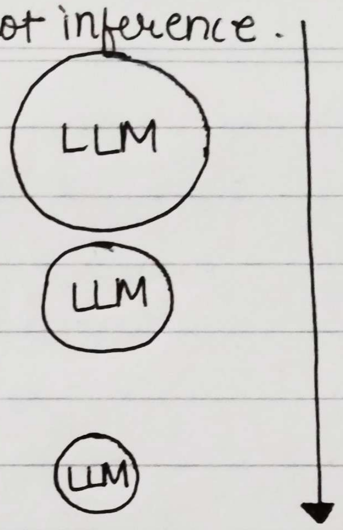
GENERATIVE AI LLMs

1) In context learning - zero shot inference.

(Majority models) ⇒ Providing no example.



Size of model (LLM) decreases.

2) 1 shot inference

⇒ Providing 1 example.

3) few shot inference

⇒ Providing few examples.

[context window: few thousand words → remember limitation]

configuration parameters

1] Max token :- limit the no. of tokens generated by model.

2] greedy :- The word with the max. probability is selected.

3] random sampling :- The word is picked randomly (can be of less probability) the generated sentence can be more creativ

4] top-k :- If k = 4, the top 4 more probable words are selecte from which one is picked. (randomly)

5] top-p :- a word is picked randomly from a group of words whose sum results in the ~~prop~~ probability p.

6] temperature :- If temp is set low, then the randomness is decreased else if it's set high, randomness increases.

LIFECYCLE OF Generative AI project :-

Define a use-case.

⇩

Choose an existing model or
pretrain your own model

⇩

Adapt & align model.

1) Prompt Engg.
2) Fine tuning.
3) Align with human feedback
↑
Reinforcement learning.

* Evaluate

⇩

Application integration
(optimization)
Additional infra?
Augument model & build
LLM-powered applications.

☐ Autoencoding models (Only encoder models)

→ Enable masked language modeling.

→ Objective: Reconstruct text ("denoising").
A word is randomly masked and is predicted by taking bidirectional context.

→ Sentiment analysis
→ NER
→ word classification

E.g. BERT
ROBERTA

☐ Autoregressive models (Only decoder models).

→ Causal L.M.

→ Objective: Predict next token. (Unidirectional context).
→ Text generation
→ Other emergent behaviour

E.g: GPT
BLOOM.

☐ Seq.-to-Seq model (Encoder-Decoder LLM)
Encoder uses span corruption to mask random words & Together words (masked) → Sentinel token.
Objective: Reconstruct span.

<X> words.
↓
Sentinel token

→ Translation
→ Text summarise
→ Q & A .

e.g: T5, BART

Reducing memory usage :

1] Quantization:

from 32-bit floating point to 16-bit floating point / 8-bit int .
FP32 → 4 bytes of memory
FP16 → 2 bytes of memory .
8-bit int → 1 byte → But loss of precision

/ BFLOAT16 / BF16 ⇒ popular choice .
2 bytes memory but increases speed & calculation
(Truncated FP32)
Supported by newer GPU's nvidia's A100 .

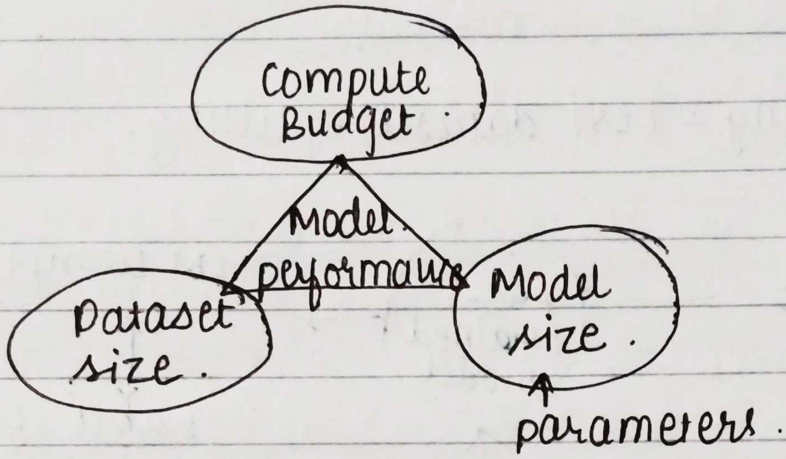
(Reduce required memory to store & train models) .
16-bit or 8 bit Quantization to train the model on a single GPU .

Model performance

↑ datasets
↑ parameters
} → Minimize loss .

→ Bigger models take more compute resources to train
→ They also take huge amount of data.

↑ compute budget → Loss decreases.



⇒ For domain adaptation we must pretrain the model from scratch.

Instruction Fine-tuning:

ICL may not work for small models.

~~Fine-tuna~~

Fine-tuning:

prompt-completion pairs are supplied to given ~~tex~~ pre-trained model

IFT, where all of model's weights are updated is called full-fine tuning.

Prompt template lib → fine tuning.

Prepared instruction dataset.

↓

~~Pre~~
Training & test dataset splitting.

↓

Prompt → pre-trained model → LLM completion.

↕

Label (from training data).

→ compute loss.

Pre trained model → Fine tuned LLM

* Fine-tuning on single task.

- <u>catastrophic forgetting</u>:
Fine-tuning can significantly increase the ability to perform the required task. But if the model previously knew how to carry out NER, it can forget that ability.

Only one task → No need to worry.

→ Fine-tune over multiple tasks at same time.

→ PEFT (Para. Eff. Fine-tuning) → Retains earlier weights → trains only small no. of task-specific adapter layers and params.

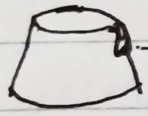
- You can use regularization to reduce the amount of weights to be trained in a model.

- Multi task fine tuning:

→ Lot of examples for training

→ FLAN model refers to sp. set of ins. used to perform ins. fine tuning.

(Fine-tuned LAnguage Net) ; FLAN-T5 , FLAN - PALM



→ dessert.

→ last step of training process.

- LLM Evaluation Challenges.

$$\text{Accuracy} = \frac{\text{correct prediction}}{\text{Total predictions}}$$

ROUGE

Text summarization

compares summary to one or more reference summaries.

BLEU SCORE

Text translation.

compares to human generated translations.

ROUGE-1

one word

$$\text{ROUGE-1 Recall} = \frac{\text{unigram matches}}{\text{unigrams in reference}}$$

$$\text{ROUGE-1 Precision} = \frac{\text{unigram matches}}{\text{unigrams in output (gen.)}}$$

$$\text{ROUGE-1 F1:} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

↓

Harmonic mean.

grp of two words.

By using bigrams you can get a better understanding of the accuracy. (ROUGE-2)

ROUGE-L score can be used to compare Long common subsequences in the reference & generated output.

↓

Used for same task.

BLEU SCORE:

BLEU metric = Avg(precision across range of n-gram sizes)

Model Evaluation

GLUE, SuperGLUE,

↓

Both have leaderboard version → Followed the benchmarks but didn't do well subjectively for a part. tasks.

compare & contrast LLM models.

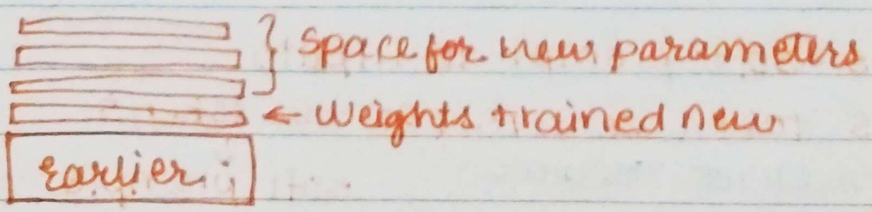
Benchmarks for massive models:

1) MMLU
2) BIG-bench ⇒ 204 tasks.
3) HELM

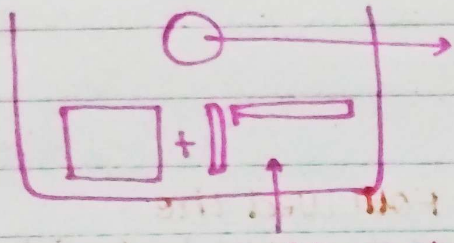
PEFT:
- → Selective
- → Reparameterization → LoRA
- → Additive → Adapters, Soft prompts.

→ Some techeniques of PEFT fine tune small trainable layers of model, while others add new trainable layers keeping ~~rest of~~ the earlier layers frozen.

→ Less prone to catastrophic forgetting.



★ LORA :



Passing through self-attention layer is enough.
can be used to train the model for various tasks.

	Base model	Full-fine tune		LoRA fine tune
FLAN-T5 dialog summarization		did well	⟷ Almost same	did well ✓

★ soft promps:

Prompt tuning adds soft prompt to inputs.
they have same len as token vectors. 20-100 tokens → soft prompts

Quantization + LoRA = QLoRA → much less resources → Memory, disk, CPU, GPU ↓↓↓↓