b5d3-3bb3d8aba6d0_1732x1084.png)

1. Stores all LoRA modules in main memory.
2. Puts modules being used to run the current query into GPU memory.
3. Uses unifed paging to allocate GPU memory and avoid fragmentation.
4. Proposes a new tensor parallelism strategy to batch LoRA computations.

By combining these techniques, S-LoRA can serve thousands of LoRA modules on a single GPU (or across multiple GPUs) and increases the throughput of prior systems (e.g., PEFT by HuggingFace) by up to 4X!

https://cameronrwolfe.substack.com/p/easily-train-a-specialized-llm-peft 38/46

14.04.2024 15:31 Easily Train a Specialized LLM: PEFT, LoRA, QLoRA, LLaMA-Adapter, and More



[(from [24])](https://substackcdn.com/image/fetch/f_auto,q_auto:good,fl_progressive:steep/https%3A%2F%2Fsubstack-post-media.s3.amazonaws.com%2Fpublic%2Fimages%2F7a67e87f-ba26-4017-8e8a-26c409d4a332_1616x764.png)

[**LLaMA-Adapter [24]** (shown above) is not based upon LoRA, but it is nonetheless a](https://substackcdn.com/image/fetch/f_auto,q_auto:good,fl_progressive:steep/https%3A%2F%2Fsubstack-post-media.s3.amazonaws.com%2Fpublic%2Fimages%2F7a67e87f-ba26-4017-8e8a-26c409d4a332_1616x764.png) recent (and popular) variant of parameter-efcient fnetuning for LLMs. At a high level, LLaMA-Adapter fnetunes pretrained LLMs to improve their instruction following capabilities using a very small number of added trainable parameters. When used to fne-tune LLaMA-7B, this approach only adds 1.2M parameters to the underlying model and requires less than an hour of fnetuning time 13. LLaMA-Adapter follows an approach similar to prefx tuning that adds a set of learnable task-adaptation prompts to the beginning of the transformers sequence at each layer. Such an approach preserves the pretrained model’s knowledge and allows adaptation to new tasks and instruction following (even including multi-modal instruction following!) to be learned with minimal data.

**Many, many more…** Although several notable LoRA variants have been covered above, there are a nearly limitless number of extensions to this technique that have been proposed. Other recent LoRA-inspired works include:

[***LQ-LoRA***](https://arxiv.org/abs/2311.12023) [26]: uses a more sophisticated quantization scheme within QLoRA that performs better and can be adapted to a target memory budget.



https://cameronrwolfe.substack.com/p/easily-train-a-specialized-llm-peft 39/46

14.04.2024 15:31 Easily Train a Specialized LLM: PEFT, LoRA, QLoRA, LLaMA-Adapter, and More

[***MultiLoRA***](https://arxiv.org/abs/2311.11501) [27]: extension of LoRA that better handles complex multi-task learning scenarios.



[***LoRA-FA***](https://arxiv.org/abs/2308.03303) [28]: freezes half of the low-rank decomposition matrix (i.e., the A matrix within the product AB) to further reduce memory overhead.



[***Tied-LoRA***](https://arxiv.org/abs/2311.09578) [29]: leverages weight tying to further improve the parameter efciency of LoRA.



[***GLoRA***](https://arxiv.org/abs/2306.07967) [30]: extends LoRA to adapt pretrained model weights and activations to each task in addition to an adapter for each layer.



Given that so many LoRA-inspired techniques exist, there are probably a few notable extensions that are missing from the list above. If you are aware of any other techniques that are worth including, let me know in the comments!

Takeaways

We should now have a working understanding of LoRA, the several variants of this technique that have been proposed, and how these ideas can be applied in practice. LoRA is arguably the most widely-used practical tool for creating specialized LLMs, as it democratizes the fnetuning process by signifcantly reducing hardware requirements. Some important takeaways are outlined below.

**Training an LLM.** The training process for language models (i.e., both encoder-only and decoder-only models) includes pretraining and fnetuning. During pretraining, we train the model via a self-supervised objective over a large amount of unlabeled text. Although pretraining is expensive, we can reuse the resulting model numerous times as a starting point for fnetuning on various tasks. Due to the public availability of many high-quality pretrained LLMs, most practitioners can simply download a pretrained model and focus upon the fnetuning process without ever having to pretrain a model from scratch.

**Afordable fnetuning.** The fnetuning process is cheap relative to the cost of pretraining. However, modern LLMs (especially GPT-style models) have many

https://cameronrwolfe.substack.com/p/easily-train-a-specialized-llm-peft 40/46

14.04.2024 15:31 Easily Train a Specialized LLM: PEFT, LoRA, QLoRA, LLaMA-Adapter, and More

parameters. As such, we need expensive hardware (i.e., GPUs with a lot of memory) to make the fnetuning tractable, thus increasing the barrier to entry for fnetuning an LLM. Several parameter-efcient fnetuning approaches have been proposed as a solution to this issue, but one of the most widely-adopted strategies is LoRA, which injects a learnable low-rank weight update into each layer of the underlying model. LoRA minimizes the memory overhead of fnetuning—***thus reducing hardware requirements***—and performs comparably to full fnetuning.

***“QLORA reduces the average memory requirements of fnetuning a 65B parameter model from >780GB of GPU memory to <48GB without degrading the runtime or predictive performance compared to a 16- bit fully fnetuned baseline.”*** - from [19]



**An ecosystem.** LoRA is a practically useful tool that gives (almost) anyone the power to train a specialized LLM over their data. As a result, LoRA has been widely studied within the AI research community, leading to a variety of extensions, alternatives, and practical tools to go along with it. One of the most notable extensions is QLoRa, which combines LoRA with model quantization to further reduce the memory overhead of LLM fnetuning. However, this reduction in memory overhead comes at the cost of a slight decrease in training speed.

New to the newsletter?

Hi! I’m [Cameron R. Wolfe](https://cameronrwolfe.me/), deep learning Ph.D. and Director of AI at [Rebuy](https://www.rebuyengine.com/). This is the Deep (Learning) Focus newsletter, where I help readers understand AI research via overviews of relevant topics from the ground up. If you like the newsletter, please subscribe, share it, or follow me on [Medium](https://medium.com/@wolfecameron), [X](https://twitter.com/cwolferesearch), and [LinkedIn](https://www.linkedin.com/in/cameron-r-wolfe-ph-d-04744a238/)!

Bibliography

[1] Hu, Edward J., et al. "Lora: Low-rank adaptation of large language models." ***arXiv preprint arXiv:2106.09685*** (2021).

[2] Houlsby, Neil, et al. "Parameter-efcient transfer learning for NLP." ***International Conference on Machine Learning***. PMLR, 2019.

https://cameronrwolfe.substack.com/p/easily-train-a-specialized-llm-peft 41/46