Week 4: Model Pre-training and Supervised Fine-tuning

Generative AI
Saarland University – Winter Semester 2024/25

Goran Radanovic
genai-w24-tutors@mpi-sws.org

MAX PLANCK INSTITUTE
FOR SOFTWARE SYSTEMS

MAX-PLANCK-GESELLSCHAFT

Outline of the Lecture

- Organizational Updates
- Pre-training: Overview
- Pre-training: Scaling Laws
- Supervised Fine-tuning: Overview
- Supervised Fine-tuning: Parameter-efficient Fine-tuning

Outline of the Lecture

- **Organizational Updates**
- Pre-training: Overview
- Pre-training: Scaling Laws
- Supervised Fine-tuning: Overview
- Supervised Fine-tuning: Parameter-efficient Fine-tuning

Organizational Updates

- **Week 3 assignment – deadline extended**: Nov 7, 6pm CET (extended by 3 days because of holiday)
- **Week 4 assignment – deadline**: Nov 18, 6pm CET
- **Week 5 assignment – deadline**: Nov 25, 6pm CET

Outline of the Lecture

- Organizational Updates
- **Pre-training: Overview**
- Pre-training: Scaling Laws
- Supervised Fine-tuning: Overview
- Supervised Fine-tuning: Parameter-efficient Fine-tuning

Pre-Training

Main idea

- Train P_θ to predict the next token x_k from the previous tokens $(x_1, x_2, \dots, x_{k-1})$ in an unlabeled corpus $\mathcal{D} = (x_1, x_2, \dots, x_D)$, i.e., minimize the objective

$$\sum_{k=1}^{D} \text{prediction\_loss}(P_\theta(\cdot \mid x_1, \dots, x_{k-1}), x_k)$$

Loss function

- Specific prediction loss of interest: $\text{prediction\_loss}(p, x) = -\log p(x)$

Optimization

- Gradient-based methods

Pre-Training: Overview

Data curation

- Collecting and filtering large scale datasets
- Next few slides: challenges related to data curation

Model architecture and size

- Designing the model architecture and determining the model size
- 3rd part of the lecture: scaling laws

Training infrastructure and recipe

- Ensuring efficient pre-training at large scale
- Pre-training recipe: adjusting context-length and training data
- This lecture does not cover this aspect in detail

Ref: [Llama Team, 2024]

Data Curation

Data Source

- $\mathcal{D}$ is a large/internet scale dataset
- Obtained by crawling the web … or use *CommonCrawl* (> 2.5 billion webpages)

Data Processing

- Data is originally in the html format
- Contains harmful and toxic content
- Contains duplicates and low quality content
- Contains private data and copyrighted data
- Multiple languages and domains

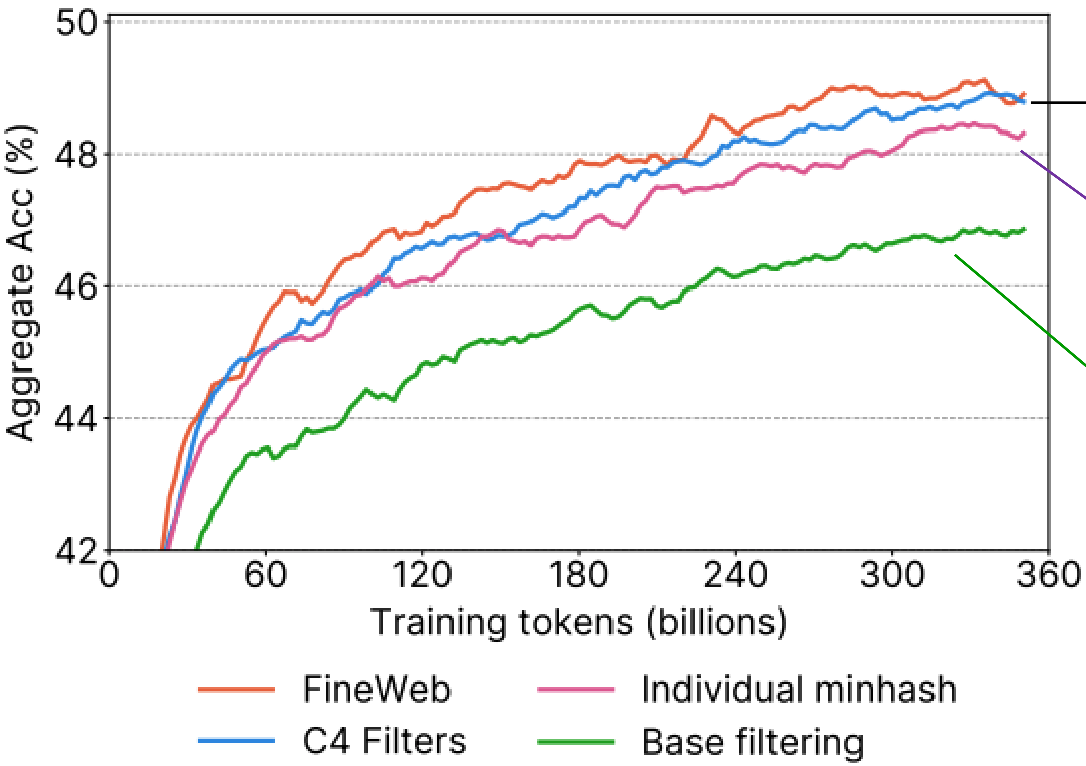
Additional Considerations

- Deciding on data mix
- Selecting annealing data

Data Curation: The Impportance of Filtering

Example: FineWeb Dataset

- 4 different filtering steps applied on 96 snapshots of CommonCrawl
- The size of the dataset: 15 trillion tokens



Heuristic filters

Deduplication

URL, language classification, quality filters

Outline of the Lecture

- Organizational Updates
- Pre-training: Overview
- **Pre-training: Scaling Laws**
- Supervised Fine-tuning: Overview
- Supervised Fine-tuning: Parameter-efficient Fine-tuning

Selecting Model Size

Challenge: Given a limited compute budget, what should be the size of our model?

- For a fixed budget, we need to make a trade-off between the model size and the number of training data points
- Model too large $\Longrightarrow$ won't be trained with enough data
- Model too small $\Longrightarrow$ underfitting

Example: Lamma 3 with **405B parameters**

- Compute budget of $3.8 \cdot 10^{25}$ FLOPs (Floating Point Operations)
- Layers: 126
- Model dimension: 16384
- Attention heads: 128 …

*"… This leads to a model size that is approximately compute-optimal according to **scaling laws** on our data for our training budget …"*

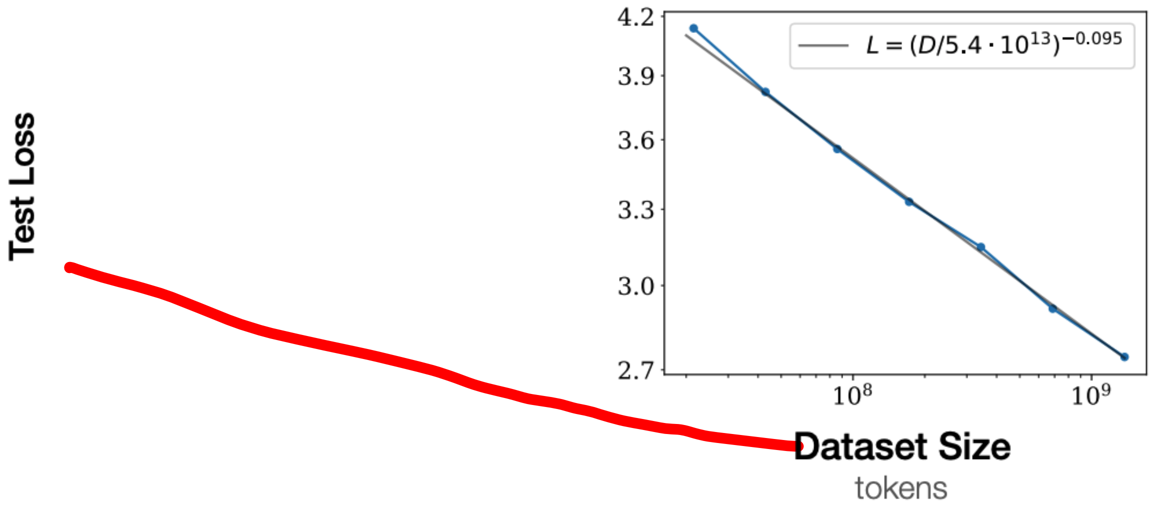
Simple Scaling Law

Illustrative example of scaling laws:

- Mean estimation: samples $x_1, \dots, x_D \sim \mathcal{N}(\mu, \sigma)$, and estimator $\hat{\mu} = \frac{1}{D}\sum_i x_i$
- Loss: mean squared error $\mathbb{E}[(\mu - \hat{\mu})^2]$
- Possible to show that $\text{Loss} = \frac{\sigma^2}{D} = \sigma^2 \cdot D^{-1}$ → this is a power law
- This means that $\log \text{Loss} = -\log D + \text{constant}$

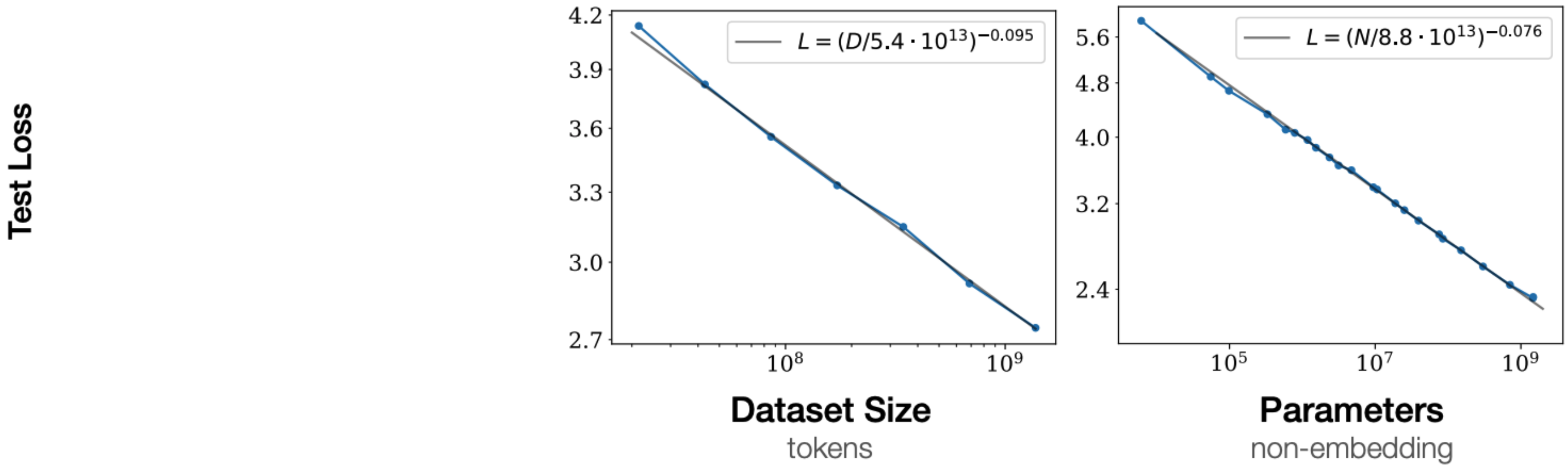
Scaling Laws for LLMs

- We can obtain similar scaling laws for LLMs through experiments



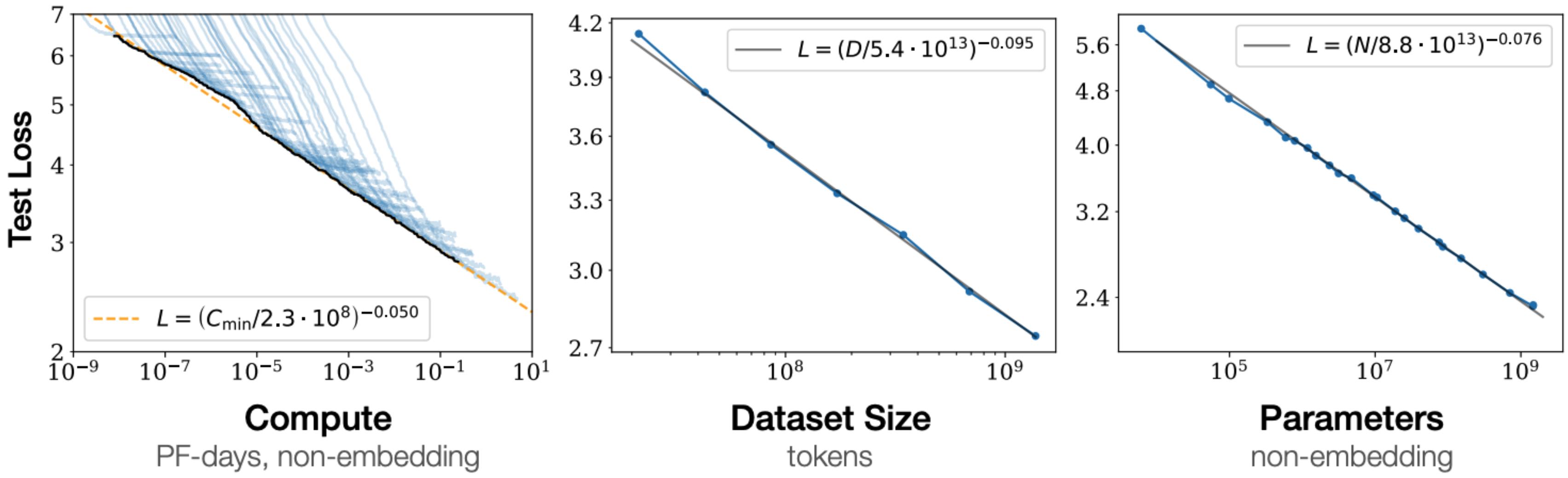
Scaling Laws for LLMs

- We can obtain similar scaling laws for LLMs through experiments



Scaling Laws for LLMs

- We can obtain similar scaling laws for LLMs through experiments



- So far: *Test error* as a function of *compute*, *dataset size*, or *model size* – when increasing one of these, we are not bottlenecked by the other two
- The behavior is predictable:
  - Infer scaling laws using small compute budgets and then infer the optimal model/dataset size for a given compute budget

Scaling Laws: Model Size

Determining optimal allocations

- **Input**: Dataset $\{(N_i, D_i, L_i)\}$, where N_i is the number of parameters, D_i is the number of training tokens, and L_i is the observed loss
- **Objective**: Find N and D that minimize loss $L(N, D)$ for a given budget C:

$$\min_{N,D} L(N, D) \;\; \text{s.t.} \;\; C = \text{FLOPs}(N, D)$$

- Consider $L(N, D)$ of the following form:

$$L(N, D) = E + \frac{A}{N^{\alpha}} + \frac{B}{D^{\beta}}$$

Captures:

Captures:	Entropy of natural text	Suboptimality of function approx.	Suboptimality of optimization

Scaling Laws: Model Size

Determining optimal allocations

- **Input**: Dataset $\{(N_i, D_i, L_i)\}$, where N_i is the number of parameters, D_i is the number of training tokens, and L_i is the observed loss
- **Objective**: Find N and D that minimize loss $L(N, D)$ for a given budget C:

$$\min_{N,D} L(N, D) \ \text{s.t.}\ C = \text{FLOPs}(N, D)$$

- Consider $L(N, D)$ of the following form:

$$L(N, D) = \frac{A}{N^{\alpha}} + \frac{B}{D^{\beta}} + E$$

- Estimate α, β, A, B, and E, by fitting $L(N, D)$ on the dataset $\{(N_i, D_i, L_i)\}$.

Scaling Laws: Model Size

Determining optimal allocations

- **Input**: Dataset $\{(N_i, D_i, L_i)\}$, where N_i is the number of parameters, D_i is the number of training tokens, and L_i is the observed loss
- **Objective**: Find N and D that minimize loss $L(N, D)$ for a given budget C:

$$\min_{N,D} L(N, D) \;\; \text{s.t.} \;\; C = \text{FLOPs}(N, D)$$

- Consider $L(N, D)$ of the following form:

$$L(N, D) = \frac{A}{N^\alpha} + \frac{B}{D^\beta} + E$$

- **Fact**: Compute C is related to N and D: $C \approx 6ND$
  - This is due to forward and backward pass in backpropagation (See **Computing Gradients*)

Scaling Laws: Model Size

Determining optimal allocations

- **Input**: Dataset $\{(N_i, D_i, L_i)\}$, where N_i is the number of parameters, D_i is the number of training tokens, and L_i is the observed loss
- **Objective**: Find N and D that minimize loss $L(N, D)$ for a given budget C:

$$\min_{N,D} L(N, D) \ \text{s.t.}\ C = \text{FLOPs}(N, D)$$

- Consider $L(N, D)$ of the following form:

$$L(N, D) = \frac{A}{N^\alpha} + \frac{B}{D^\beta} + E$$

- Using $C \approx 6ND$, we obtain:

$$L(N) = \frac{A}{N^\alpha} + \frac{B}{C^\beta}(6N)^\beta + E \qquad L(D) = \frac{A}{C^\alpha}(6D)^\alpha + \frac{B}{D^\beta} + E$$

Scaling Laws: Model Size

Example

- Suppose that $\alpha = 0.3478$, $\beta = 0.3658$, $A = 482.01$, $B = 2085.43$, and $E = 2085.43$. How does optimal N_{opt} scale with C?

导数

$$L(N) = \frac{A}{N^{\alpha}} + \frac{B}{C^{\beta}}(6N)^{\beta} + E$$

- By setting $\frac{dL(N)}{dN} = 0$, we obtain:

$$\alpha \frac{A}{{N_{opt}}^{\alpha+1}} = \beta \frac{B}{C^{\beta}} 6^{\beta} {N_{opt}}^{\beta-1} \Rightarrow N_{opt} \propto C^{\frac{\beta}{\alpha+\beta}} \approx C^{0.513}$$

→ Power law

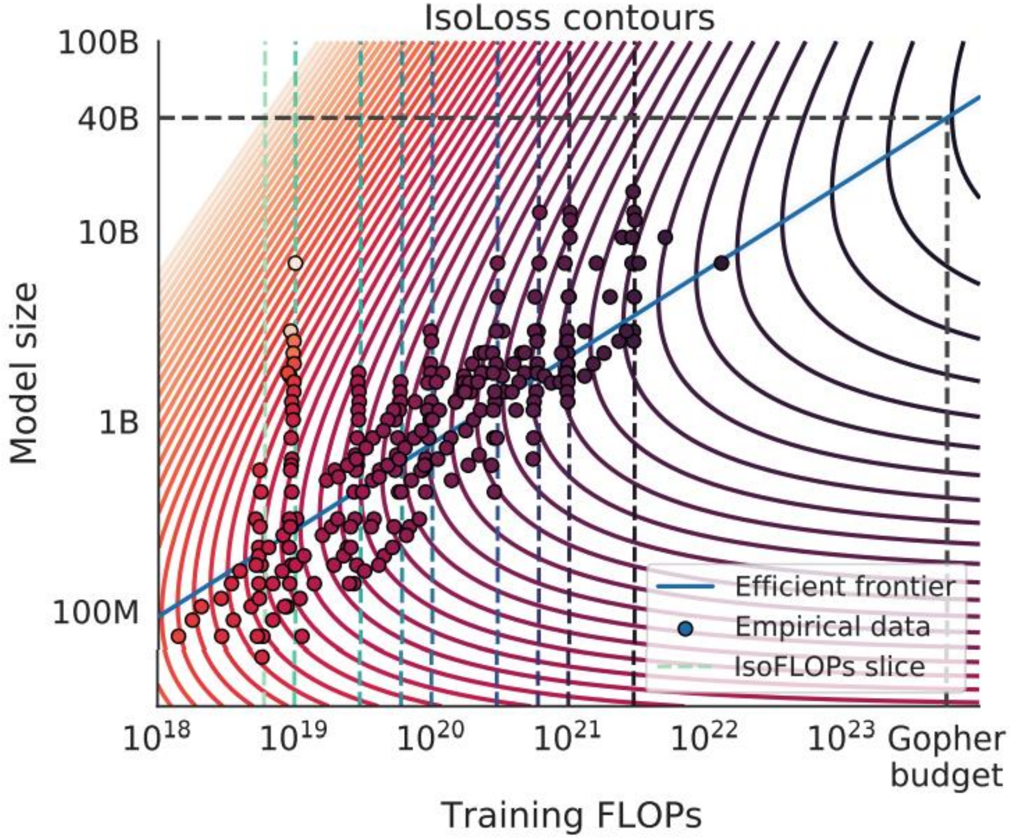
- For $C = 5.76 \cdot 10^{23}$, we obtain $N_{opt} \approx 72B$

- We can analogously obtain $D_{opt} \propto C^{\frac{\alpha}{\alpha+\beta}} \approx C^{0.487}$ →

≈ For every doubling of model size, the number of training tokens should double

Scaling Laws: Model Size

- The efficient frontier is shown in blue



- **Remark**: On the previous slide, we used different coefficients α, β, A, B, E!

Scaling Laws: Model Size

Other Approaches

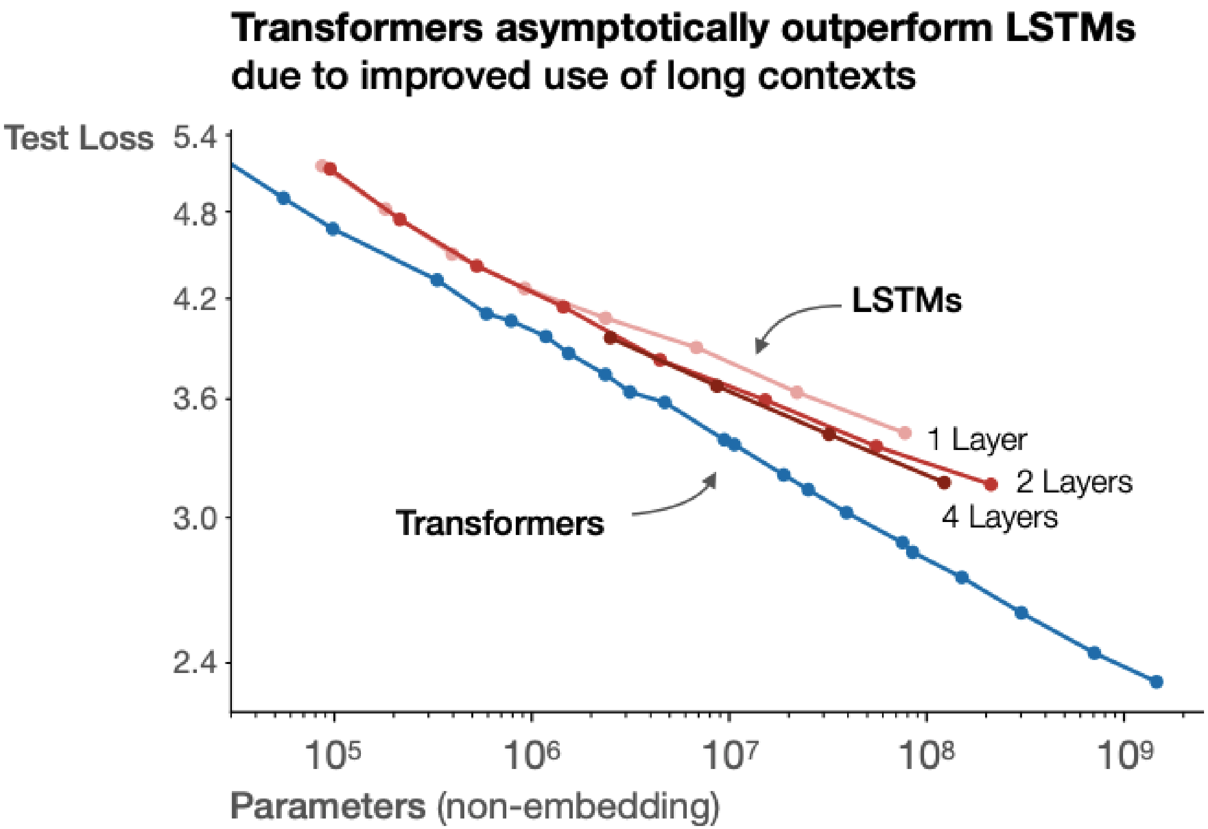
- Other approaches to determining scaling laws yield similar results.
- **Approach I**: Vary the number of training steps for a fixed family of models, and extract an estimate of the minimum loss for a given budget. Identify the model size that achieves the minimum loss.
- **Approach II**: For each compute budget from a set of compute budgets, vary the model size and identify which one achieves the minimum loss for that budget.
- Fit scaling laws based on the optimal model sizes obtained for the compute budgets.

Week 4 Assignment

- A reading assignment: a paper that explains Approach I and Approach II.
- An exercise on scaling laws where the amount of available data is constrained.

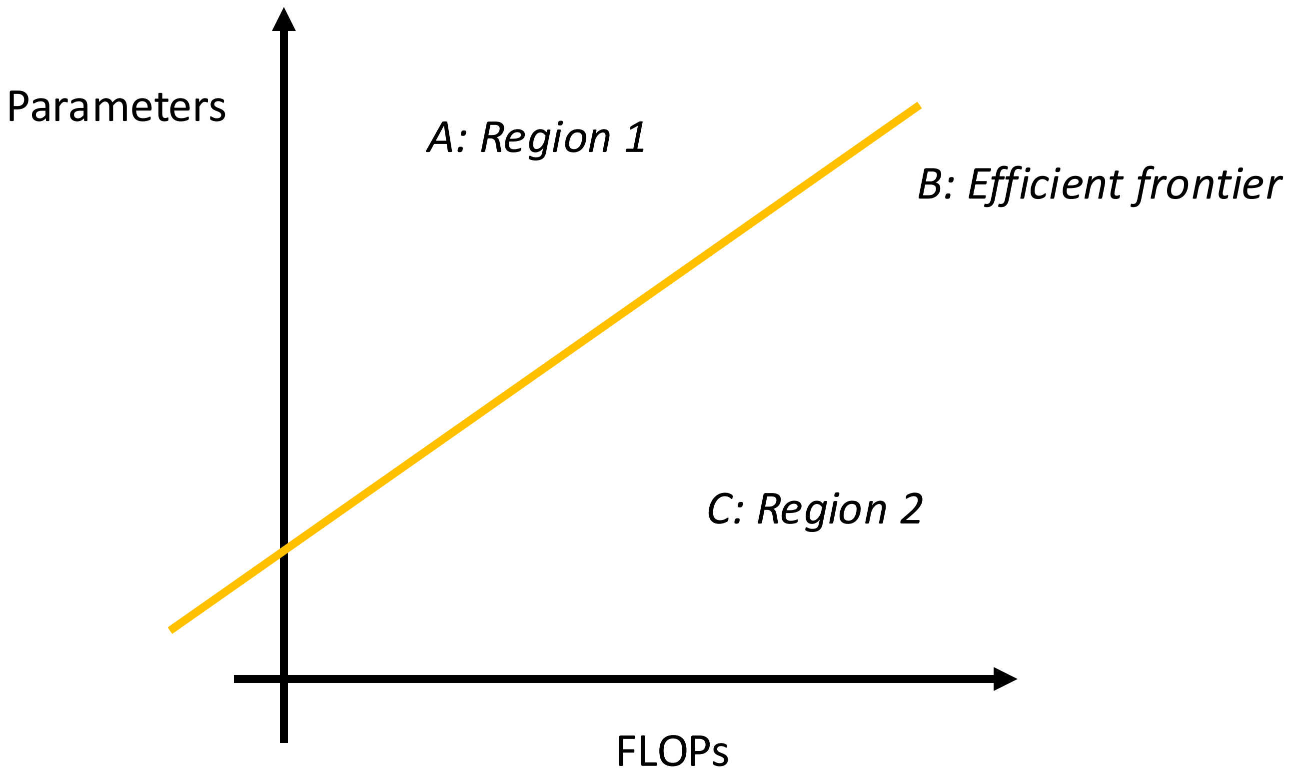
Quiz – Scaling Laws

Q: Do scaling laws hold for other architectures (e.g., LSTMs)?



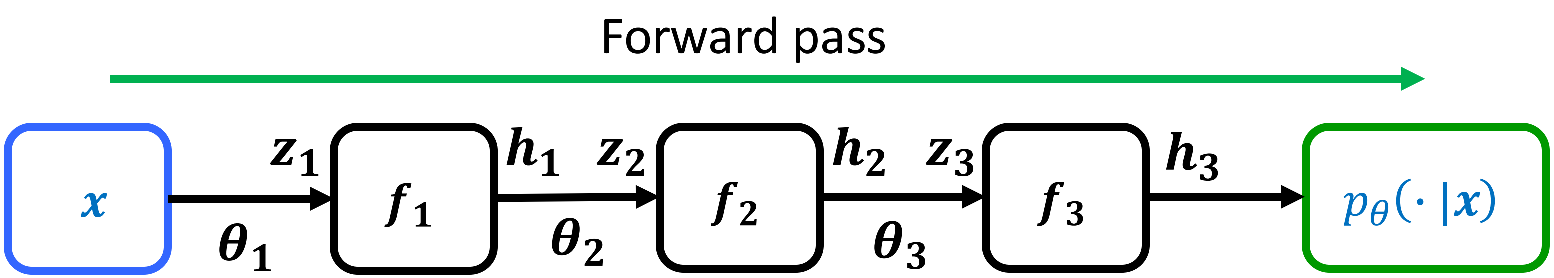
Quiz – Scaling Laws

Q: Where to choose from if we account for inference costs?



*Computing Gradients (Optional)

- **Reminder**: gradient update rule for loss $\mathcal{L}(\theta)$: $\theta \leftarrow \theta - \text{learn\_rate} \cdot \nabla_\theta \mathcal{L}(\theta)$
- We can use *backpropagation* to obtain $\nabla_\theta \mathcal{L}(\theta)$
- A simplified illustration (based on layered feedforward NN):

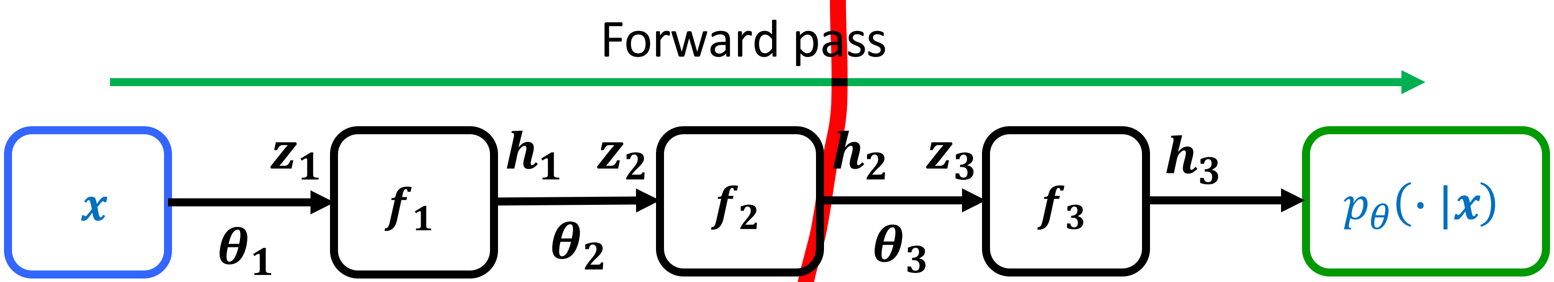


- If we only have 1 parameter per layer and h_i are scalars, what is $\frac{\partial \mathcal{L}(\theta)}{\partial \theta_i}$?

By the chain rule: $$\frac{\partial \mathcal{L}(\theta)}{\partial \theta_2} = \frac{\partial z_2}{\partial \theta_2} \cdot \frac{\partial \mathcal{L}(\theta)}{\partial z_2}$$

*Computing Gradients (Optional)

- **Reminder**: gradient update rule for loss $\mathcal{L}(\theta)$: $\theta \leftarrow \theta - \text{learn\_rate} \cdot \nabla_\theta \mathcal{L}(\theta)$
- We can use *backpropagation* to obtain $\nabla_\theta \mathcal{L}(\theta)$
- A simplified illustration (based on layered feedforward NN):

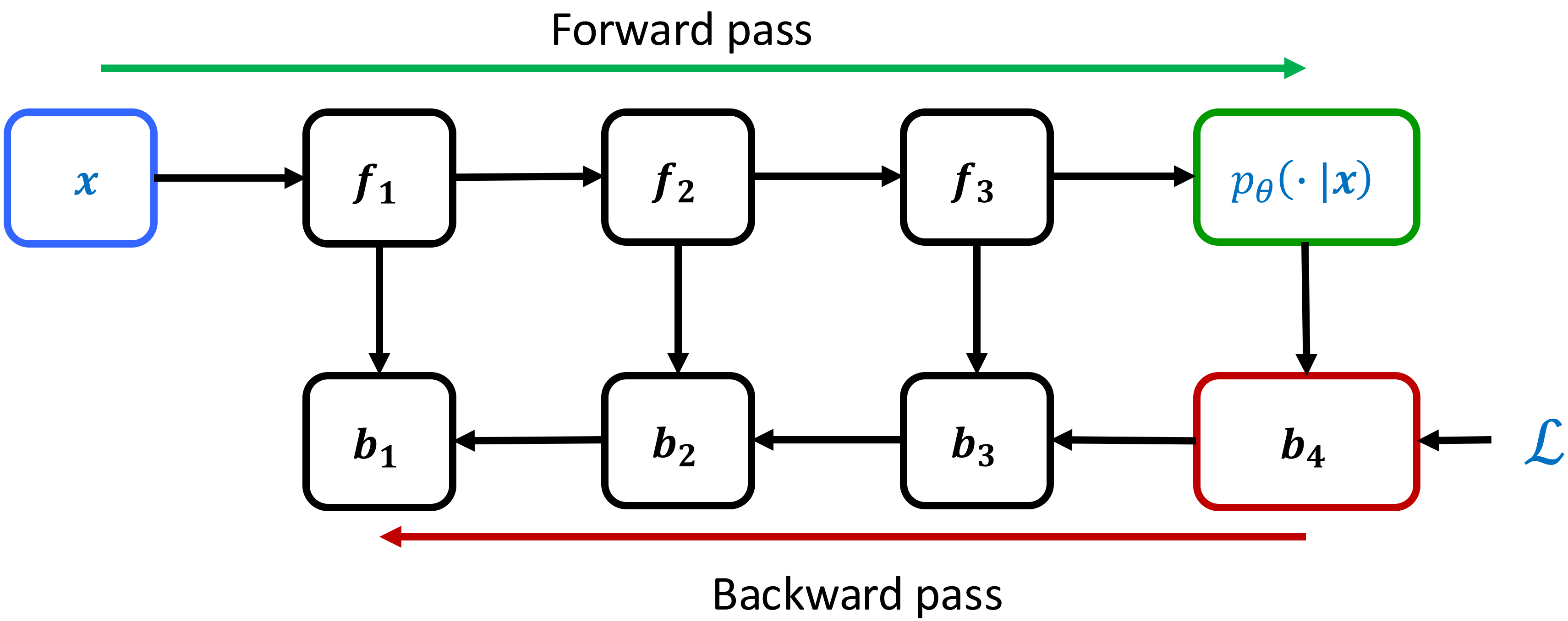


- If we only have 1 parameter per layer and h_i are scalars, what is $\frac{\partial \mathcal{L}(\theta)}{\partial \theta_i}$?

By the chain rule: $\frac{\partial \mathcal{L}(\theta)}{\partial \theta_2} = h_1 \boxed{\frac{\partial \mathcal{L}(\theta)}{\partial z_2}}$ ⇨ Information that we calculate from the future layers

*Computing Gradients (Optional)

- **Reminder**: gradient update rule for loss $\mathcal{L}(\theta)$: $\theta \leftarrow \theta - \text{learn\_rate} \cdot \nabla_\theta \mathcal{L}(\theta)$
- We can use *backpropagation* to obtain $\nabla_\theta \mathcal{L}(\theta)$
- A simplified illustration (based on layered feedforward NN):



*Computing Gradients (Optional)

- **Reminder**: gradient update rule for loss $\mathcal{L}(\theta)$: $\theta \leftarrow \theta - \text{learn\_rate} \cdot \nabla_\theta \mathcal{L}(\theta)$
- We can use *backpropagation* to obtain $\nabla_\theta \mathcal{L}(\theta)$

Compute requirements

- For one data input, about $6 \cdot \text{number of parameters}$
- In the forward pass, matrix multiplications are a dominant factor: each parameter is associated with 1 multiplication and 1 summation
- The cost of the backward pass is approx. 2 times the cost of the forward pass

Memory requirements

- Weights, gradients, optimizer states, activations, etc.

Outline of the Lecture

- Organizational Updates
- Pre-training: Overview
- Pre-training: Scaling Laws
- **Supervised Fine-tuning: Overview**
- Supervised Fine-tuning: Parameter-efficient Fine-tuning

Why Fine-tuning?

- Pre-training enables model to understand language, but not necessarily to follow instructions

Prompt:
Create a shopping list from this recipe:
Trim the ends off zucchini. Cut zucchini in half lengthwise; scoop out pulp, leaving 1/2-in. shells. Finely chop pulp. In a skillet, cook beef, zucchini pulp, onion, mushrooms and peppers over medium heat until meat is no longer pink; drain. Remove from the heat. Add 1/2 cup cheese, ketchup, salt and pepper; mix well. Spoon into the zucchini shells. Place in a greased 13x9-in. baking dish. Sprinkle with remaining cheese.

Labeler demonstration
ucchini, beef, onion, mushroom, peppers, cheese, ketchup, salt, pepper

GPT-3 175B completion:
Bake, uncovered, at 350° for 20-25 minutes or until zucchini is tender and cheese is melted.

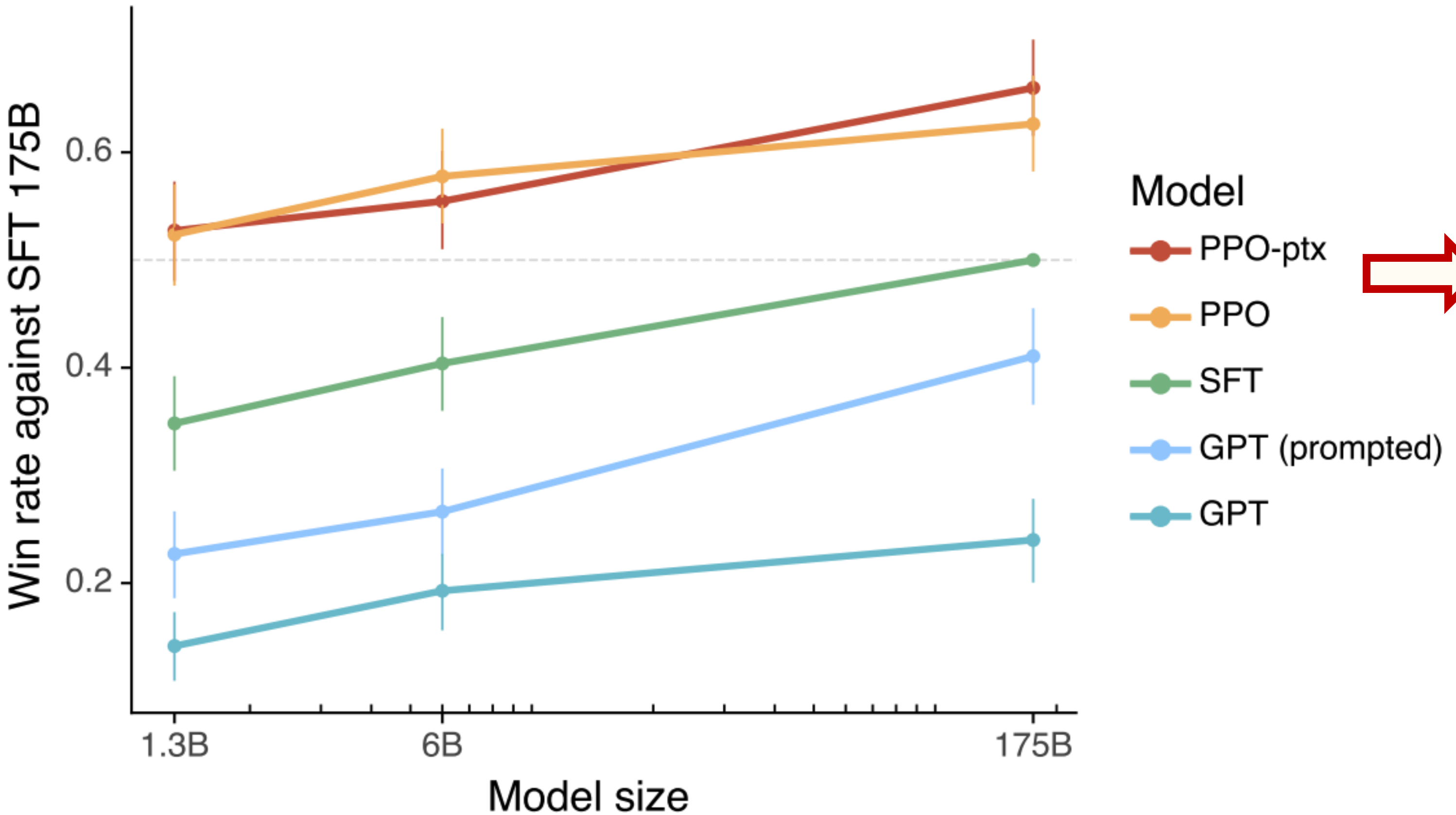
InstructGPT 175B completion:
Bake, uncovered, at 350° for 20-25 minutes.

- 8 zucchini squash, ends trimmed
- 1 pound ground beef
- 1 cup finely chopped zucchini pulp
- 1 small onion, chopped
- 8 ounces diced mushrooms
- 1 sweet red or green pepper, diced
- 1/2 cup grated cheese
- 1/4 cup ketchup
- Salt and pepper to taste
- 1 cup shredded cheese

Ref: [Ouyng et al., 2021]

Why Fine-tuning?

- Human evaluations of the outputs



Next lecture

Supervised Fine-tuning

Main idea

- Now the dataset is labelled: $\mathcal{D} = \{(x_p, y)\}$

  Examples:

  Alignment: x_p can be an instruction and y can be a demonstration

  Downstream task: x_p can be a text and y can be a summary

- The same objective (next token prediction) and loss as in pre-training, but only applied over response y

$$\max_\theta \sum_{(x_p, y) \in \mathcal{D}} \sum_{k=1}^{|y|} \log P_\theta(y_k | x_p, y_1, \dots, y_{k-1})$$

Supervised Fine-tuning

Main idea

- Now the dataset is labelled: $\mathcal{D} = \{(x_p, y)\}$

  Examples:

  Alignment: x_p can be an instruction and y can be a demonstration

  Downstream task: x_p can be a text and y can be a summary

- The same objective (next token prediction) and loss as in pre-training, but only applied over response y

Week 4 Assignment

- SFT for Shakespeare completion and text summarization

Note: There are other forms of fine-tuning

- *Continued pre-training*: continue pre-training on a specific domain
- *Preference-based fine-tuning*: Next lecture

Fine-tuning Quiz

Q: Suppose that we have 100 downstream tasks to consider. How would you fine-tune GPT3 175B for these 100 tasks?

Week 4 Assignment

- An exercise demonstrating the importance of parameter-efficient fine-tuning.

Outline of the Lecture

- Organizational Updates
- Pre-training: Overview
- Pre-training: Scaling Laws
- Fine-tuning: Overview
- **Fine-tuning: Parameter-efficient Fine-tuning**

Parameter-efficient Fine-tuning

- Reduce the number of trainable parameters. Different approaches, e.g.:
  - Prefix-tuning: Prepends special tokens with trainable embeddings to the input
  - Adapter Layers: Add trainable layers to the transformer network
  - …

LoRA: Low-Rank Adaptation

- **Main idea**: Encode finetuning parameter increment $\Delta\theta$ using a much smaller set of parameters ϕ
- The SFT objective is to optimize ϕ using the next-token prediction loss:

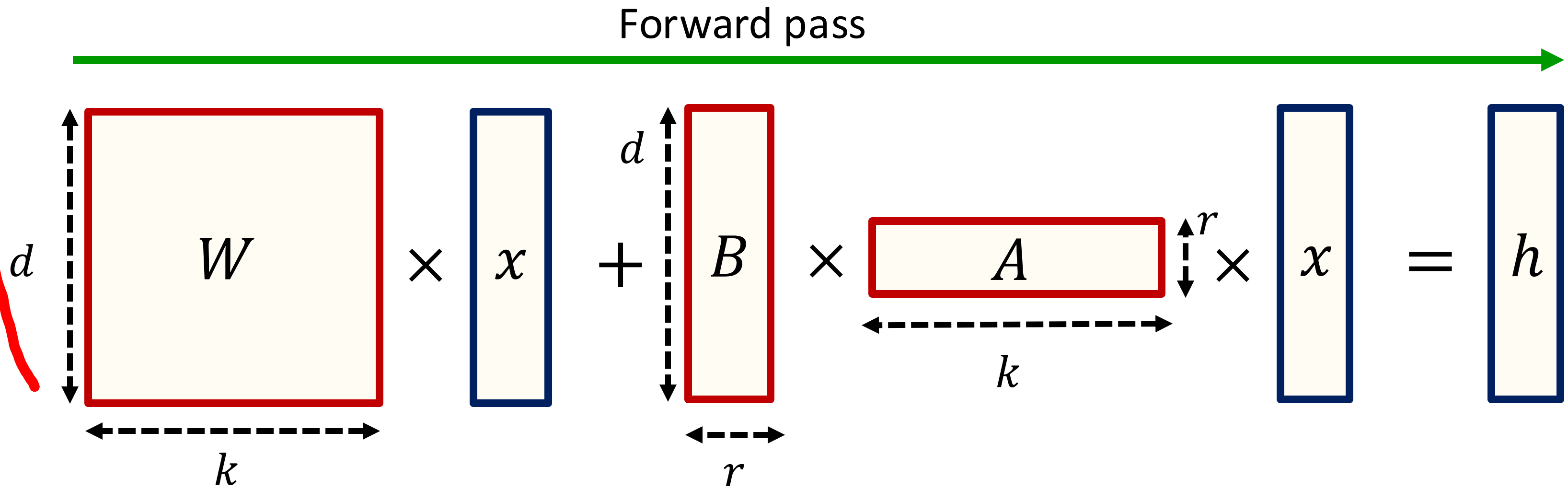
$$\max_{\phi} \sum_{(x_p,y)\in\mathcal{D}} \sum_{k=1}^{|y|} \log P_{\theta+\Delta\theta(\phi)}(y_k | x_p, y_1, \ldots, y_{k-1})$$

Parameter-efficient Fine-tuning

- Reduce the number of trainable parameters. Different approaches, e.g.:
  - Prefix-tuning: Prepends special tokens with trainable embeddings to the input
  - Adapter Layers: Add trainable layers to the transformer network
  - …

LoRA: Low-Rank Adaptation

- **Main idea**: Add trainable rank decomposition matrices
- Illustration:



Parameter-efficient Fine-tuning

LoRA: Low-Rank Adaptation

- Freeze pretrained model – the model weights are not updated
- For a weight matrix $W \in \mathbb{R}^{d\times k}$, define trainable matrices $B \in \mathbb{R}^{d\times r}$ and $A \in \mathbb{R}^{r\times k}$, with $r \ll d$ and $r \ll k$
  - We can choose which matrices W, typically from the self-attention module
- Modified forward pass: $h = Wx + s \cdot \Delta Wx = Wx + s \cdot BAx$, where s is a scalar, which is often defined through r
- Train only A and B: A is initialized with a random Gaussian initialization, while B is initially set to $\mathbf{0}$

- For different tasks: Use the same W but different A and B!

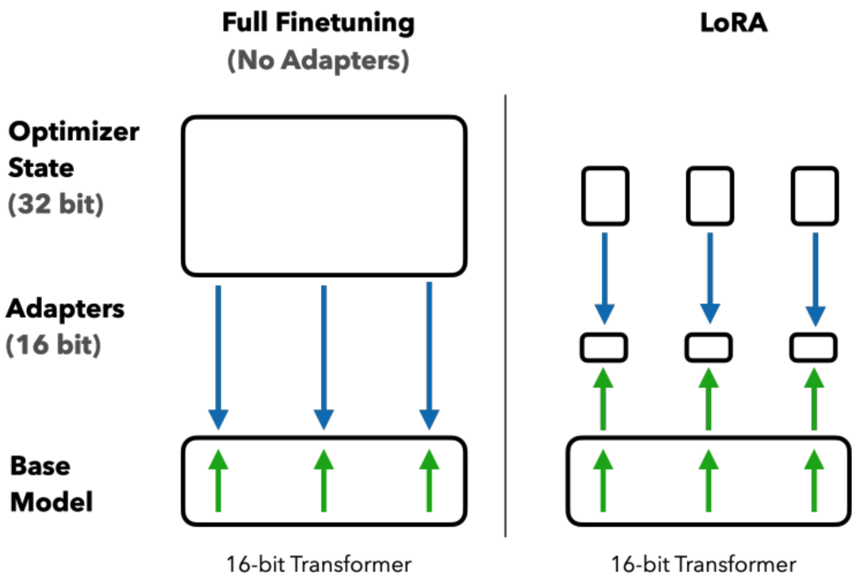
Week 4 Assignment

- An exercise on the storage-efficiency of LoRA.

Quiz – LoRA

Q: Consider GPT3 175B. What is the memory usage reduction if we set $r = 1$? Can we train it on H100 GPU with 80GB?

- Model weights also need to be stored!



- ... but the memory is reduced for optimizer states and gradients.
- **Quantization** can reduce the memory requirement for model weights

Quantization

Basic idea

- Quantize weights when storing them
- Dequantize them when needed, e.g., for inference

Example

- Quantizing 32-bit floating point tensor into 8-bit int tensor

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

8bit -128 -> 128

absmax(x
FP32
)：计算该张量中的最大绝对值，作为缩放因子。

De-quantization: $$\mathbf{X}^{\text{FP32}} \leftarrow \frac{\mathbf{X}^{\text{Int8}}}{c}$$

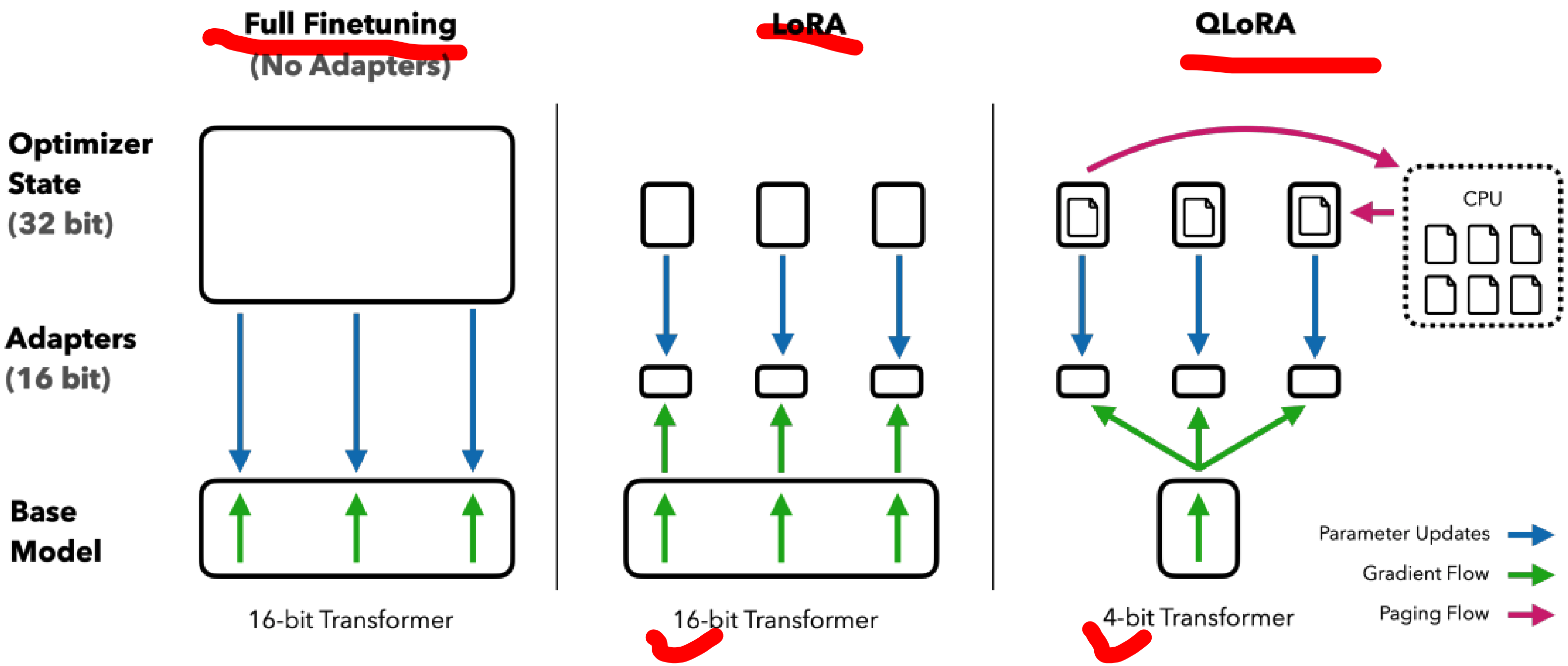
- Quantization can be done block-wise, each having its own quant. constant c_B
  - Motivation: if one element of $\mathbf{X}^{\text{FP32}}$ has a large value, many quantization bins (integer values) are not utilized

4. 块级量化（Block-wise Quantization）
问题：如果
中某些值很大，则小的数值信息可能丢失，导致精度下降。
解决方案：块级量化（Block-wise Quantization），
即不同块的数值采用不同的缩放因子
，提高精度。

*Quantization: QLoRA (Optional)

QLoRA:

- A quantized version of LoRA with a specific type of quantization



Week 4 Assignment

- Optional: reading materials and an exercise on QLoRA

Summary

- Pre-training is not only about self-supervised learning: data curation, selecting model architecture, pre-training recipe
- We can utilize scaling laws to determine which models to train
- Importance of fine-tuning: pre-training does not suffice for creating helpful assistants
- Supervised fine-tuning: useful for alignment and adaptation to downstream tasks
- Parameter-efficient fine-tuning can reduce the memory footprint
- **Next lecture**: preference-based fine-tuning for alignment!

References

- Radford et al., Improving Language Understanding by Generative Pre-Training, 2018.
- Llama Team, The Llama 3 Herd of Models, 2024.
- Kaplan et al., Scaling Laws for Neural Language Models, 2020.
- Hofmann et al., Training Compute Optimal Language Models, 2022.
- Besiroglu et al., Chinchilla Scaling: A Replication Attempt, 2024.
- Muennighoff et al., Scaling Data-constrained Language Models, 2023.
- Penedo et al., The FineWeb Datasets: Decanting the Web for the Finest Text Data at Scale, 2024.
- Ouyang et al., Training Language Models to Follow Instructions with Human Feedback, 2022.
- Hu et al., LoRA: Low-Rank Adaptation of Large Language Models, 2021.
- Book: Jurafsky and Martin, Speech and Language Processing, 2024.
- Dettmers et al., QLoRA: Efficient Finetuning of Quantized LLMs, 2023.

- ***Acknowledgements***: The content of this lecture is partly based on lectures from Stanford courses CS336 (https://stanford-cs336.github.io/spring2024/) and CS229 (more specifically, the guest lecture: https://www.youtube.com/watch?v=9vM4p9NN0Ts).