Department of Computing

Imperial College of Science, Technology and Medicine

Deep learning for GPU poor

Author: Anton Zhitomirsky

Contents

1 Calculating statistics of a model

1.1 Foundation model parameter count

Commonly when you look online for models, you will see their parameter counts. *Transformers are typically described by the #parameters* which impacts the computational requirements to store and run these models.

1.2 Floating Point System

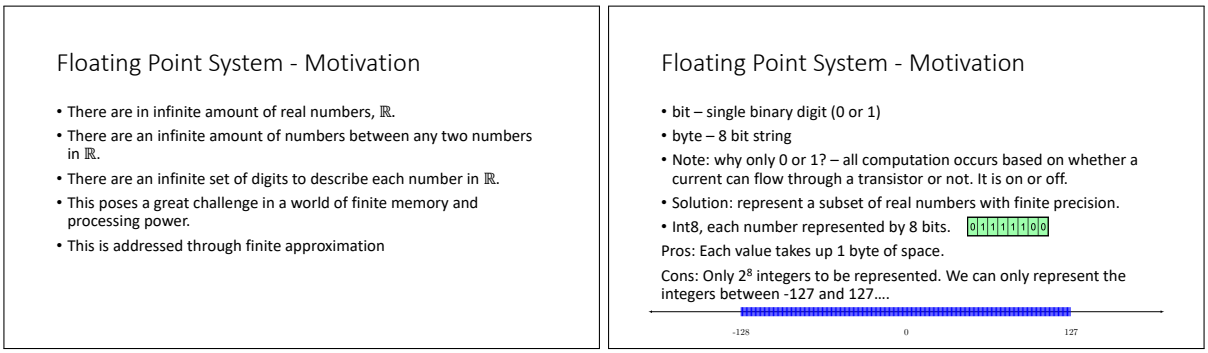
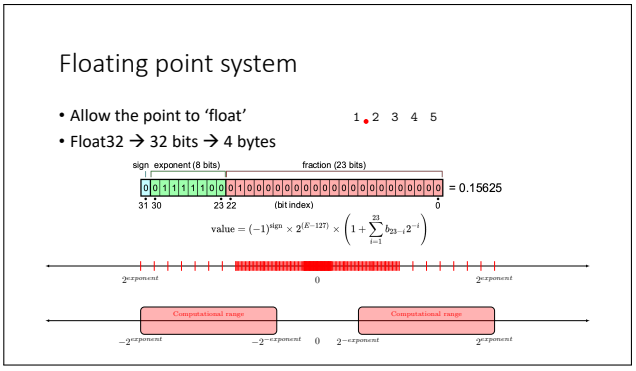
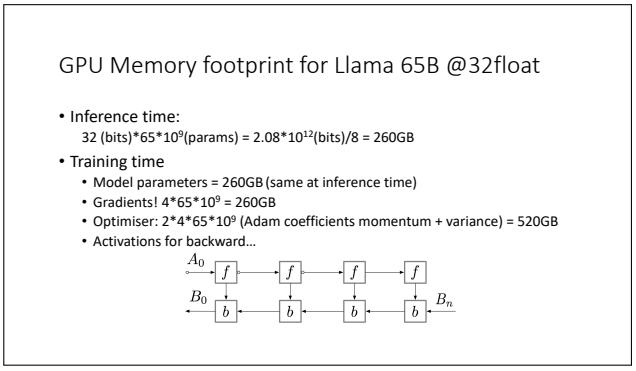


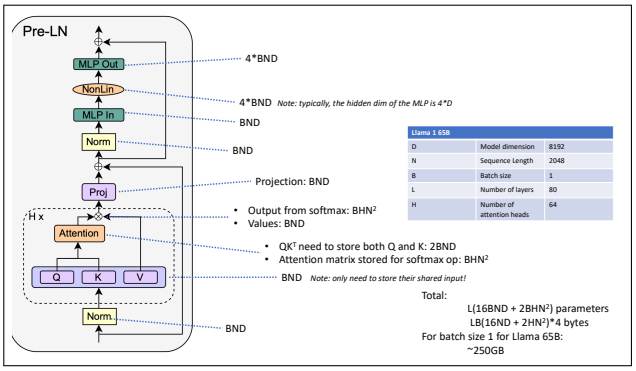
Figure 1: We store data in computers in bits with transistors. The more bits, the more precision.



- **Floating point** is a way to represent numbers in a computer where the decimal point can float.
- A certain section is allocated to represent the precision (actual number) the exponent and the sign.
- still has the problem of underflow or overflow.



- At inference time, just to store the data it is already 260GB
- At training time, we also require gradients and optimiser values (momentum, etc) which can be 10x the size of the model and the activations.



So if we pass through the network, we need to store the inputs to every single layer that theres a change in the computational graph. Here:

- The first is the batch times the number of tokens or the number of patches times by that dimensionality.
- For the query key value, they all get the same value.
- The attention matrix requires query with key-values
- We also need to store the n^2 complexity of the attention matrix.

- Also we store the result of the soft max which is the batch times the number of heads times the sequence dimension.
- With multi-head attention we usually have another layer that learns how to combine these values together.
- All in all, we have 250GB for a batch size of 1.

GPU Memory footprint for Llama 65B @32float

- Inference time:
  32 (bits)*65*10^9(params) = 2.08*10^{12}(bits)/8 = 260GB
- Training time
  - Model parameters = 260GB (same at inference time)
  - Gradients! 4*65*10^9 = 260GB
  - Optimiser: 2*4*65*10^9 (Adam coefficients momentum + variance) = 520GB
  - Activations for backward assuming batch size of 1 = 250GB
- Activations for backward pass are strongly dependent on sequence length and batch size
- Model params + Gradients + optimizer + activations = 1290GB
- 25 RTX 6000s ! – just for a batch size of 1….

1.3 Floating Point Operations (FLOPs)

Floating point operations (FLOPs)

- Basic unit of computation
- e.g. addition, subtraction, multiplication, divide of 2 floating point numbers

Examples

Given $\boldsymbol{w}, \boldsymbol{x} \in \mathbb{R}^n$

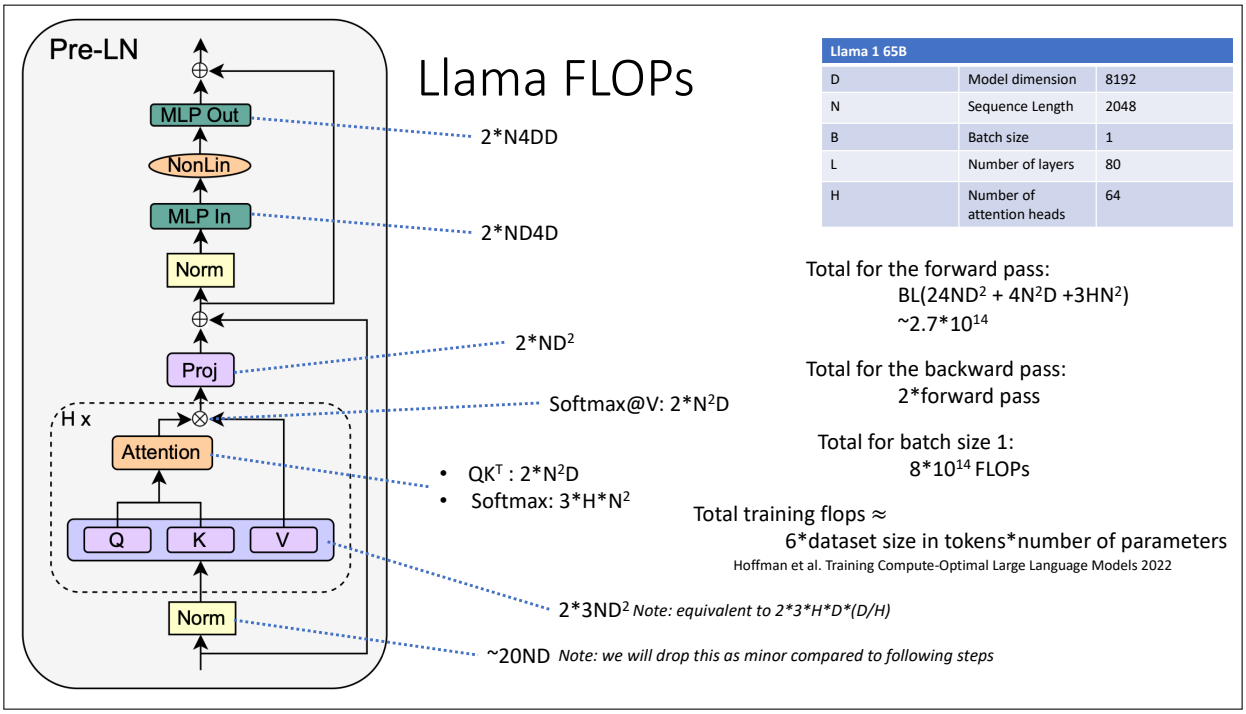
- $\boldsymbol{w} + \boldsymbol{x}$:
  n FLOPs
- $\boldsymbol{w}^T \boldsymbol{x}$:
  2n FLOPs (multiplication + (n-1) addition)

Given $\boldsymbol{W} \in \mathbb{R}^{m \times p}$, $\boldsymbol{X} \in \mathbb{R}^{p \times n}$

- $\boldsymbol{W}\boldsymbol{X}$:
  2nmp

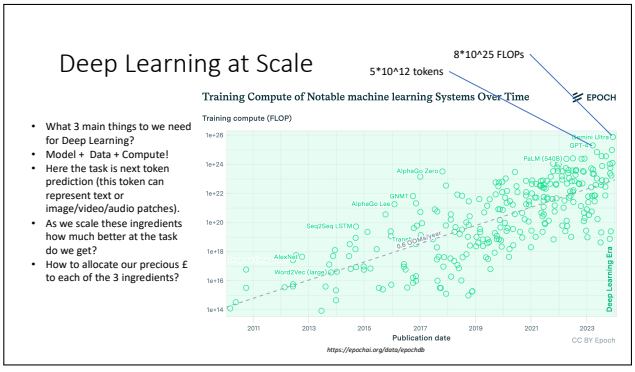
Computational requirements:

- given two vectors, if we add this will reuqire n operations.
- For the dot products, there is $2n$ flops.
- For the matrix multiplication, there is $2nmp$ multiplications

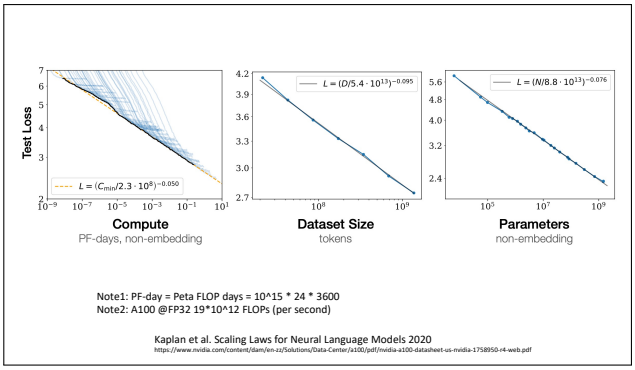


1. the layer norm is roughly 20 operations for every feature and token
2. for the query key values we have it 3 times, and 2 becuase its a dot product.
3. The softmax: for every single query we dot product it for every single value.
4. projectino is same as query key values (but only 1)

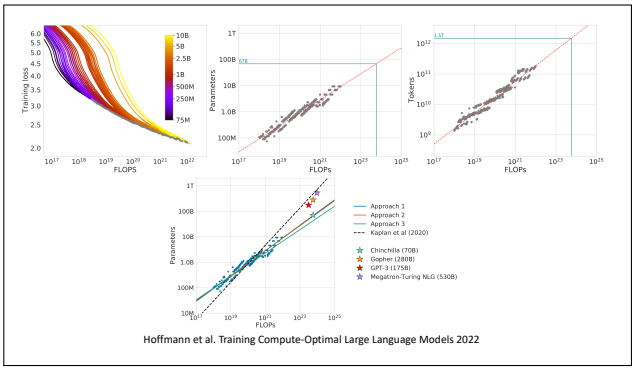
2 Deep Learning At Scale



- For deeplearning we need model data and compute.
- At scale, you can't train the model many times for hyper-parmaeter tuning. A single model may take many dollars to train just once.

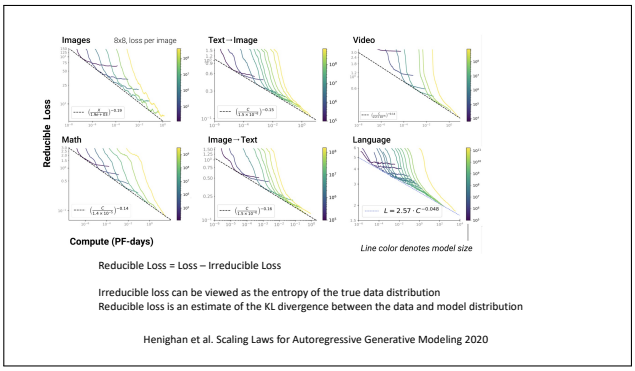


"When they have limitless compute and limitless datasets, they saw this really nice power law trend where it sticks very closely to the trend. This says that generally you can predict quite well, if you have limited compute and limited dataset size what loss you would get at the end." The values here were proven to be wrong because the smaller models were not fully optimized; they used the same learning rate schedule as they did for the larger models. Therefore, the line was too steep.



Hoffmann et al. Training Compute-Optimal Large Language Models 2022

"openAI published finidngs about colour representing the size of the model, and they fit for given flops and compute budgets which model gets the lowest loss." Here, we see that chinchilla is favoured since it prefers the number of data points over the model size.



Reducible Loss = Loss – Irreducible Loss

Irreducible loss can be viewed as the entropy of the true data distribution
Reducible loss is an estimate of the KL divergence between the data and model distribution

Henighan et al. Scaling Laws for Autoregressive Generative Modeling 2020

Here we see that 'deep learning has bene sovled'. "This is showing that this happens in all modalities; these losses can just keep going until we get to super human performance."

Definition 2.1 (Reducible Loss)**.** It is the actual loss - the irreductible loss, where irreducible loss is viewed as the true entropy of the data i.e. if I had a perfect fit of the data, what would be my loss even if I had a perfect fit?
It can be viewed as the KL divergence between the mainfold of the dataspace and the manafold of the learnt model.

SORA

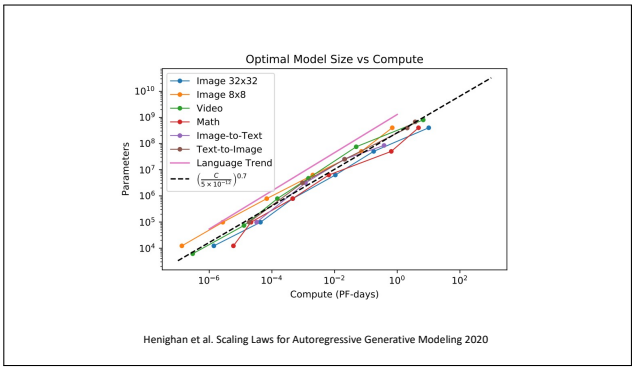
Base compute

4*compute

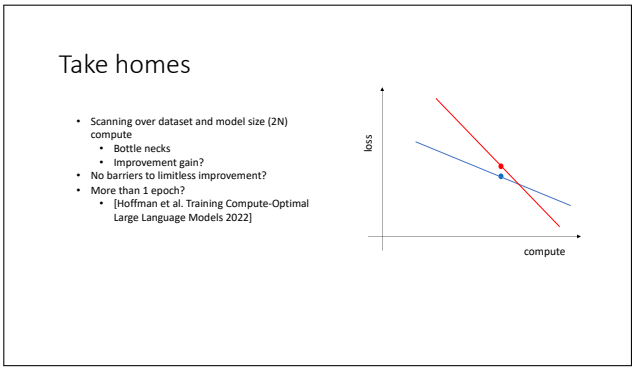
16*compute

https://openai.com/research/video-generation-models-as-world-simulators

Figure 2: SORA AI just used more compute!



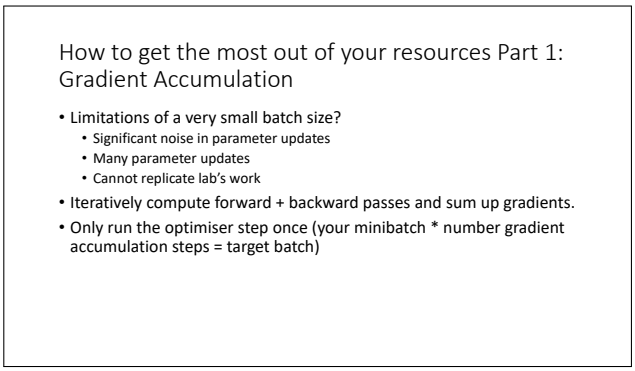
"When we plot all modalities the number of compute days vs the number of parameters for the optimal setting. There are correlation between what it is like to compute potimal size model for videa and for language model"



- if blue is state of the art and red is my implementation,
- then we can do a scaling analysis and conclude that since the curve is steeper then it may perform better at high levels of compute.

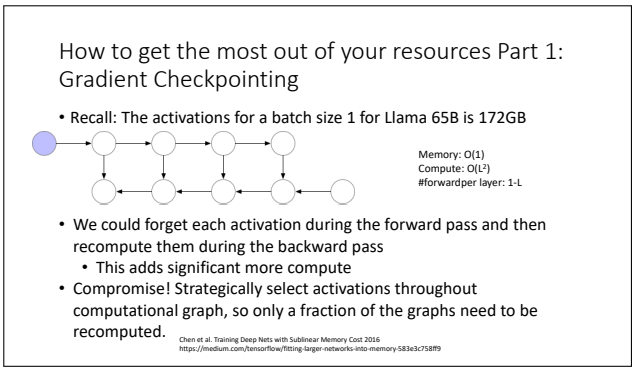
3 Methods for reducing computational requirements

3.1 Gradient Accumulation

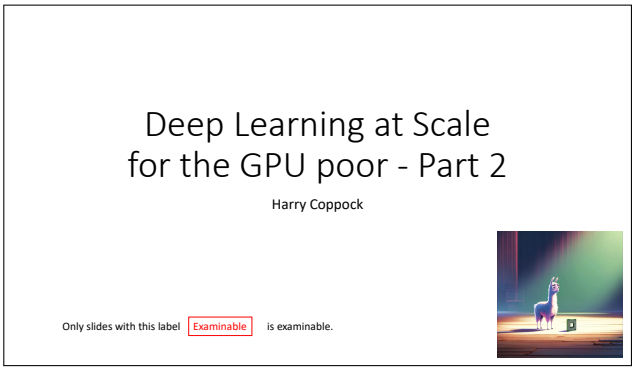


- As we increase the batch size our computation requirements increase
- If we use 1 batch, this is just Stochastic Gradient Descent so you're approximating the true gradient with one sample; your loss will be jumping around wihtout convergence
- *accumulate the gradients without updating the optimizer, then when the mini-batch size times the number of cumulative gradient accumulation steps equals the target batch then we do the step in the loss surface.*
- Useful if you want a larger batch size

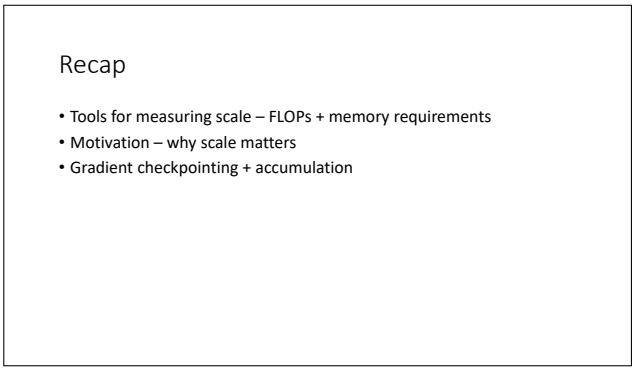
3.2 Gradient Checkpointing



- idea is to initially throw out the activation in the forward pass after we've done it, but in the backpropogation step we redo the entire forward computation step.
- This is computationally expensive, but the memory requirements stay low; it doesn't scale with the number of layers.
- The forward pass is much faster than the backwards pass
- Therefore, we select checkpoints from which we continue the forward pass.



-



-

-

Examinable

Finetuning problem setting

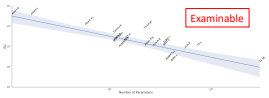
- After pretraining we may want to adapt our model to a specific downstream task.
- The pretrained weights Φ, are updated $\Phi + \Delta\Phi$ according to
- $\max_{\Phi} \sum_{\{x,y\}\sim Z} \sum_{t=1}^{y} \log(P_{\Phi}(y_t|x, y_{<t}))$

Context-target tokens — Autoregressive foundation model

- Here, every parameter in Φ is updated therefore need to store gradients + optimisers of every weight.
- For every downstream task a new model of size Φ is required + we need to relearn ΔΦ parameters.

Examinable

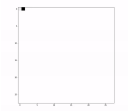
Low Rank Adaption (LoRA)



- Motivation
  - Li et al. demonstrated that the model-loss overparametrised space has a low intrinsic dimension/rank.
  - What if ΔΦ also has low intrinsic rank we can approximate ΔΦ with Θ where |Θ| << |Φ|

$$\max_{\Theta} \sum_{\{x,y\}\sim Z} \sum_{t=1}^{y} \log(P_{\Phi+\Theta}(y_t|x, y_{<t}))$$

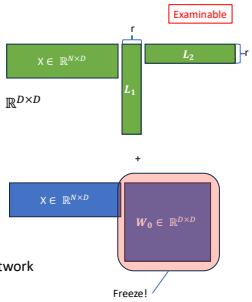
Only optimise Θ



Li et al. Measuring the intrinsic dimension of objective landscapes 2018
Goodfellow et al. Qualitatively Characterizing Neural Network Optimization Problems 2015
Aghajanyan et al. Intrinsic Dimensionality Explains the Effectiveness of Language Model Fine-Tuning 2020

Examinable

Low Rank Adaption (LoRA)



- For a weight matrix in a pretrained model: $\boldsymbol{W_0} \in \mathbb{R}^{D\times D}$
- Following convention that at fine tuning we are learning: $\Delta\boldsymbol{W} \in \mathbb{R}^{D\times D}$
- For a pretrained set up of $\boldsymbol{y} = \boldsymbol{W_0 x}$
- $\boldsymbol{y} = \boldsymbol{XW_0} + \boldsymbol{X\Delta W} \approx \boldsymbol{XW_0} + \boldsymbol{XL_1L_2}$

Where , $\boldsymbol{L_1} \in \mathbb{R}^{D\times r}$ and $\boldsymbol{L_2} \in \mathbb{R}^{r\times D}$ and r << D
(note: matrix multiplication is distributive!)
(note: $\boldsymbol{L_1L_2} \in \mathbb{R}^{D\times D}$)

- This can be done for every weight layer in the network

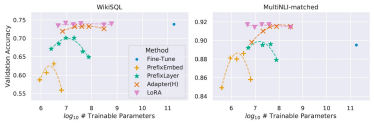
Hu et al. LoRA: Low-Rank Adaption of Large Language 2021

Examinable

Low Rank Adaption (LoRA)

- Empirically show that r = 1 or r = 2 yields good results for GPT-3 which has a full rank of 12,288!
- r will depend on the degree of shift between the pretraining dataset and the finetuning dataset.
- Here, the number of trainable parameters, |Θ| can be 0.01% of the pretrained parameters, |Φ|.

	Weight Type	$r=1$	$r=2$	$r=4$	$r=8$	$r=64$
WikiSQL(±0.5%)	W_q	68.8	69.6	70.5	70.4	70.0
	W_q, W_v	73.4	73.3	73.7	73.8	73.5
	W_q, W_k, W_v, W_o	74.1	73.7	74.0	74.0	73.9
MultiNLI (±0.1%)	W_q	90.7	90.9	91.1	90.7	90.7
	W_q, W_v	91.3	91.4	91.3	91.6	91.4
	W_q, W_k, W_v, W_o	91.2	91.7	91.7	91.5	91.4



Hu et al. LoRA: Low-Rank Adaption of Large Language 2021

Examinable

Low Rank Adaption (LoRA)

- At inference time we combine: $\boldsymbol{W_0 + L_1 L_2}$
  - No additional inference time (no extra steps to compute both $\boldsymbol{XW_0}$ and $\mathbf{X}\boldsymbol{L_1 L_2}$)
  - We can always subtract $\boldsymbol{L_1 L_2}$ when we are done and combine with a different set of parameters $\boldsymbol{L_1^B L_2^B}$
- How much GPU memory does this save?
- No longer need to train 99.99% of parameters (number from Llama 65B in part1):
  - Gradients $4*65*10^9 = 260GB$
  - Optimiser: $2*4*65*10^9 = 520GB$
- Still need to store:
  - Model parameters = 260GB
  - Activations for backward = 172GB
- Assuming batch size of 1 we need 3 times less GPU resources.

Hu et al. LoRA: Low-Rank Adaption of Large Language 2021

Examinable

QLoRA Background: k-bit quantisation

- Teaser: QLoRA enables finetuning of a 65B parameter transformer model on a single 48GB GPU + in 12 hours leads to SOTA opensource model performance.
- K-bit quantisation. Example float32 → Int8

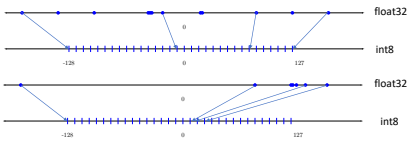
$$\mathbf{X}^{\text{Int8}} = \text{round}\left(\frac{127}{\text{absmax}(\mathbf{X}^{\text{FP32}})}\mathbf{X}^{\text{FP32}}\right) = \text{round}(c^{\text{FP32}} \cdot \mathbf{X}^{\text{FP32}}),$$

$$\text{dequant}(c^{\text{FP32}}, \mathbf{X}^{\text{Int8}}) = \frac{\mathbf{X}^{\text{Int8}}}{c^{\text{FP32}}} = \mathbf{X}^{\text{FP32}}$$

- To ensure full range of input is captured in Int8, input is rescaled to target range here [-127, 127]
- Where c^X is the quantisation constant

Examinable

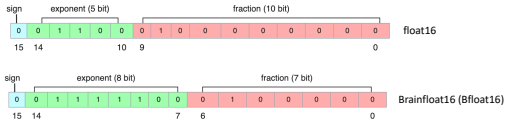
QLoRA: Background: k-bit quantisation



Outliers result suboptimal allocation of quantized bins
Chunking! Slice $\boldsymbol{W} \in \mathbb{R}^{D \times D}$ into n chunks of size C where $n = \frac{D \times D}{C}$
Quantise each block separately with unique c^X

Examinable

QLoRA: BrainFloat



- Deep learning models are more sensitive to underflow + overflow than to precision
- Hardware requirements scale quadradically with fraction but not exponent

[developed at google] https://cloud.google.com/tpu/docs/bfloat16

- QLoRA

Examinable

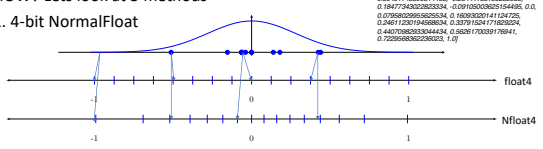
Dwarfs other memory requirements

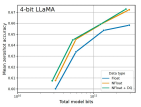
LoRA recap: $y = X\boldsymbol{W_0} + XL_1L_2$

QLoRA looks to quantise $\boldsymbol{W_0}$ from bfloat16 to 4 bit!

HOW? Lets look at 3 methods

1. 4-bit NormalFloat





[-1.0, -0.6961928009986877, -0.5250730514526367, -0.39491748809814453, -0.2844413816928864, -0.18477343022823334, -0.09105003625154495, 0.0, 0.07958029955625534, 0.16093020141124725, 0.24611230194568634, 0.33791524171829224, 0.44070982933044434, 0.5626170039176941, 0.7229568362236023, 1.0]

Dettmers et al. QLoRA: Efficient Finetuning of Quantised LLMs 2023

- QLoRA

Examinable

Dwarfs other memory requirements

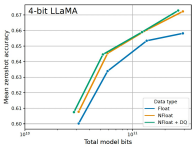
LoRA recap: $y = X\boldsymbol{W_0} + XL_1L_2$

QLoRA looks to quantise $\boldsymbol{W_0}$ from bfloat16 to 4 bit!

HOW? Lets look at 2 methods

2. Double Quantisation

- For each block, each quantisation constant takes up 32 bits of memory.
- For a chunk size of 64 that is an extra 32/64 = 0.5 bits per parameter!
- Quantise $c^{32float} \rightarrow c^{8float}$ and use a chunk size of 256
- 8/64v+32/(64*256) = 0.127 bits



Dettmers et al. QLoRA: Efficient Finetuning of Quantised LLMs 2023

- QLoRA

Examinable

Dwarfs other memory requirements

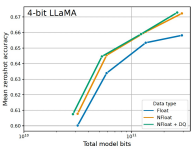
LoRA recap: $y = X\boldsymbol{W_0} + XL_1L_2$

QLoRA looks to quantise $\boldsymbol{W_0}$ from bfloat16 to 4 bit!

HOW? Lets look at 3 methods

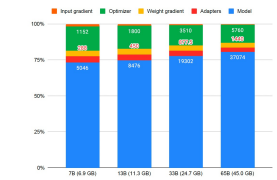
3. When required for computation we dequantise weights from NormalFloat4 to BFLoat16

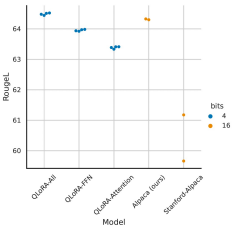
$$y = X\boldsymbol{dequant}(\boldsymbol{W_0}) + XL_1L_2$$



Dettmers et al. QLoRA: Efficient Finetuning of Quantised LLMs 2023

- QLoRA





Implementation: https://huggingface.co/docs/peft/main/en/index

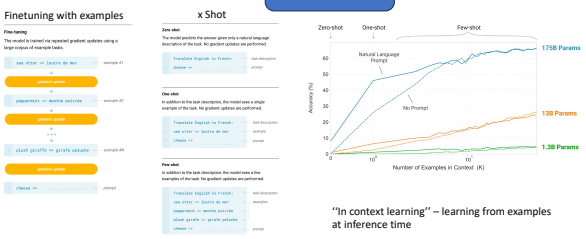
Prompt Engineering – What is it?

- Performance of autoregressive models at inference time varies greatly depending on the input/prompt.
- $P_\Phi(x_t|x_{<t})$ where $x_{<t}$ is the input/context
- Here, P_Φ, selects the response x_t which has the highest probability of following the prompt given the training data.
- Therefore, if we are looking to elicit a certain response, x_t, we need to spend some time thinking about what $x_{<t}$ would maximise the chances of seeing a desirable $x_{<t}$.

Few Shot | Context | Chain of Thought

Prompt Engineering

Few Shot



"In context learning" – learning from examples at inference time

Brown et al. Language Models are Few-Shot Learners 2020

Prompt Engineering – What is it?

- Some call it 'in context learning' – learning from examples
- However, this wording is confusing as no weights are updated.
- Another interpretation:
  - Not learning *how* to perform the task.
  - Rather, instructing where in the model's learnt data likelihood space we want to sample from.

Reynolds et al. Prompt Programming for Large Language Models: Beyond the Few-Shot Paradigm 2021

The Dynamics of Language

- What function are these autoregressive models attempting to fit?
  - NOT just grammar + ngram probabilities
  - Language has been designed to fit the world we live in.
  - A good fit requires modelling how the real world affects language.
  - Modelling language is as difficult as modelling everything that influences it!

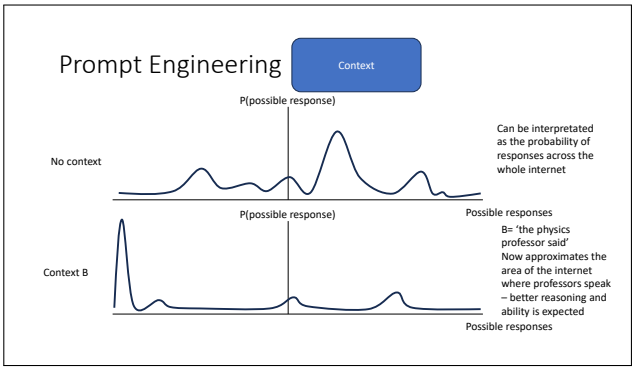
dropped the ball it {fell, floated, ...}

The scientist was surprised to see that when she dropped the ball it {fell, floated, ...}

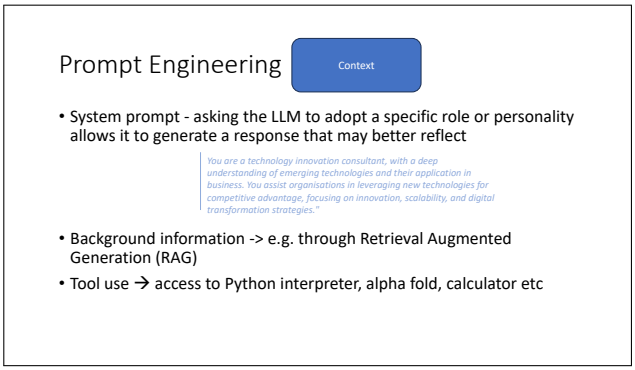
The ball was filled with helium gas. When I dropped the ball it {fell, floated, ...}

When the boy dropped the ball it {fell, floated, ...}

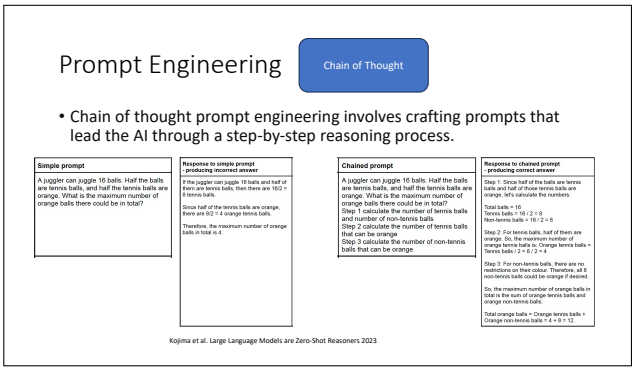
Reynolds et al. Prompt Programming for Large Language Models: Beyond the Few-Shot Paradigm 2021


Prompt Engineering
Context
P(possible response)
No context
Can be interpretated as the probability of responses across the whole internet
Possible responses
P(possible response)
Context B
B= 'the physics professor said'
Now approximates the area of the internet where professors speak – better reasoning and ability is expected
Possible responses

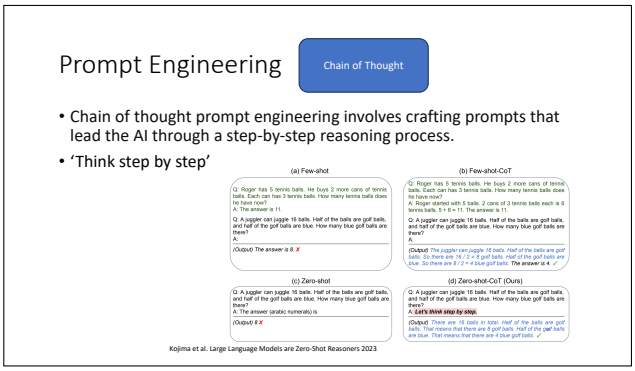
-


Prompt Engineering
Context
• System prompt - asking the LLM to adopt a specific role or personality allows it to generate a response that may better reflect
"You are a technology innovation consultant, with a deep understanding of emerging technologies and their application in business. You assist organisations in leveraging new technologies for competitive advantage, focusing on innovation, scalability, and digital transformation strategies."
• Background information -> e.g. through Retrieval Augmented Generation (RAG)
• Tool use → access to Python interpreter, alpha fold, calculator etc

-


Prompt Engineering
Chain of Thought
• Chain of thought prompt engineering involves crafting prompts that lead the AI through a step-by-step reasoning process.
Simple prompt
A juggler can juggle 16 balls. Half the balls are tennis balls, and half the tennis balls are orange. What is the maximum number of orange balls there could be in total?
Response to simple prompt - producing incorrect answer
If the juggler can juggle 16 balls and half of them are tennis balls, then there are 16/2 = 8 tennis balls.
Since half of the tennis balls are orange, there are 8/2 = 4 orange tennis balls.
Therefore, the maximum number of orange balls in total is 4.
Chained prompt
A juggler can juggle 16 balls. Half the balls are tennis balls, and half the tennis balls are orange. What is the maximum number of orange balls there could be in total?
Step 1 calculate the number of tennis balls and number of non-tennis balls
Step 2 calculate the number of tennis balls that can be orange
Step 3 calculate the number of non-tennis balls that can be orange
Response to chained prompt - producing correct answer
Step 1: Since half of the balls are tennis balls and half of those tennis balls are orange, let's calculate the numbers.
Total balls = 16
Tennis balls = 16 / 2 = 8
Non-tennis balls = 16 / 2 = 8
Step 2: For tennis balls, half of them are orange. So, the maximum number of orange tennis balls is: Orange tennis balls = Tennis balls / 2 = 8 / 2 = 4
Step 3: For non-tennis balls, there are no restrictions on their colour. Therefore, all 8 non-tennis balls could be orange if desired.
So, the maximum number of orange balls in total is the sum of orange tennis balls and orange non-tennis balls:
Total orange balls = Orange tennis balls + Orange non-tennis balls = 4 + 8 = 12.
Kojima et al. Large Language Models are Zero-Shot Reasoners 2023

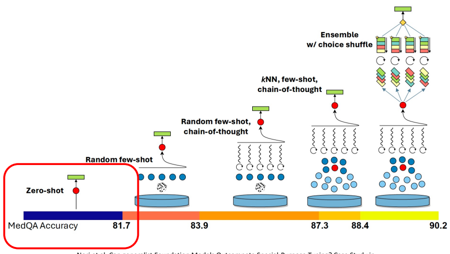
-


Prompt Engineering
Chain of Thought
• Chain of thought prompt engineering involves crafting prompts that lead the AI through a step-by-step reasoning process.
• 'Think step by step'
(a) Few-shot
Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?
A: The answer is 11.
Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?
A:
(Output) The answer is 8 ✗
(b) Few-shot-CoT
Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?
A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. 5 + 6 = 11. The answer is 11.
Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?
A:
(Output) The juggler can juggle 16 balls. Half of the balls are golf balls. So there are 16 / 2 = 8 golf balls. Half of the golf balls are blue. So there are 8 / 2 = 4 blue golf balls. The answer is 4. ✓
(c) Zero-shot
Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?
A: The answer (arabic numerals) is
(Output) 8 ✗
(d) Zero-shot-CoT (Ours)
Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?
A: Let's think step by step.
(Output) There are 16 balls in total. Half of the balls are golf balls. That means that there are 8 golf balls. Half of the golf balls are blue. That means that there are 4 blue golf balls. ✓
Kojima et al. Large Language Models are Zero-Shot Reasoners 2023

-

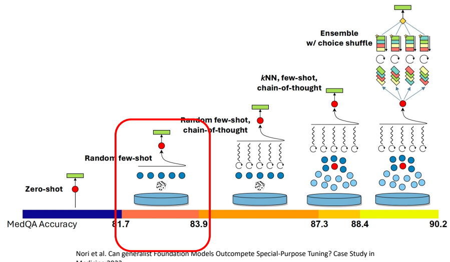
Nice practical guide + tips and tricks:
https://platform.openai.com/docs/guides/prompt-engineering

Medprompt case study



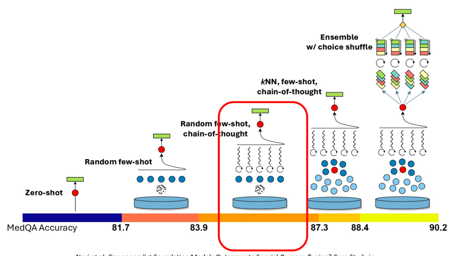
Nori et al. Can generalist Foundation Models Outcompete Special-Purpose Tuning? Case Study in Medicine 2023

Medprompt case study

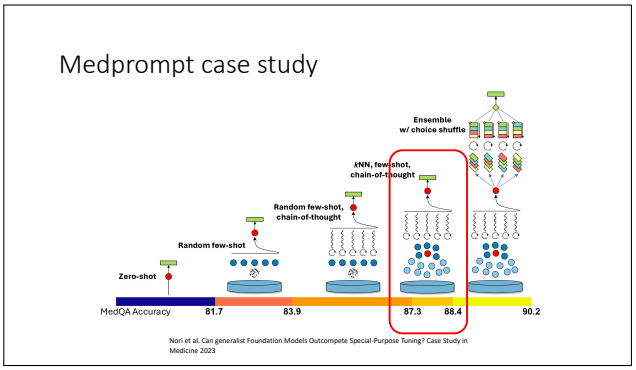


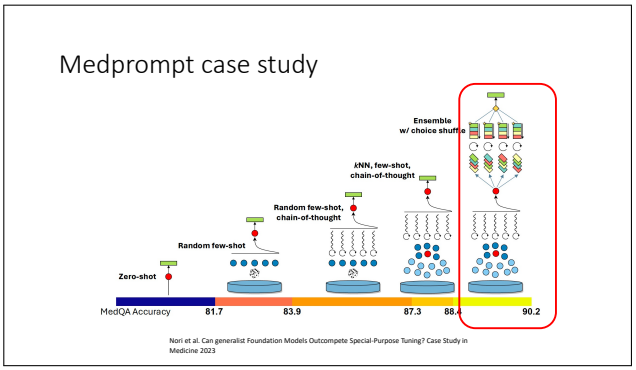
Nori et al. Can generalist Foundation Models Outcompete Special-Purpose Tuning? Case Study in Medicine 2023

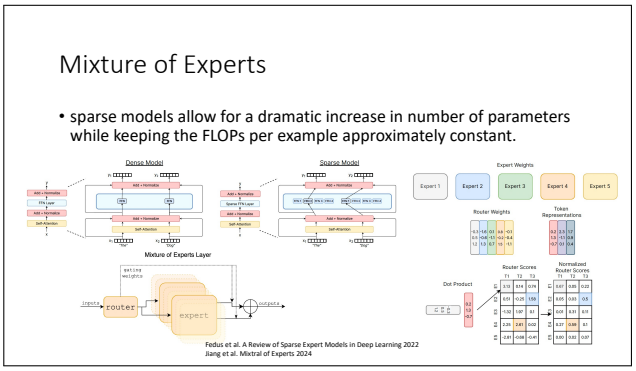
Medprompt case study



Nori et al. Can generalist Foundation Models Outcompete Special-Purpose Tuning? Case Study in Medicine 2023


Medprompt case study
Ensemble w/ choice shuffle
kNN, few-shot, chain-of-thought
Random few-shot, chain-of-thought
Random few-shot
Zero-shot
MedQA Accuracy
81.7
83.9
87.3
88.4
90.2
Nori et al. Can generalist Foundation Models Outcompete Special-Purpose Tuning? Case Study in Medicine 2023


Medprompt case study
Ensemble w/ choice shuffle
kNN, few-shot, chain-of-thought
Random few-shot, chain-of-thought
Random few-shot
Zero-shot
MedQA Accuracy
81.7
83.9
87.3
88.4
90.2
Nori et al. Can generalist Foundation Models Outcompete Special-Purpose Tuning? Case Study in Medicine 2023


Mixture of Experts
• sparse models allow for a dramatic increase in number of parameters while keeping the FLOPs per example approximately constant.
Dense Model
Sparse Model
Mixture of Experts Layer
router
expert
Expert Weights
Expert 1
Expert 2
Expert 3
Expert 4
Expert 5
Router Weights
Token Representations
Router Scores
Dot Product
Normalized Router Scores
Fedus et al. A Review of Sparse Expert Models in Deep Learning 2022
Jiang et al. Mixtral of Experts 2024