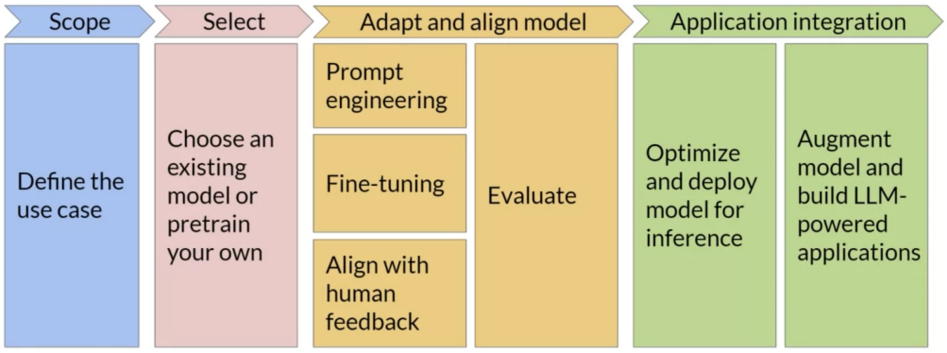
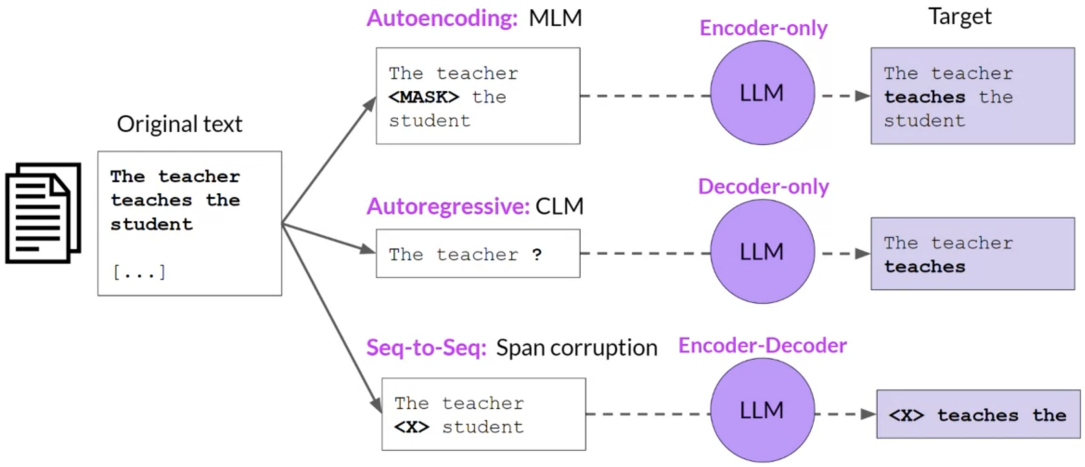
Wednesday, 21 October 2026

Pre-training LLMs & Scaling Laws

Generative AI project lifecycle



Based on the representation above we will be going through this module.



- Three Transformer Architectures and Their Pretraining Objectives

Quantization is covered in Basics of Training Optimization .

Multi GPUs efficient compute

There are 2 scenarios when train models on multiple GPUs

1 - Model parameters fit entirely on a single GPU

2 - Model parameters doesn’t fit on a single GPU

For scenario 1 we use

DDP (Distributed Data Parallel)

Its a form of model training where we where we have multiple GPUs and each GPU can store the full parameters of the model,

Steps for the DDP
1- parameter distribution
    we distribute the same copy of the model to each gpu
2- Forward pass
    each copy of the model is provided with their own batch to do the forward pass and calculate the loss
3-Backprop
    Every model calculates the gradients according to the params
4- Synchronisation
    We synchronise models by taking over the average of the gradients by each model and then we update the weights according to those average gradients, this way the model is being trained and every gpu remains synchronised

For scenario 2 we use

FSDP (Fully Sharded Data Parallel)

Its a form of Model training where Models parameters dosent fit entirely on a single GPU so you have to use FSDP

- it divides the model parameters into shards
- Its distributes the models optimization state, gradients and parameters to all the available GPUs.

Forward pass -

every GPU gathers the shards of a single layer and performs the forwards pass and then stores its activations and gathers another layer till we get the final output.

Backward pass -

Reconstruct parameters as needed and calculate gradients in reverse layer order, and combine the gradient across the GPUs and distribute the reduced gradient shards.

Optimization step -

Update the local parameter and optimizer-state shards. Repeat with the next batch.

Note - each gpu must be capable of holding all shards of a single layer while a single shard of the every other layer, plus their gradients and optimiser states (Two floating point integers per parameter)

*The explanation above for DDP and FSDP are for intuitive understanding to get the full understand you can read - What is Distributed Data Parallel, Getting Started with Fully Sharded Data Parallel.

LLM Scaling Laws

tldr -> ~ 20 tokens per param

Chinchilla paper

Training Compute-Optimal Large Language Models

Jordan Hoffmann*, Sebastian Borgeaud*, Arthur Mensch*, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals and Laurent Sifre*

*Equal contributions

We investigate the optimal model size and number of tokens for training a transformer language model under a given compute budget. We find that current large language models are significantly under-trained, a consequence of the recent focus on scaling language models whilst keeping the amount of training data constant. By training over 400 language models ranging from 70 million to over 16 billion parameters on 5 to 500 billion tokens, we find that for compute-optimal training, the model size and the number of training tokens should be scaled equally: for every doubling of model size the number of training tokens should also be doubled. We test this hypothesis by training a predicted compute-optimal model, *Chinchilla*, that uses the same compute budget as *Gopher* but with 70B parameters and 4× more more data. *Chinchilla* uniformly and significantly outperforms *Gopher* (280B), GPT-3 (175B), Jurassic-1 (178B), and Megatron-Turing NLG (530B) on a large range of downstream evaluation tasks. This also means that *Chinchilla* uses substantially less compute for fine-tuning and inference, greatly facilitating downstream usage. As a highlight, *Chinchilla* reaches a state-of-the-art average accuracy of 67.5% on the MMLU benchmark, greater than a 7% improvement over *Gopher*.

Jordan et al. 2022

According to Chinchilla scaling laws you should be training your model on ~ 20x the number of parameters

---

Pretraining for domain adaptation

If the vocabulary of the work you are doing or the information that you are working with is very niche you would need domain adaptation, which is to train the model on the specific data for the specific job.
For example - creating a language model to solely answer the questions on law, you would need domain adaptation.
Ex - Bloomberg gpt
It was trained on 51% financial data and 49% public data.