

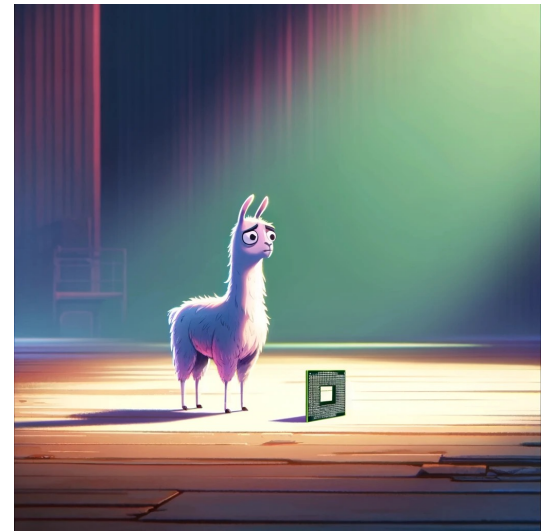
Deep Learning at Scale for the GPU poor - Part 2

Harry Coppock

Only slides with this label

Examinable

is examinable.



Recap

- Tools for measuring scale – FLOPs + memory requirements
- Motivation – why scale matters
- Gradient checkpointing + accumulation

Plan

- Efficient Finetuning
- Prompt Engineering
- Mixture of experts

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

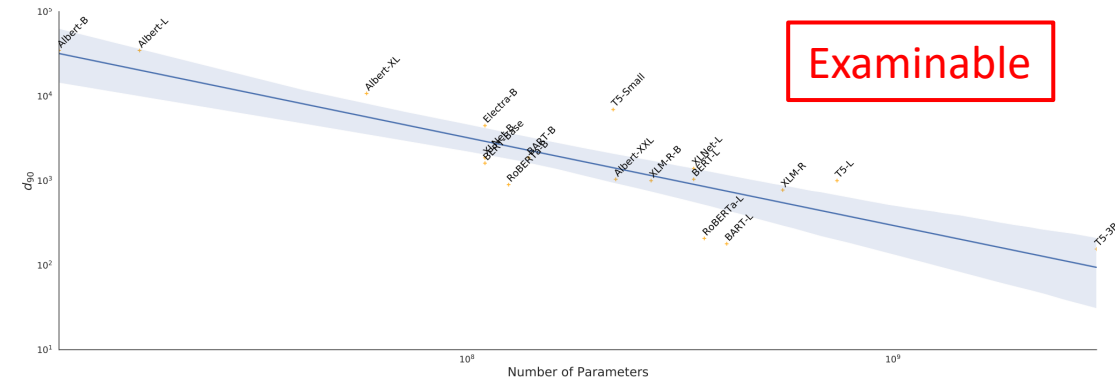
$$\max_{\Phi} \sum_{\{x,y\} \sim \mathcal{Z}} \sum_{t=1}^y \log(P_{\Phi}(y_t | x, y_{<t}))$$

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 $\Delta\Phi$ parameters.

Low Rank Adaption (LoRA)



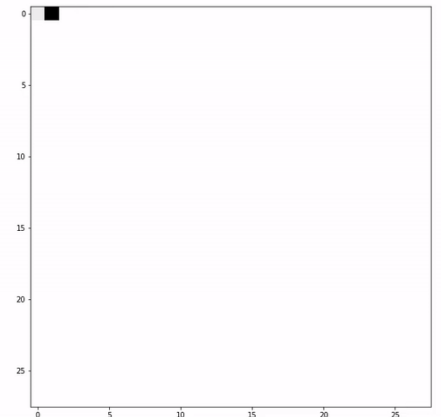
• Motivation

- Li et al. demonstrated that the model-loss overparametrised space has a low intrinsic dimension/rank.
- What if $\Delta\Phi$ also has low intrinsic rank we can approximate $\Delta\Phi$ with Θ where $|\Theta| \ll |\Phi|$

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



Only optimise Θ



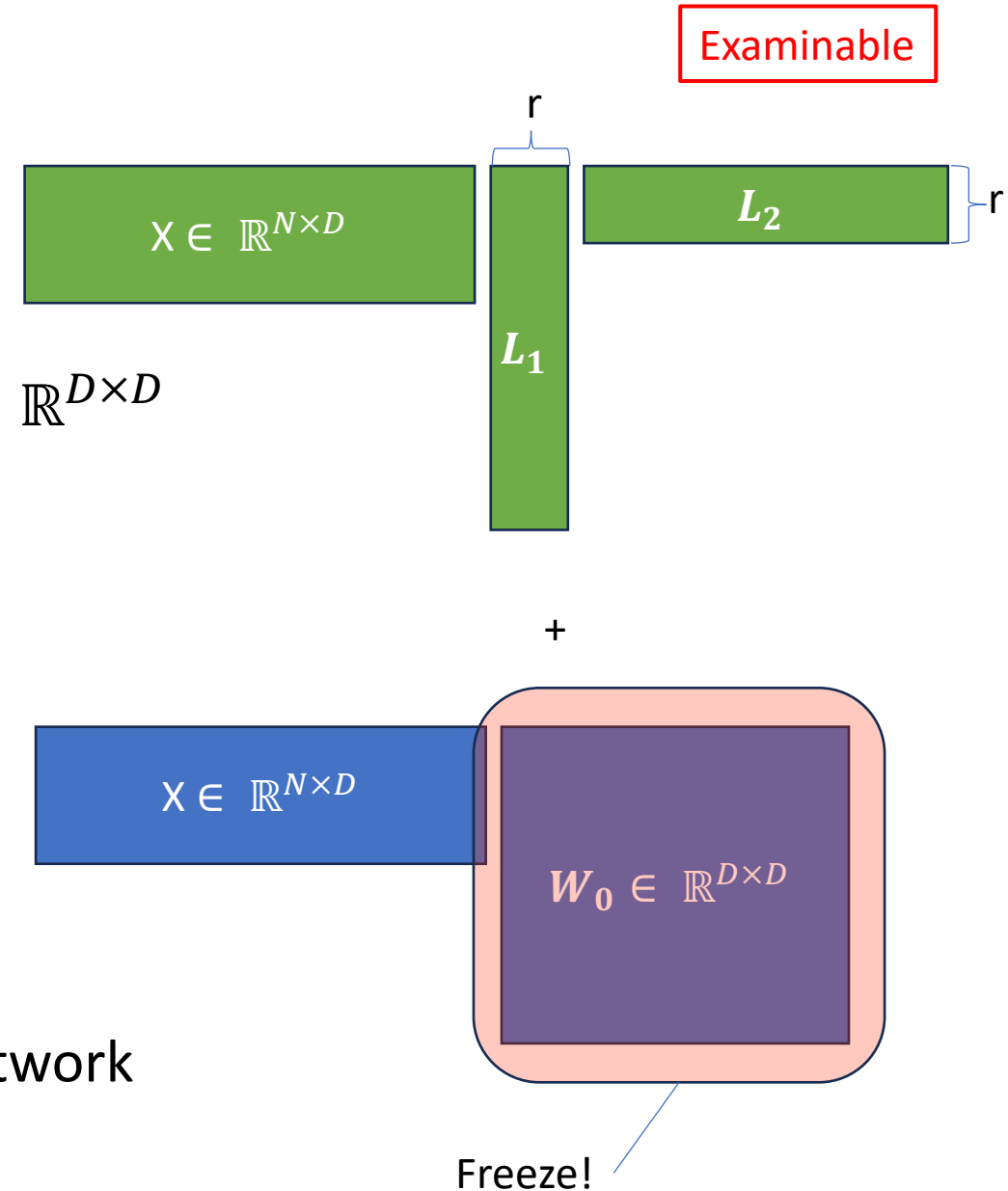
Li et al. Measuring the intrinsic dimension of objective landscapes 2018

Goodfellow et al. Qualitatively Characterizing Neural Network Optimization Problems 2015

Aghajanyan et al. Intrinsic Dimensionality Explains the Effectiveness of Language Model Fine-Tuning 2020

Low Rank Adaption (LoRA)

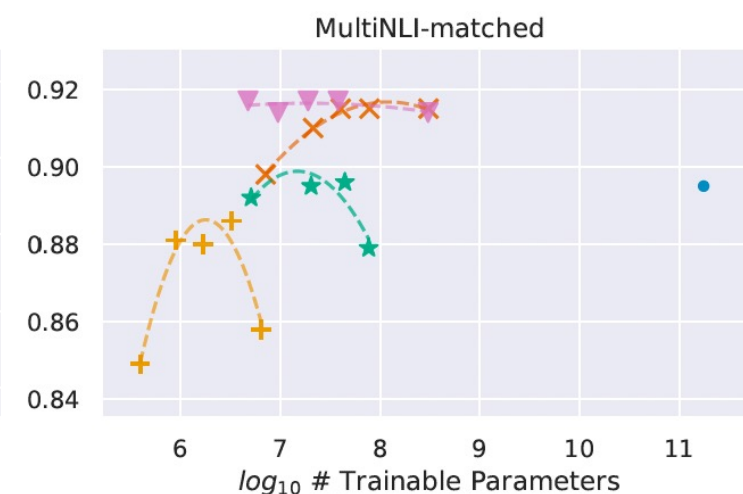
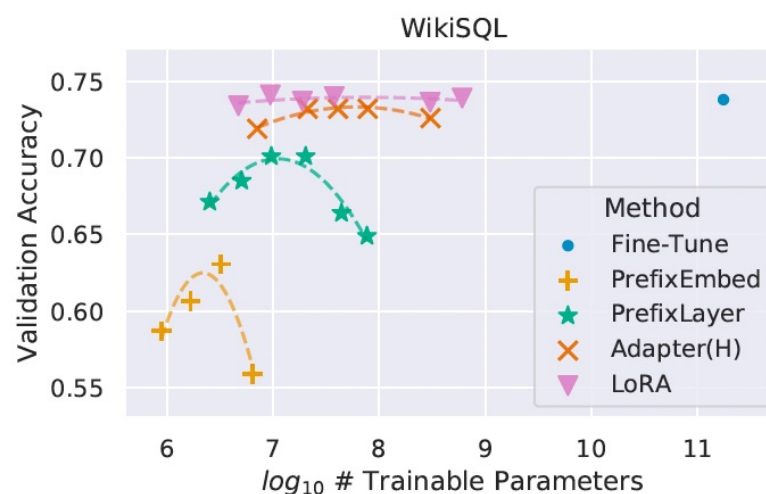
- For a weight matrix in a pretrained model: $W_0 \in \mathbb{R}^{D \times D}$
 - Following convention that at fine tuning we are learning: $\Delta W \in \mathbb{R}^{D \times D}$
 - For a pretrained set up of $y = W_0 x$
 - $y = XW_0 + X\Delta W \approx XW_0 + XL_1L_2$
- Where , $L_1 \in \mathbb{R}^{D \times r}$ and $L_2 \in \mathbb{R}^{r \times D}$ and $r \ll D$
(note: matrix multiplication is distributive!)
(note: $L_1L_2 \in \mathbb{R}^{D \times D}$)
- This can be done for every weight layer in the network



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, $|\Theta|$ can be 0.01% of the pretrained parameters, $|\Phi|$.

	Weight Type	$r = 1$	$r = 2$	$r = 4$	$r = 8$	$r = 64$
WikiSQL($\pm 0.5\%$)	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 ($\pm 0.1\%$)	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



Low Rank Adaption (LoRA)

- At inference time we combine: $W_0 + L_1 L_2$
 - No additional inference time (no extra steps to compute both XW_0 and $XL_1 L_2$)
 - We can always subtract $L_1 L_2$ when we are done and combine with a different set of parameters $L_1^b L_2^b$
- How much GPU memory does this save?
- No longer need to train 99.99% of parameters (number from Llama 65B in part1):
 - Gradients $4 * 65 * 10^9 = 260\text{GB}$
 - Optimiser: $2 * 4 * 65 * 10^9 = 520\text{GB}$
- Still need to store:
 - Model parameters = 260GB
 - Activations for backward = 172GB
- Assuming batch size of 1 we need 3 times less GPU resources.

QLoRA Background: k-bit quantisation

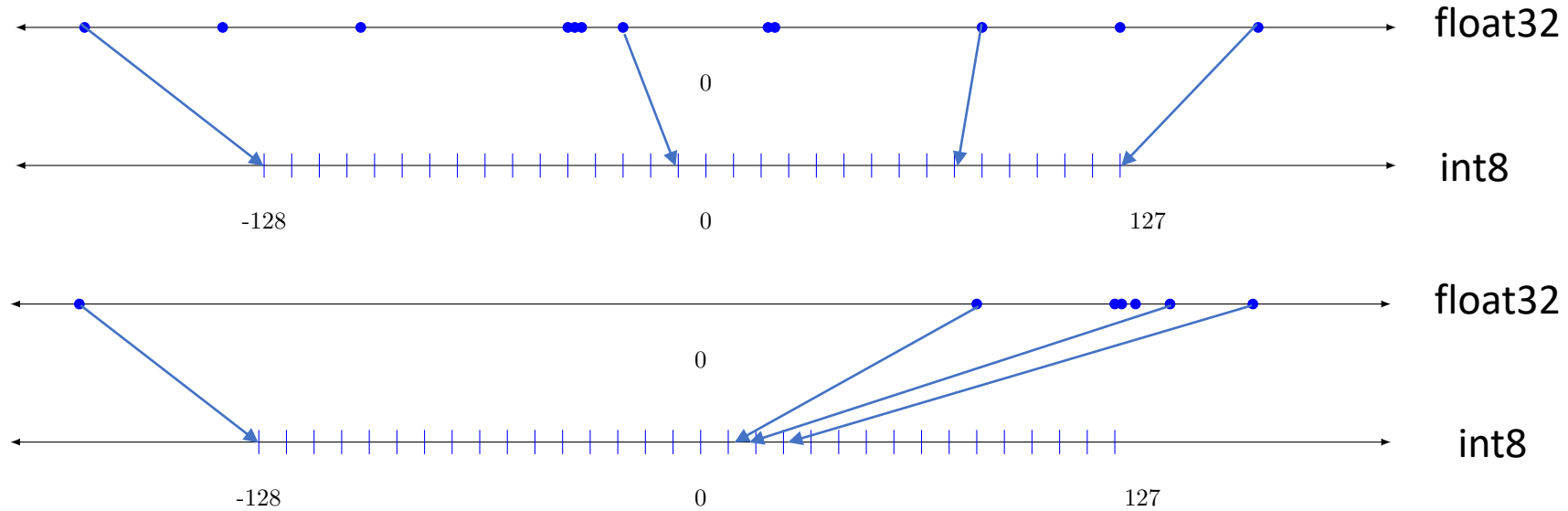
- Teaser: QLoRA enables finetuning of a 65B parameter transformer model on a single 48GB GPU + in 12 hours leads to SOTA opensource model performance.
- K-bit quantisation. Example float32 \rightarrow Int8

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

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

- 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

QLoRA: Background: k-bit quantisation

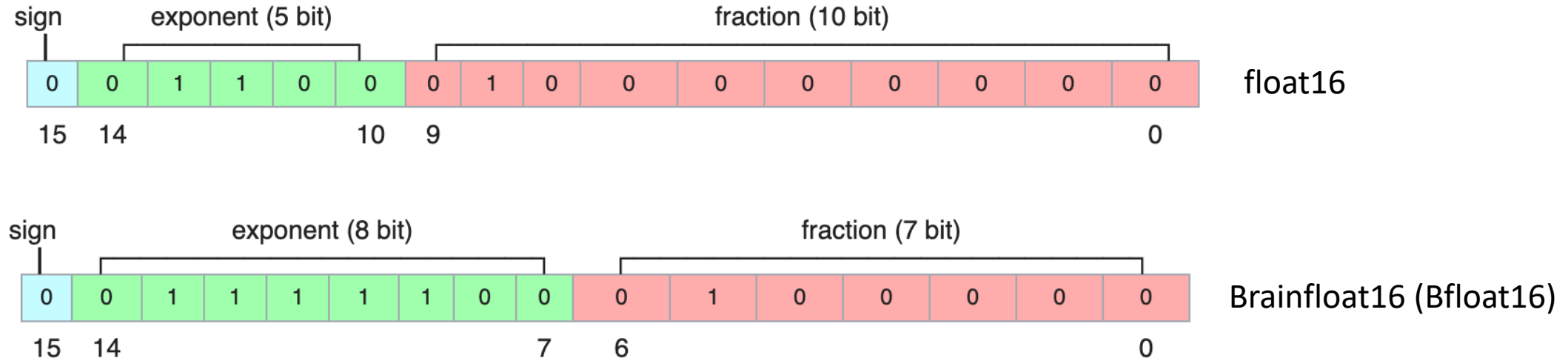


Outliers result suboptimal allocation of quantized bins

Chunking! Slice $\mathbf{W} \in \mathbb{R}^{D \times D}$ into n chunks of size C where $n = \frac{D \times D}{C}$

Quantise each block separately with unique c^x

QLoRA: BrainFloat



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

QLoRA

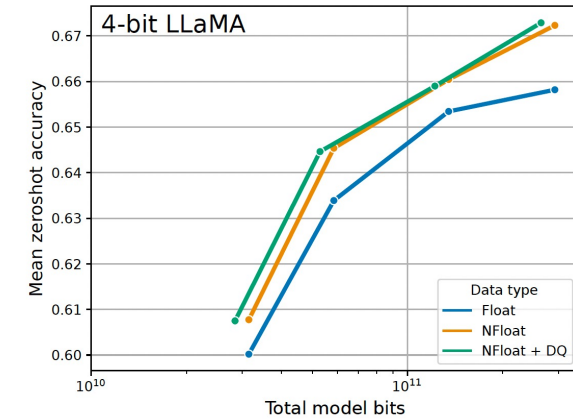
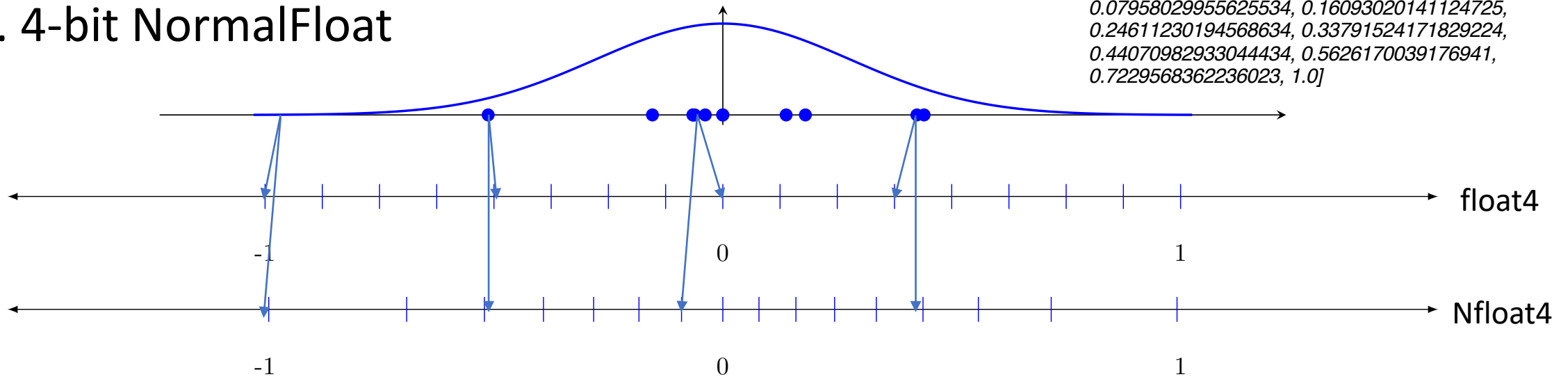
Dwarfs other memory requirements

LoRA recap: $\mathbf{y} = \mathbf{X}\mathbf{W}_0 + \mathbf{X}\mathbf{L}_1\mathbf{L}_2$

QLoRA looks to quantise \mathbf{W}_0 from bfloat16 to 4 bit!

HOW? Lets look at 3 methods

1. 4-bit NormalFloat



QLoRA

Dwarfs other memory requirements

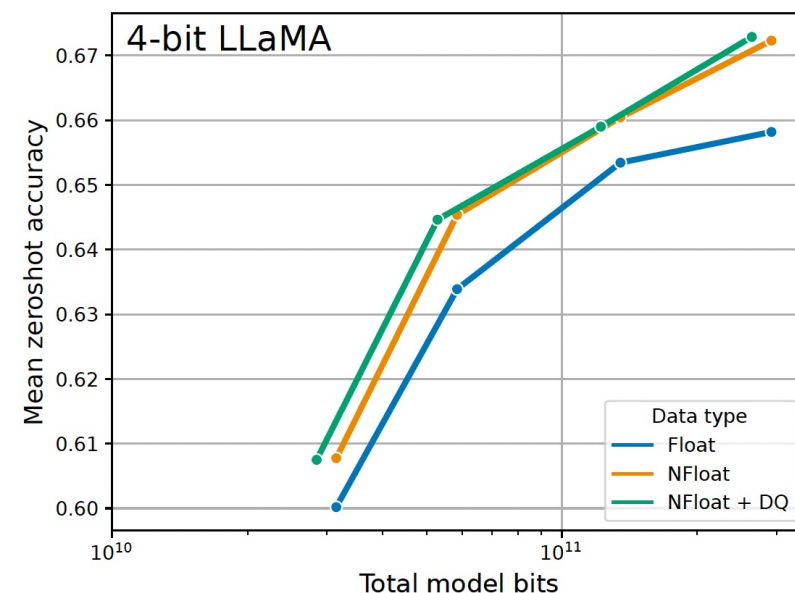
LoRA recap: $y = XW_0 + XL_1L_2$

QLoRA looks to quantise 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



QLoRA

Dwarfs other memory requirements

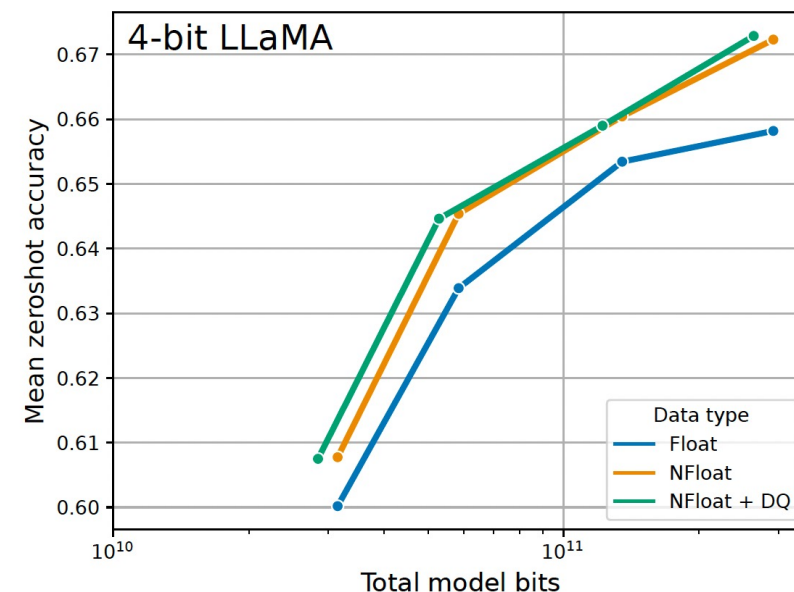
LoRA recap: $y = XW_0 + XL_1L_2$

QLoRA looks to quantise W_0 from bfloat16 to 4 bit!

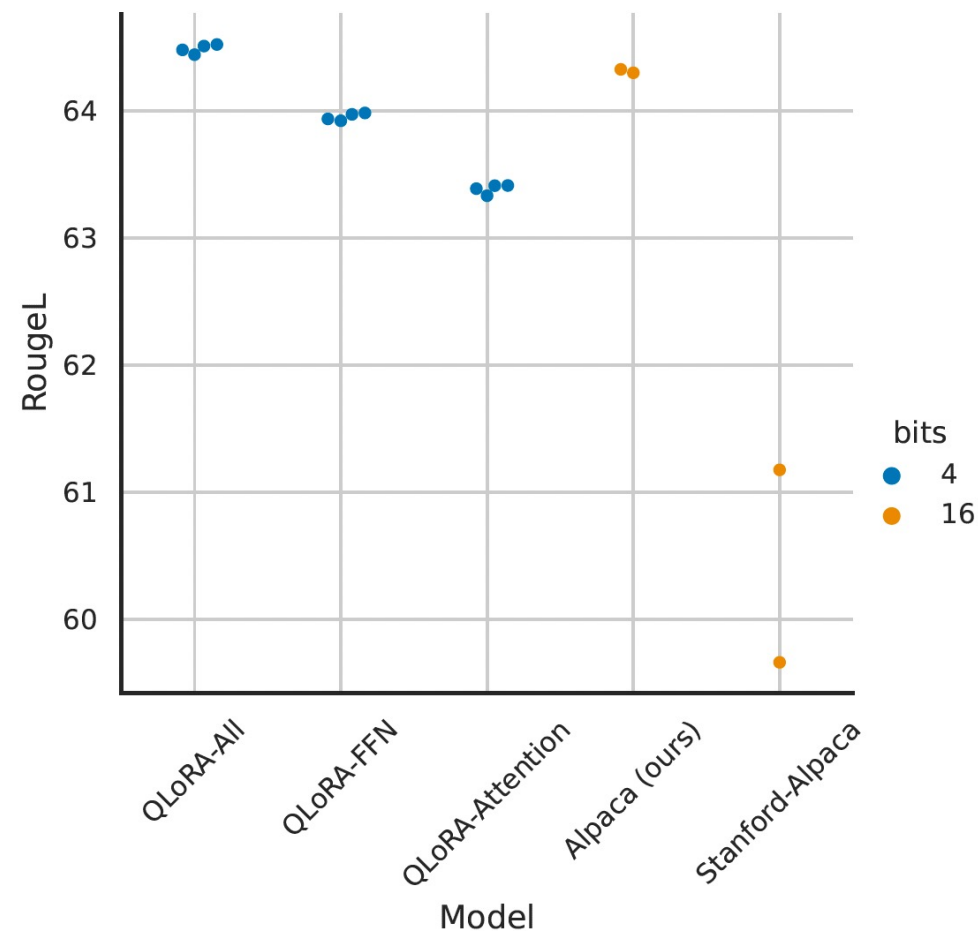
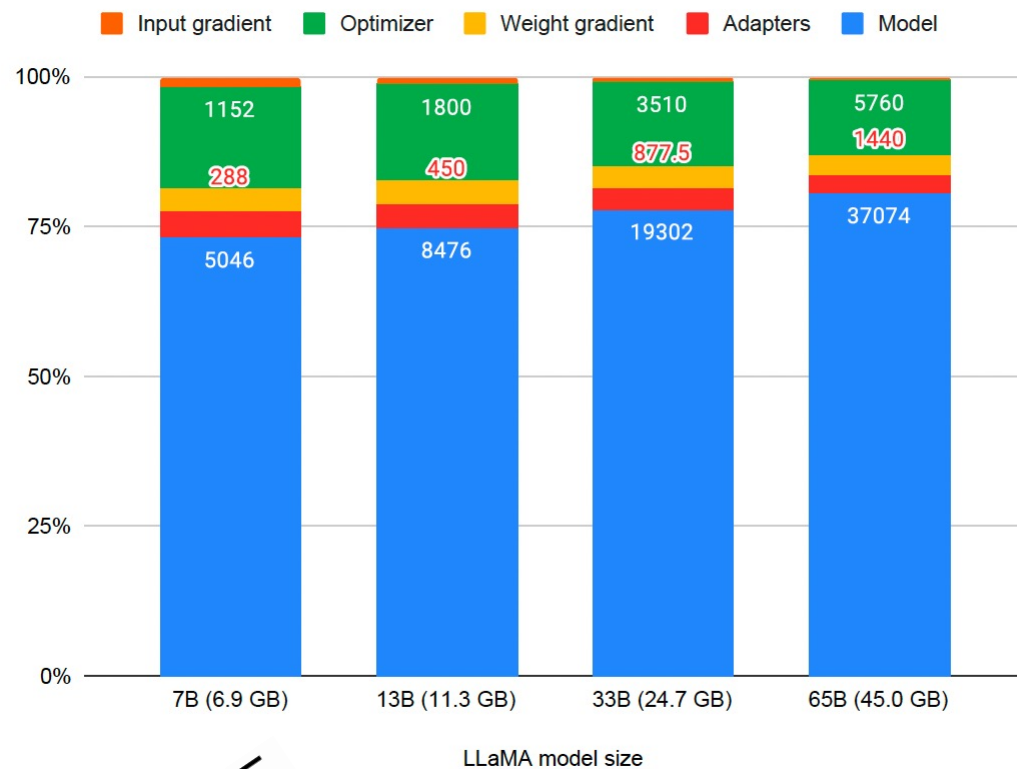
HOW? Lets look at 3 methods

3. When required for computation we dequantise weights from NormalFloat4 to BFloat16

$$y = Xdequant(W_0) + XL_1L_2$$



QLoRA



The Era of 1-bit LLMs:
All Large Language Models are in 1.58 Bits

Shundian Ma, Hongyi Wang, Li Dong, Jingbo Ma, Li Wang, Weishi Wang, Shaoan Huang, Ruofeng Wang, Zhong Xue, Furu Wei

<https://arxiv.org/abs/2308.13025>

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

Finetuning with examples

Fine-tuning

The model is trained via repeated gradient updates using a large corpus of example tasks.



x Shot

Zero-shot

The model predicts the answer given only a natural language description of the task. No gradient updates are performed.

```
1 Translate English to French: ← task description
2 cheese =>                    ← prompt
```

One-shot

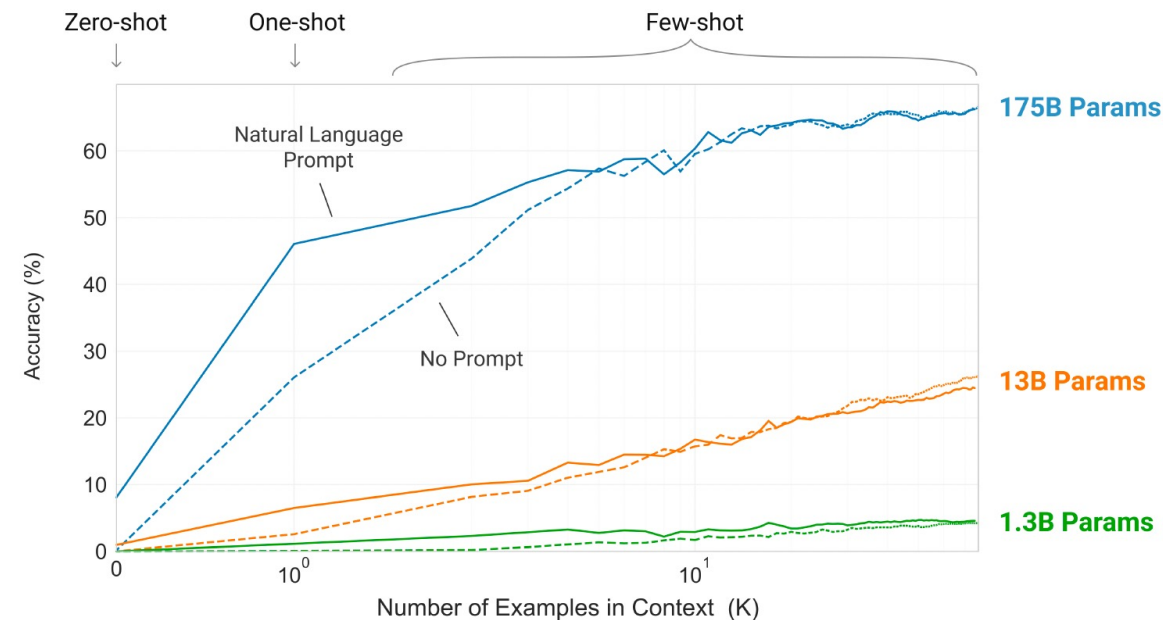
In addition to the task description, the model sees a single example of the task. No gradient updates are performed.

```
1 Translate English to French: ← task description
2 sea otter => loutre de mer   ← example
3 cheese =>                   ← prompt
```

Few-shot

In addition to the task description, the model sees a few examples of the task. No gradient updates are performed.

```
1 Translate English to French: ← task description
2 sea otter => loutre de mer   ← examples
3 peppermint => menthe poivrée ←
4 plush girafe => girafe peluche ←
5 cheese =>                   ← prompt
```



“In context learning” – learning from examples at inference time

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.

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, ...}

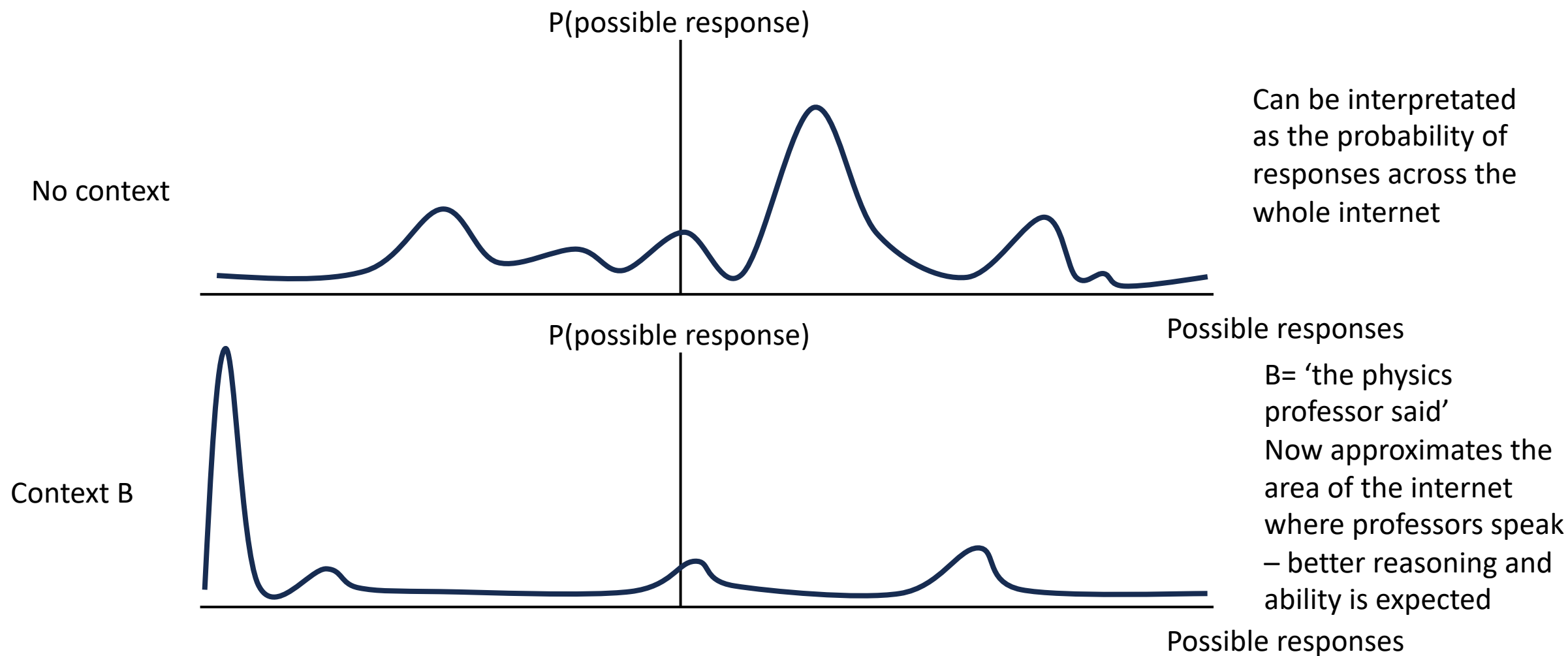
The scientist was surprised to see that when she dropped the ball it {fell, floated, ...}

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

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

Prompt Engineering

Context



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	Response to simple prompt - producing incorrect answer	Chained prompt	Response to chained prompt - producing correct answer
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?	<p>If the juggler can juggle 16 balls and half of them are tennis balls, then there are $16/2 = 8$ tennis balls.</p> <p>Since half of the tennis balls are orange, there are $8/2 = 4$ orange tennis balls.</p> <p>Therefore, the maximum number of orange balls in total is 4.</p>	<p>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?</p> <p>Step 1 calculate the number of tennis balls and number of non-tennis balls</p> <p>Step 2 calculate the number of tennis balls that can be orange</p> <p>Step 3 calculate the number of non-tennis balls that can be orange</p>	<p>Step 1: Since half of the balls are tennis balls and half of those tennis balls are orange, let's calculate the numbers:</p> <p>Total balls = 16 Tennis balls = $16 / 2 = 8$ Non-tennis balls = $16 / 2 = 8$</p> <p>Step 2: For tennis balls, half of them are orange. So, the maximum number of orange tennis balls is: Orange tennis balls = Tennis balls $/ 2 = 8 / 2 = 4$</p> <p>Step 3: For non-tennis balls, there are no restrictions on their colour. Therefore, all 8 non-tennis balls could be orange if desired.</p> <p>So, the maximum number of orange balls in total is the sum of orange tennis balls and orange non-tennis balls:</p> <p>Total orange balls = Orange tennis balls + Orange non-tennis balls = $4 + 8 = 12$.</p>

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?

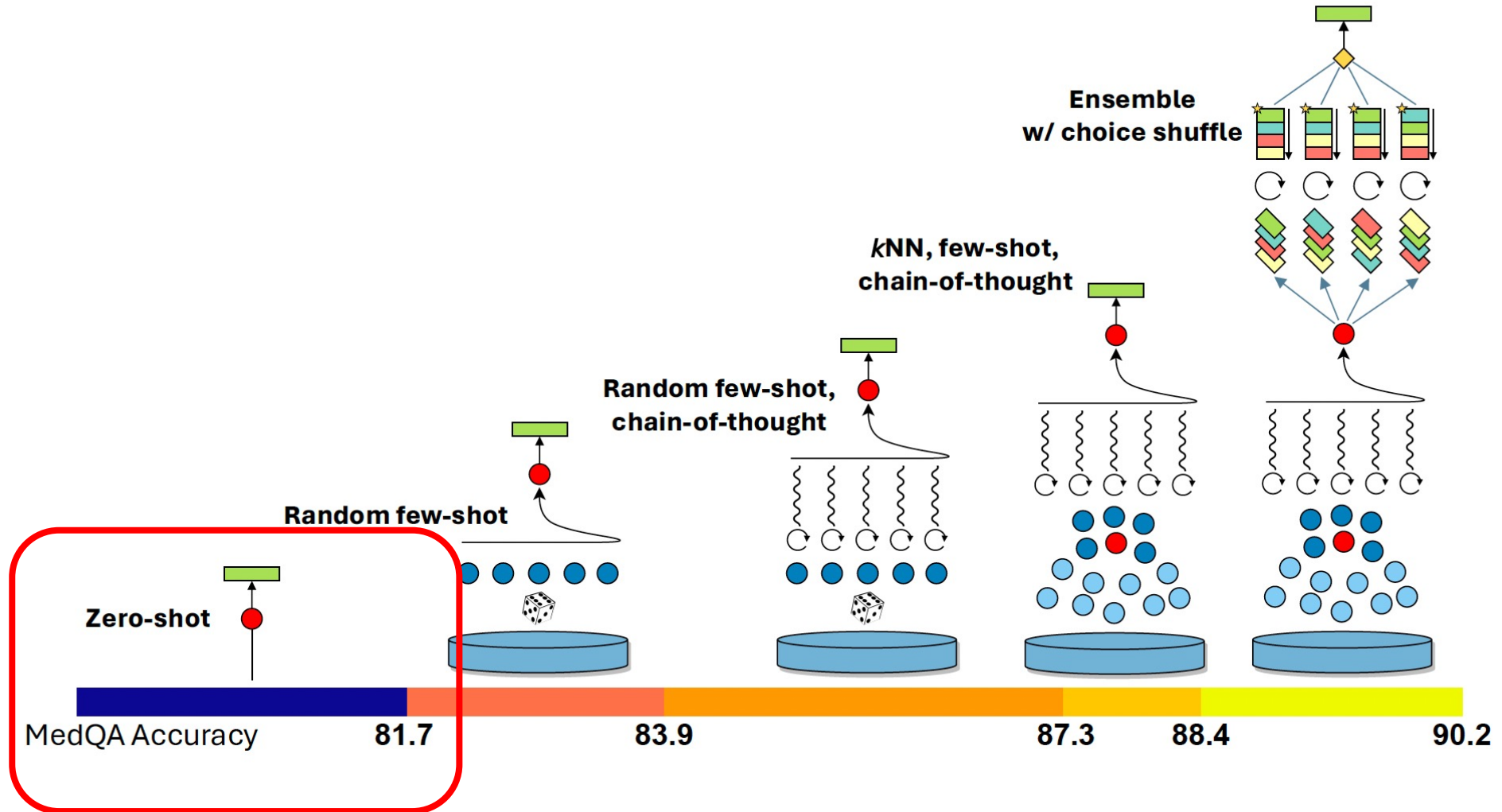
A: **Let's think step by step.**

(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. ✓

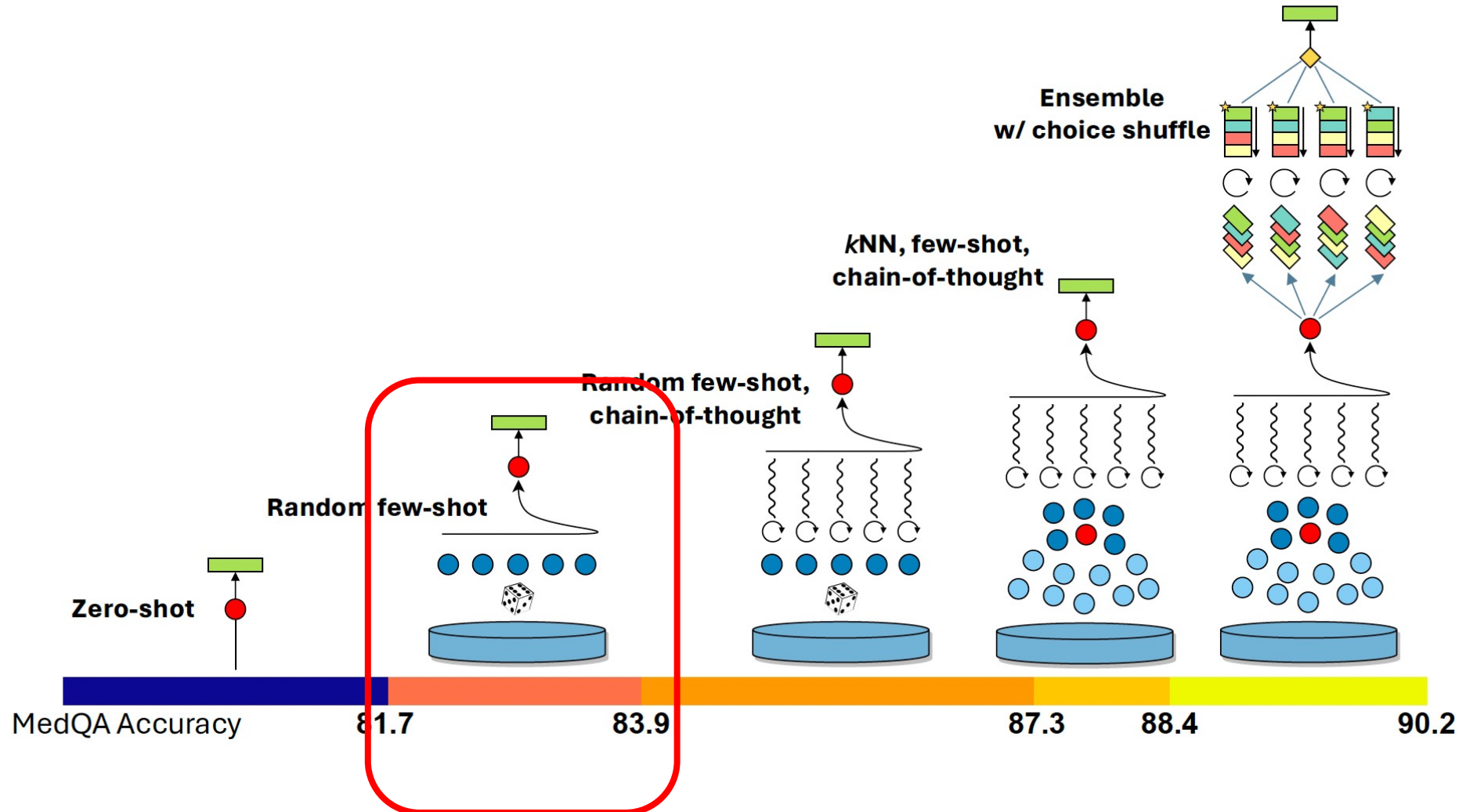
Nice practical guide + tips and tricks:

<https://platform.openai.com/docs/guides/prompt-engineering>

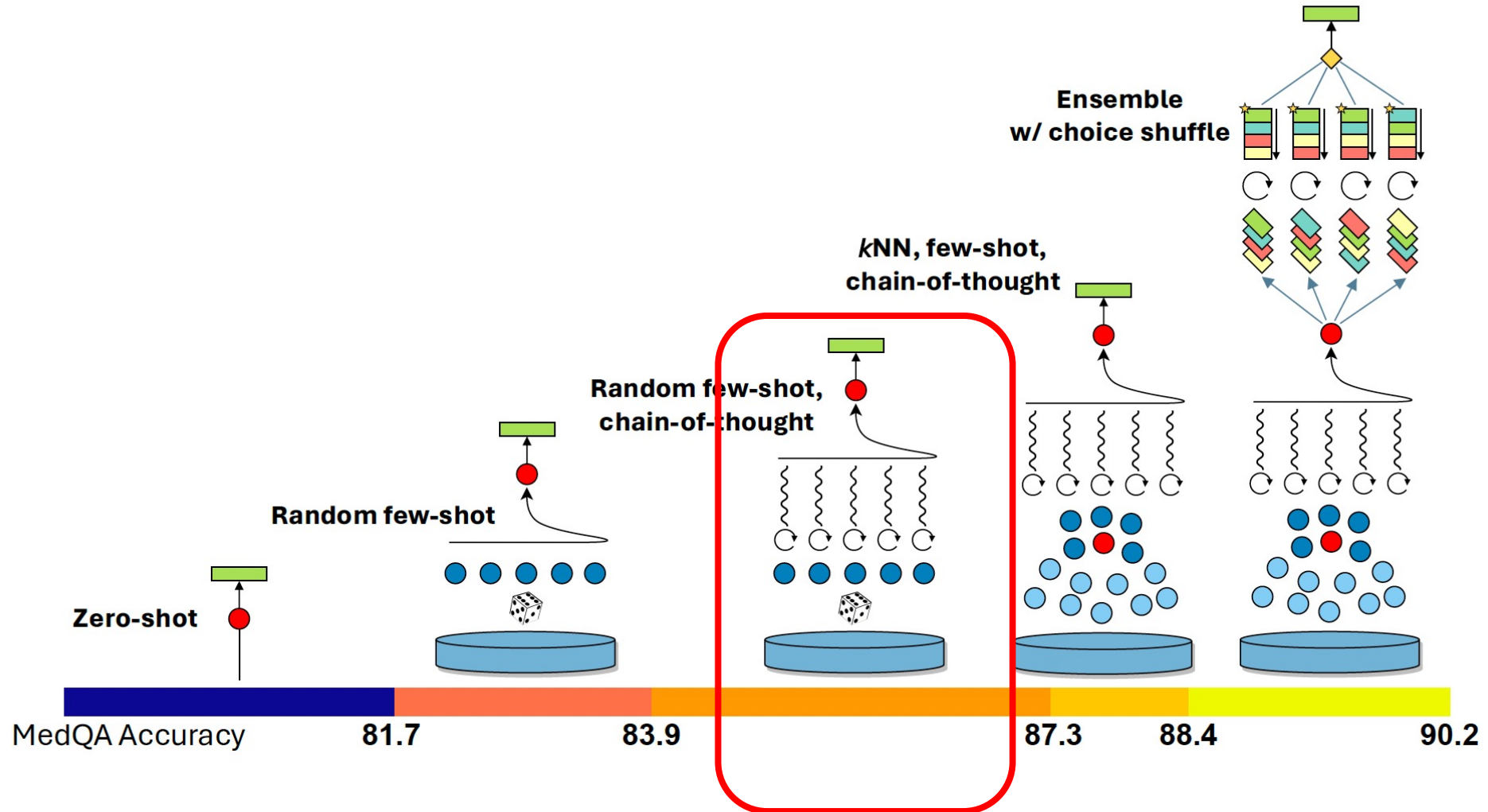
Medprompt case study



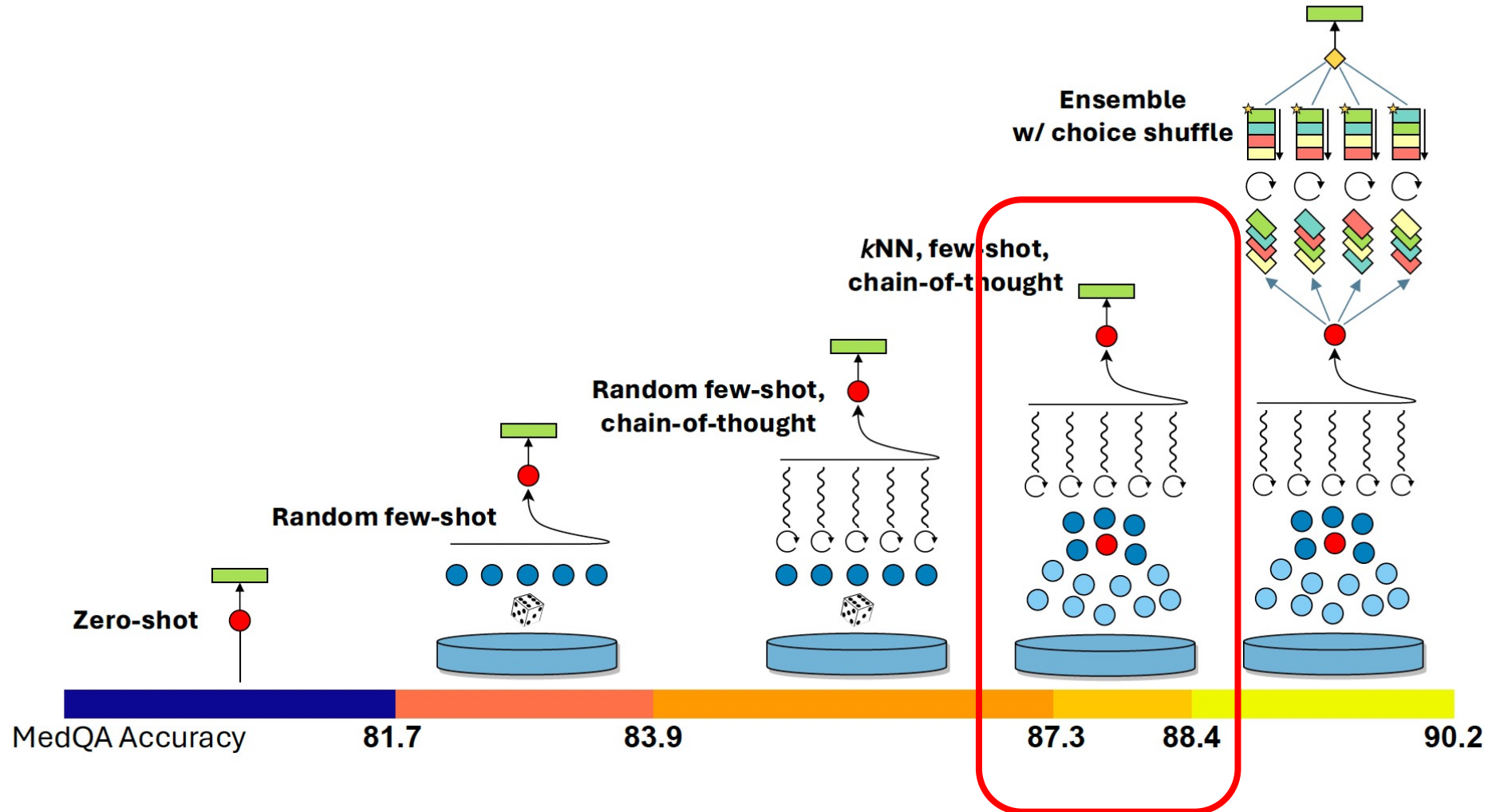
Medprompt case study



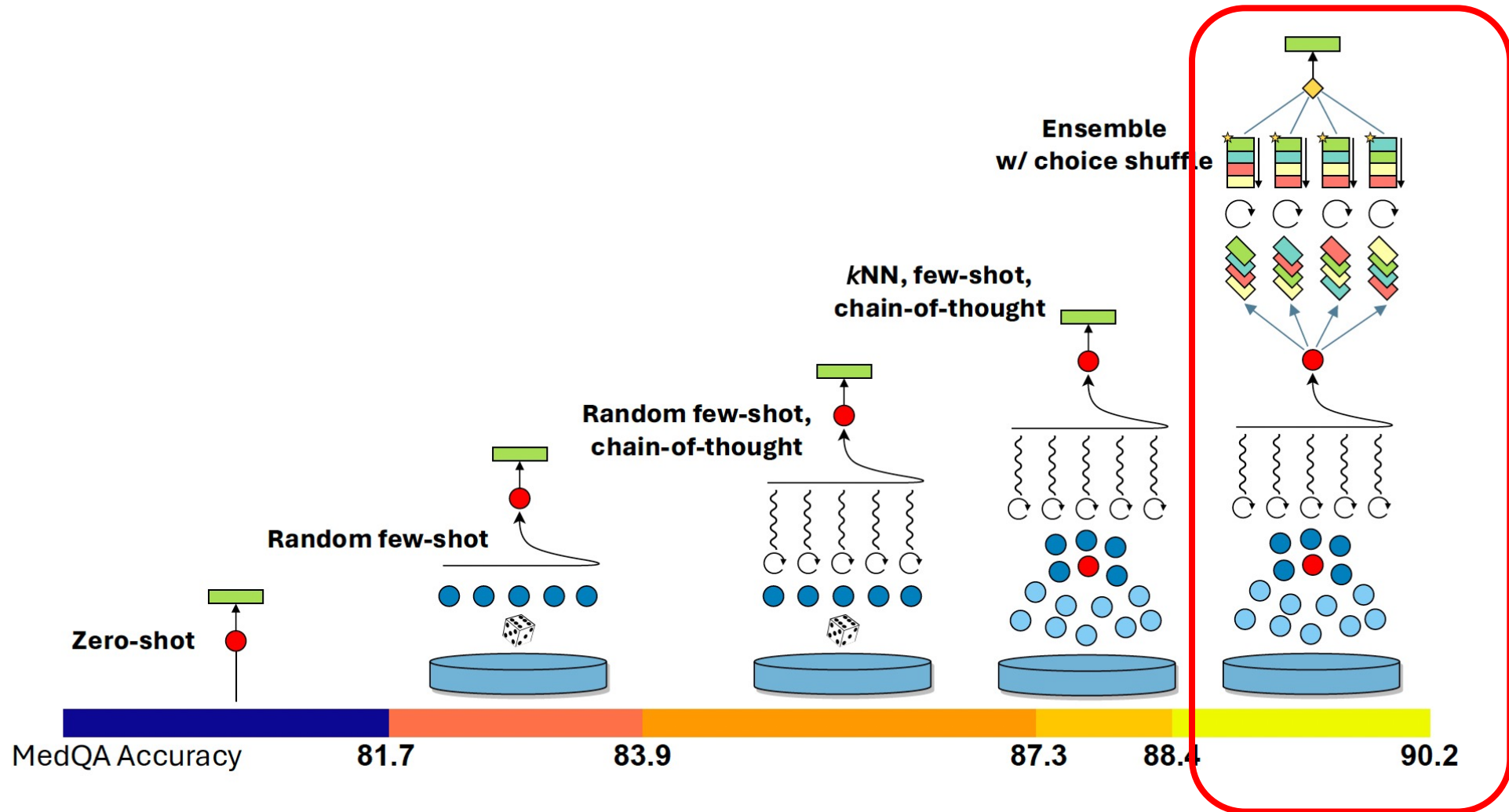
Medprompt case study



Medprompt case study

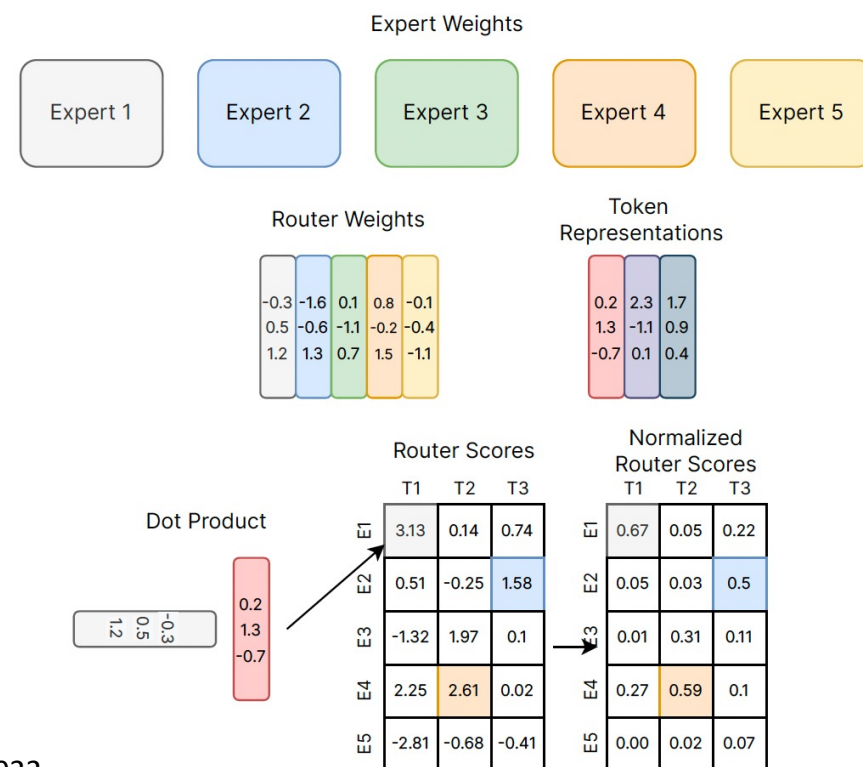
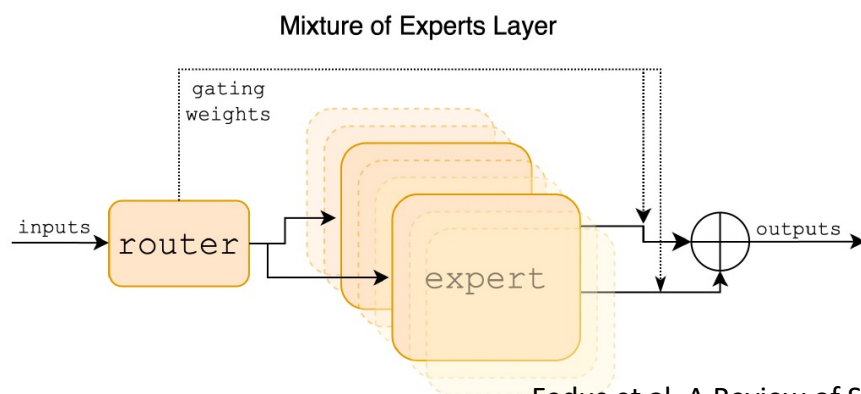
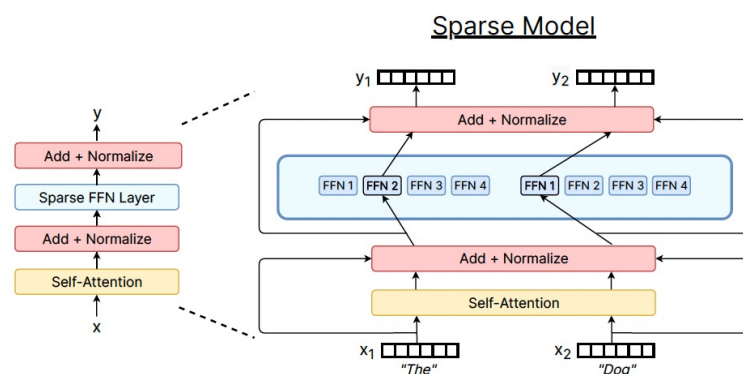
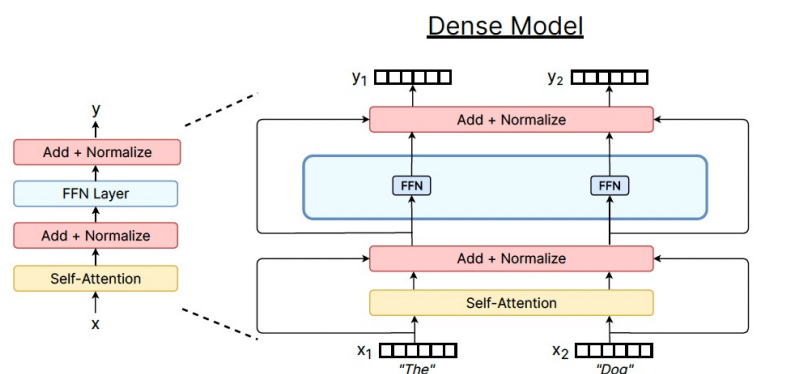


Medprompt case study



Mixture of Experts

- sparse models allow for a dramatic increase in number of parameters while keeping the FLOPs per example approximately constant.



Fedus et al. A Review of Sparse Expert Models in Deep Learning 2022

Jiang et al. Mixtral of Experts 2024

FLASHATTENTION: Fast and Memory-Efficient Exact Attention with IO-Awareness

Tri Dao[†], Daniel Y. Fu[†], Stefano Ermon[†], Atri Rudra[‡], and Christopher Ré[†]

[†]Department of Computer Science, Stanford University

[‡]Department of Computer Science and Engineering, University at Buffalo, SUNY

{trid,danfu}@cs.stanford.edu, ermon@stanford.edu, atri@buffalo.edu,
chrismre@cs.stanford.edu

June 24, 2022

