Text Analysis and Retrieval

7. Neural NLP – Transformers

Josip Jukić

University of Zagreb
Faculty of Electrical Engineering and Computing (FER)

Academic Year 2022/2023

v3.0

Outline

Outline

Self-attention

- Self-attention (intra-attention) is a powerful mechanism focusing on the relationships between various positions within a single sequence to generate a comprehensive representation
- The model weighs the importance of different elements in a sequence, ultimately resulting in more effective representations

Self-attention

- Self-attention (intra-attention) is a powerful mechanism focusing on the relationships between various positions within a single sequence to generate a comprehensive representation
- The model weighs the importance of different elements in a sequence, ultimately resulting in more effective representations

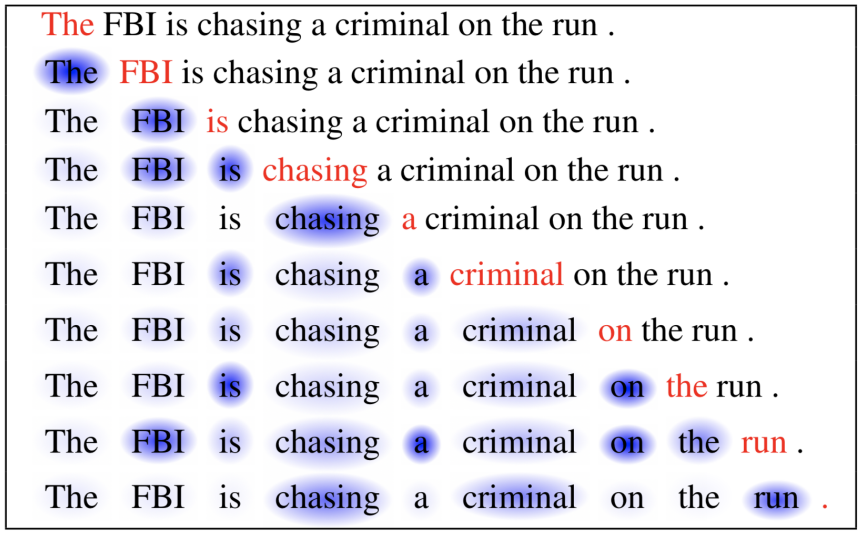


Figure: Self-attention: the current word is in red, and the intensity of the blue shade indicates the activation level (Cheng et al., 2016)

Fully attentional networks

The attention mechanism proved to be an excellent improvement, helping with learning *long-term* dependencies as well as *cleaner* gradient propagation through the linear combination.

- Idea: a model based only on attention?
- We can parallelize the computation of tokens in the input sequence, but what about the positional information of these tokens?

Fully attentional networks

The attention mechanism proved to be an excellent improvement, helping with learning *long-term* dependencies as well as *cleaner* gradient propagation through the linear combination.

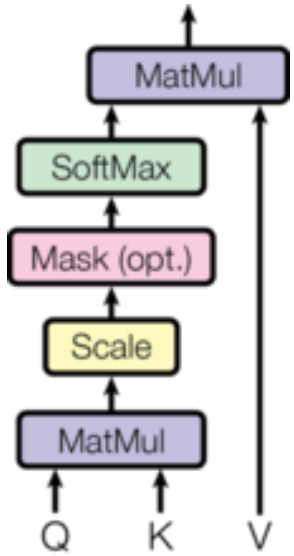
- Idea: a model based only on attention?
- We can parallelize the computation of tokens in the input sequence, but what about the positional information of these tokens?

Positional embeddings

1. Fixed positional embeddings
   - Sine and cosine waves
   - $PE_{pos,2i} = sin(pos/10000^{2i/d_{model}})$
   - $PE_{pos,2i+1} = cos(pos/10000^{2i/d_{model}})$
   - Cosine similarity between *adjacent positions* is 0.9
2. Learned positional embeddings
   - Initialize randomly and learn by backpropagation

Attention in fully attentional networks

In fully attentional networks, each token in the input sequence attends to all other tokens![1]



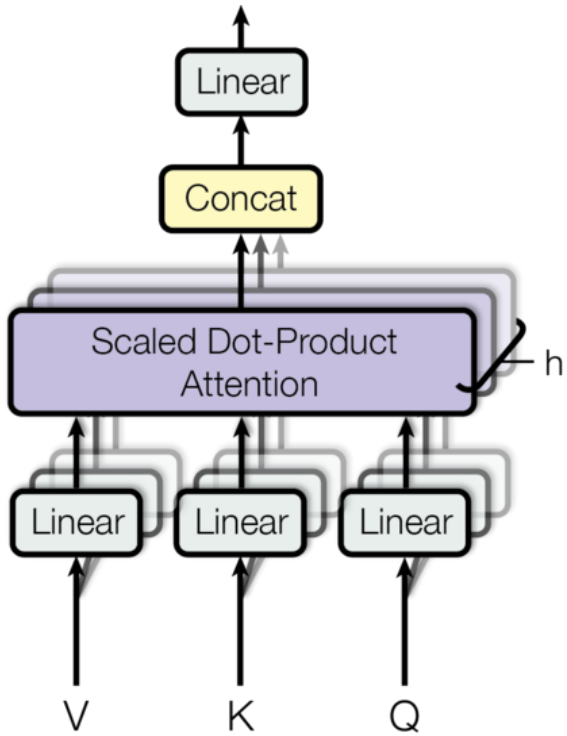
- We now also have T query vectors (one for each input). We can compactly write the attention expression as:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$

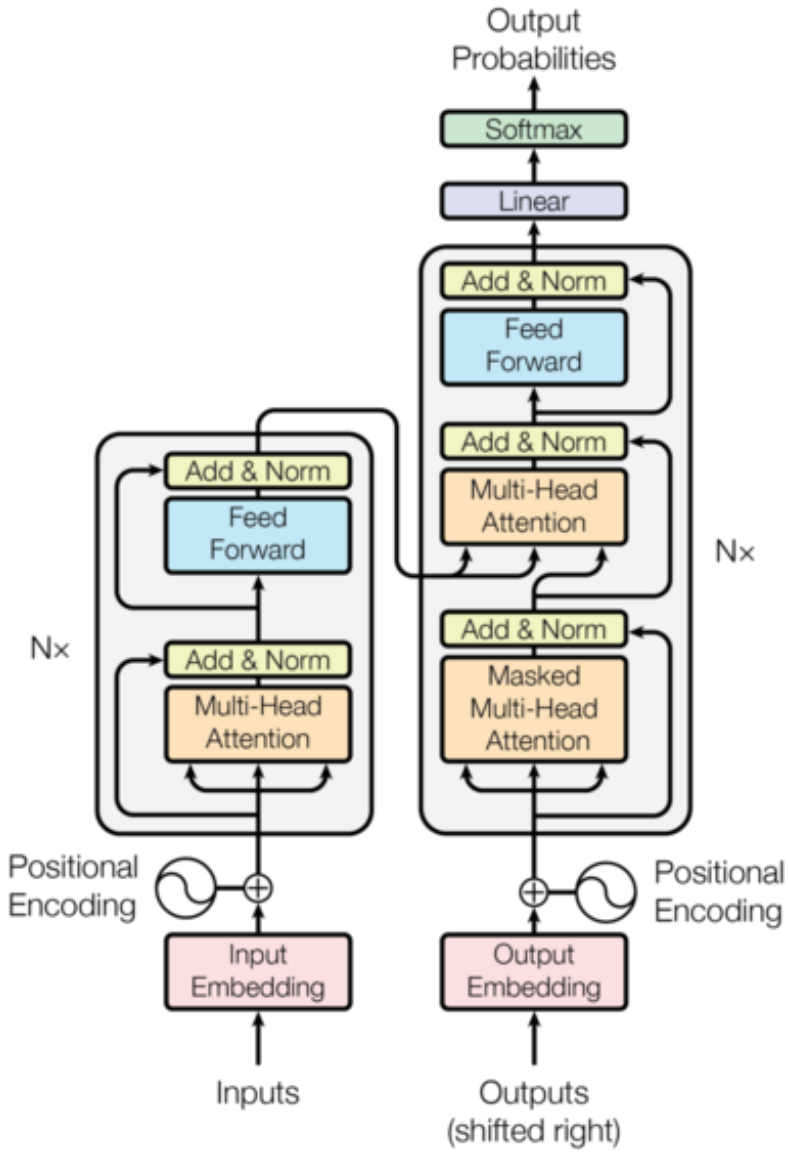
[1] except in cases where this would introduce information leakage

Multi-head attention (MHA)

- Multiple query vectors for multiple linear combinations (each head has its own set of parameters)
- Idea: cover different aspects of context
- The independent attention outputs are concatenated and linearly transformed into the expected dimension



The Transformer architecture



The Transformer architecture

For a thorough overview of the (initial) Transformer network, refer to Alexander Rush's "The annotated Transformer":
http://nlp.seas.harvard.edu/2018/04/03/attention.html

Some tricks and components used in the implementation:

- **Residual connections**: output each transformer block (MHA + linear) is summed to the input of that block
- **Layer normalization**: smoother gradients, faster training, and better generalization accuracy
- **Byte-pair encoding**: "subword" vocabulary

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

By Devlin et al., NAACL **2019.**

- BERT stands for Bidirectional Encoder Representations from Transformers

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

By Devlin et al., NAACL **2019.**

- BERT stands for Bidirectional Encoder Representations from Transformers

Idea:

1. Use MLM style training and deep transformers instead of LSTMs
2. Use the trained network as a **sentence encoder** for other tasks
3. Profit (*significant* performance gains across tasks)

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

By Devlin et al., NAACL **2019.**

- BERT stands for Bidirectional Encoder Representations from Transformers

Idea:

1. Use MLM style training and deep transformers instead of LSTMs
2. Use the trained network as a **sentence encoder** for other tasks
3. Profit (*significant* performance gains across tasks)

Tricks:

- MLM procedure (80% replace token with [MASK], 10% keep token, 10% replace with random token)
- Byte-pair encoding
- Learned positional embeddings

Learning outcomes 1

1. Explain the attention mechanism in fully attentional networks
2. Describe the Transformer architecture in terms of its main components
3. Describe BERT in terms of its purpose, the underlying neural model, and the prediction task

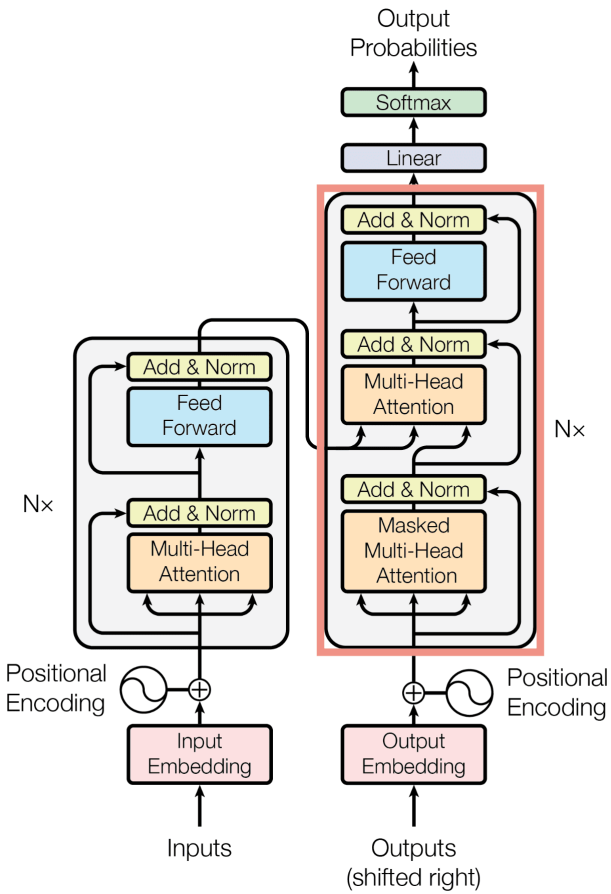
Outline

Autoregressive Transformers

- Can we leverage the Transformer architecture to generate text?
- Our language model should generate one token at a time $\rightarrow$ causal language modeling (CLM)

Autoregressive Transformers

- Can we leverage the Transformer architecture to generate text?
- Our language model should generate one token at a time $\rightarrow$ causal language modeling (CLM)

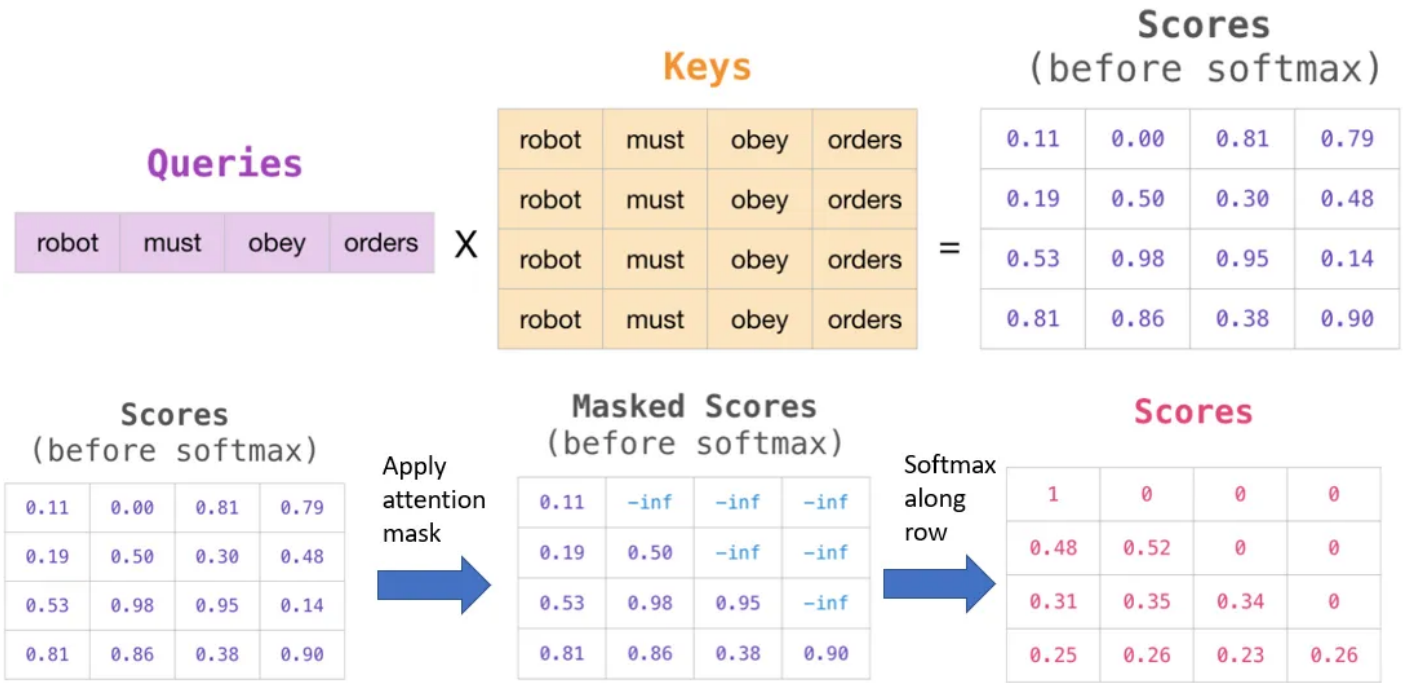


Attention in autoregressive language models

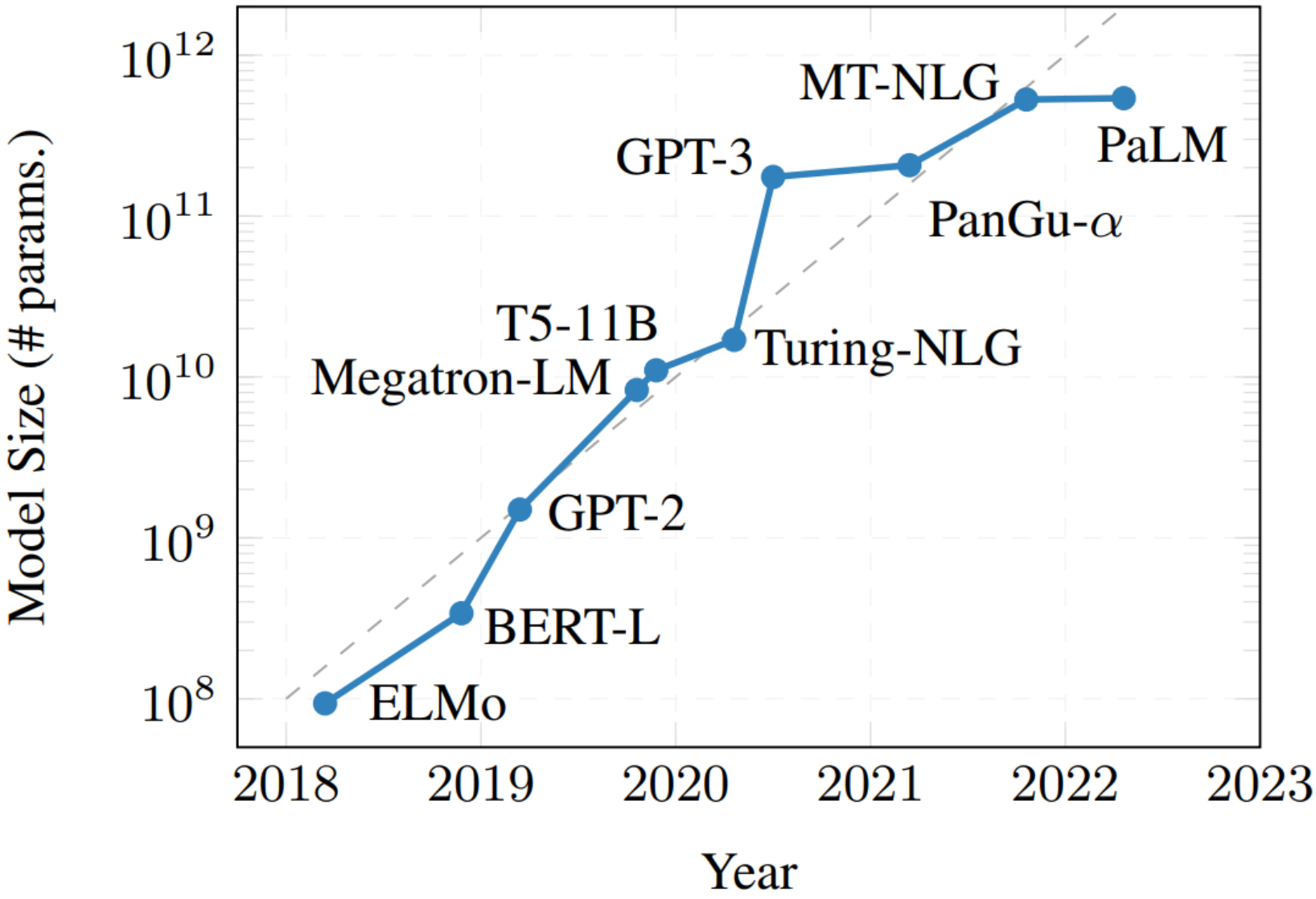
- We need to modify the attention mechanism by **masking** the future tokens

Attention in autoregressive language models

- We need to modify the attention mechanism by **masking** the future tokens



Landscape of large language models



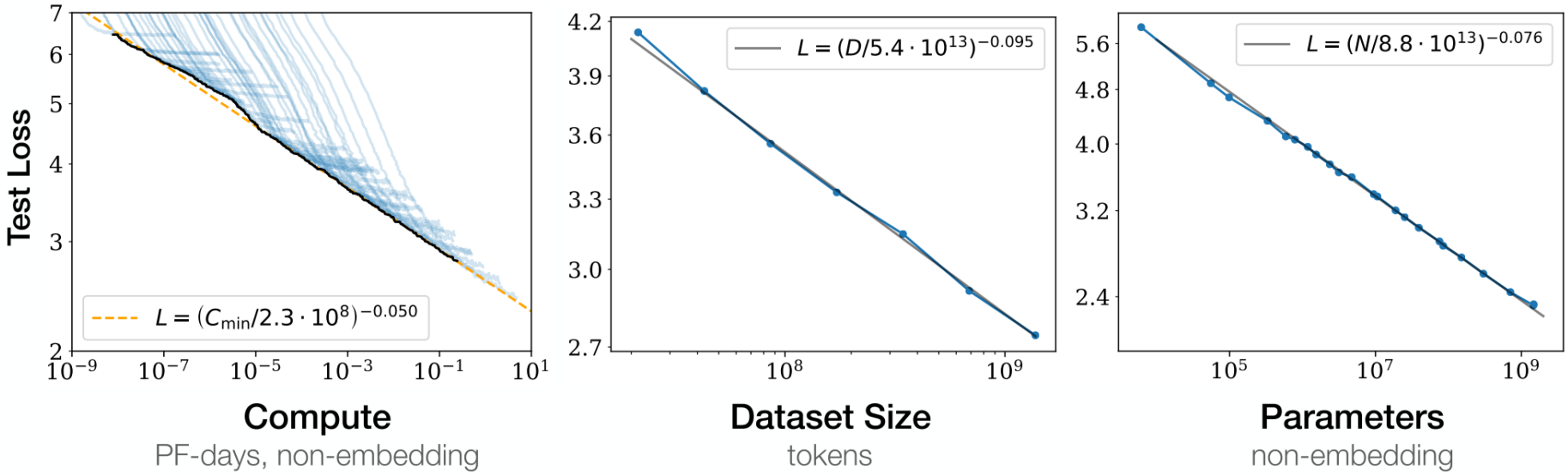
Scaling laws for large language models

- If we continue to decrease the test loss, LLMs show **emerging properties**
- LLMs are not directly trained to have these abilities, and they appear in **rapid** and **unpredictable** ways (e.g., arithmetic, summarization, question answering, understanding jokes, . . .)

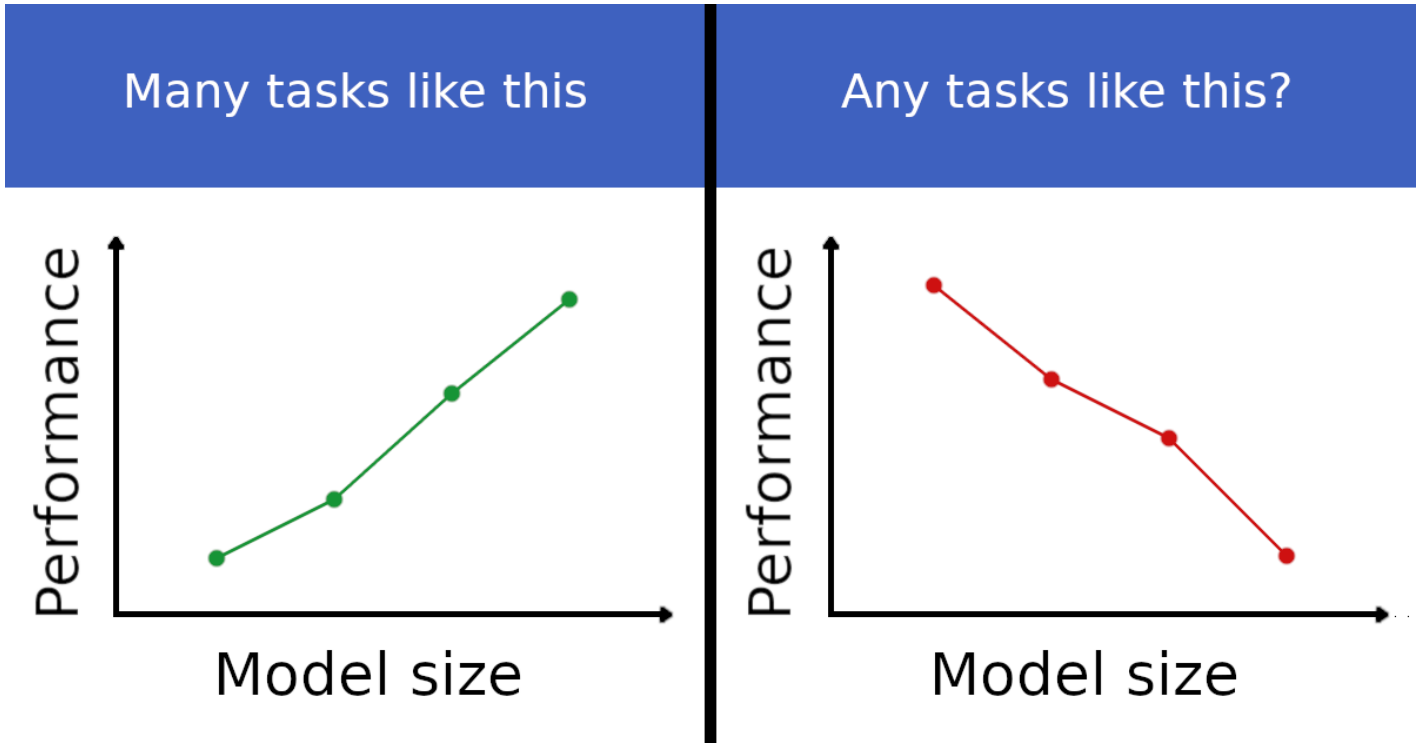
Scaling laws for large language models

- If we continue to decrease the test loss, LLMs show **emerging properties**
- LLMs are not directly trained to have these abilities, and they appear in **rapid** and **unpredictable** ways (e.g., arithmetic, summarization, question answering, understanding jokes, . . .)

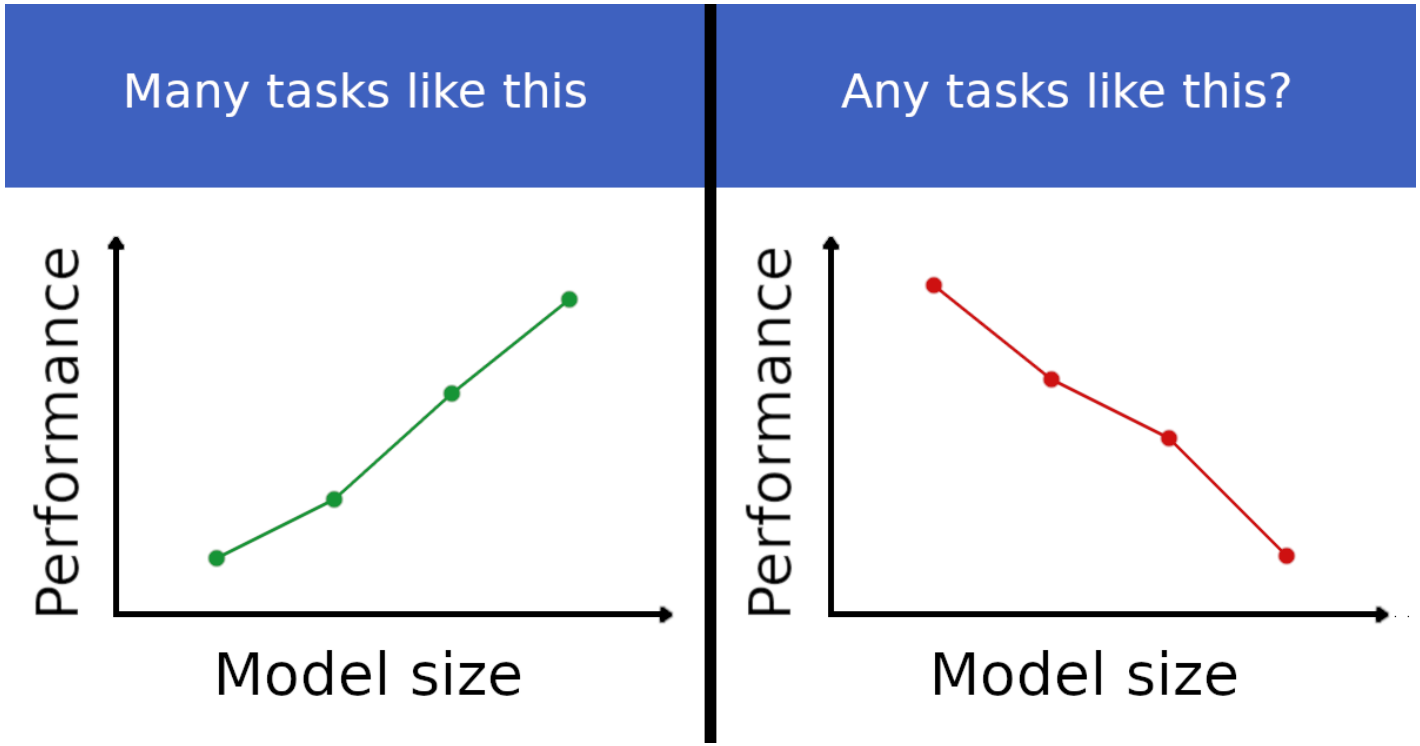
- The loss **scales** as a power law with model size, dataset size, and the amount of computing power used for training



Inverse scaling



Inverse scaling



- Negation question answering, modus tollens, quote repetition, . . .
- https://github.com/inverse-scaling/prize

In-context learning

Idea: we can unlock LLM's capabilities with clever prompts **without updating** any weights in our model

In-context learning

Idea: we can unlock LLM's capabilities with clever prompts **without updating** any weights in our model

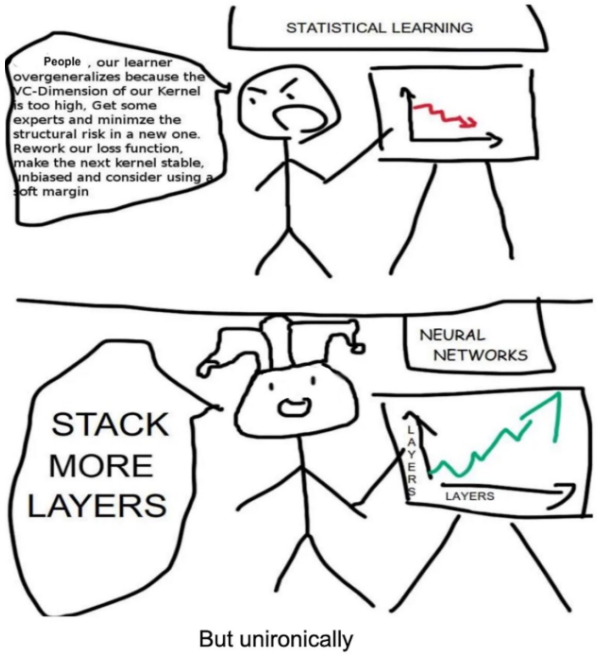
Examples:

1. **Demonstrations** – provide a few examples within the prompt for the task at hand
2. **Chain-of-thought** reasoning – *Think step-by-step...*
3. **Reflection** – self-improvement through reflection (*Did your answer meet all of the requirements?*)

Capabilities?

GPT-4 visual input example, Moar Layers:

User: Can you explain why this is funny. Think about it step-by-step.



GPT-4: The comic is satirizing the difference in approaches to improving model performance between statistical learning and neural networks.

In statistical learning, the character is shown to be concerned with overfitting and suggests a series of complex and technical solutions, such as minimizing structural risk, reworking the loss function, and using a soft margin.

In contrast, the neural networks character simply suggests adding more layers to the model. This is often seen as a common solution to improving performance in neural networks, but it's also considered a simplistic and brute-force approach.

The humor comes from the contrast between the complexity and specificity of the statistical learning approach and the simplicity and generality of the neural network approach. The "But unironically" comment adds to the humor by implying that, despite being simplistic, the "stack more layers" approach is often effective in practice.

The alignment problem

When developing powerful LLMs, we need to investigate them by evaluating their outputs, considering human values and interests.

Alignment:

- The **alignment** process is hard and difficult to control
- So far, research is mostly oriented on the *outer* alignment

Reinforcement learning from human feedback (RLHF)

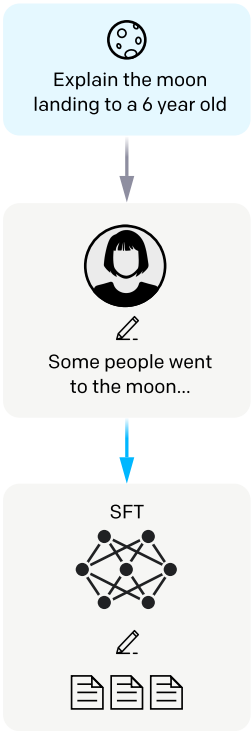
Step 1

Collect demonstration data, and train a supervised policy.

A prompt is sampled from our prompt dataset.

A labeler demonstrates the desired output behavior.

This data is used to fine-tune GPT-3 with supervised learning.



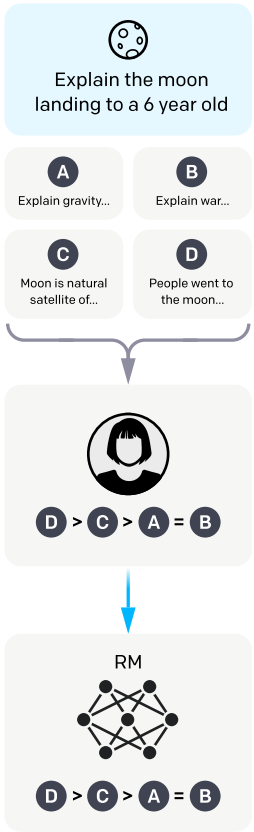
Step 2

Collect comparison data, and train a reward model.

A prompt and several model outputs are sampled.

A labeler ranks the outputs from best to worst.

This data is used to train our reward model.



Step 3

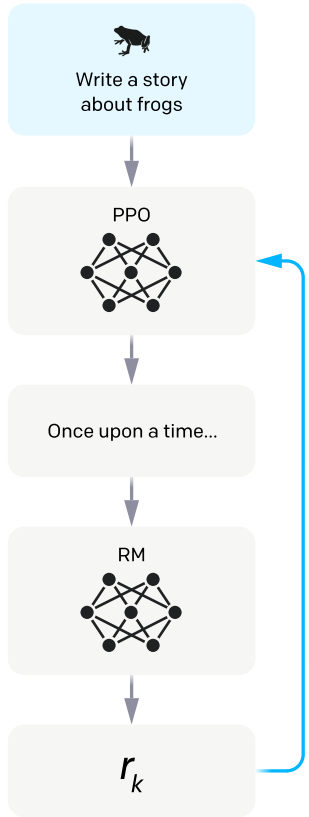
Optimize a policy against the reward model using reinforcement learning.

A new prompt is sampled from the dataset.

The policy generates an output.

The reward model calculates a reward for the output.

The reward is used to update the policy using PPO.



Challenges in alignment

Is RLHF the silver bullet?

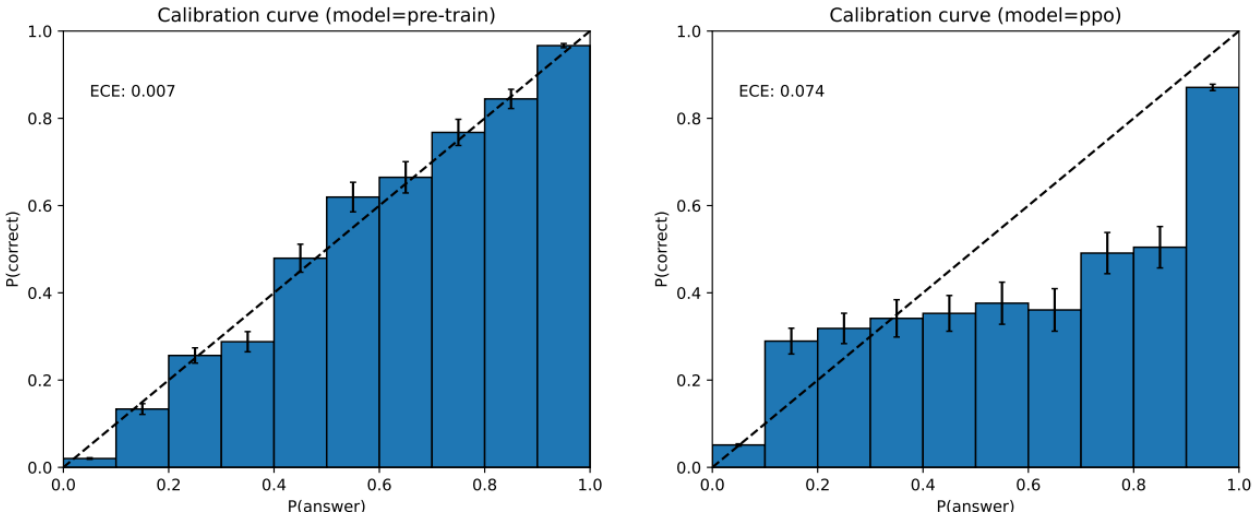
- No, it's far from perfect
- **Alignment tax**: aligning the models on certain tasks can make their performance worse on some other NLP tasks
- The models can still generate biased outputs, hallucinate, and generate inappropriate content without explicit prompting

Challenges in alignment

Is RLHF the silver bullet?

- No, it's far from perfect
- **Alignment tax**: aligning the models on certain tasks can make their performance worse on some other NLP tasks
- The models can still generate biased outputs, hallucinate, and generate inappropriate content without explicit prompting

Unexpected consequences can appear, e.g., RLHF degraded the calibration of GPT-4:



Learning outcomes 2

1. Explain how the Transformer decoder architecture is used for generative LLMs
2. Explain the scaling laws for LLMs
3. Explain the idea of in-context learning
4. Describe the alignment problem for LLMs and how one can mitigate it with RLHF

Outline

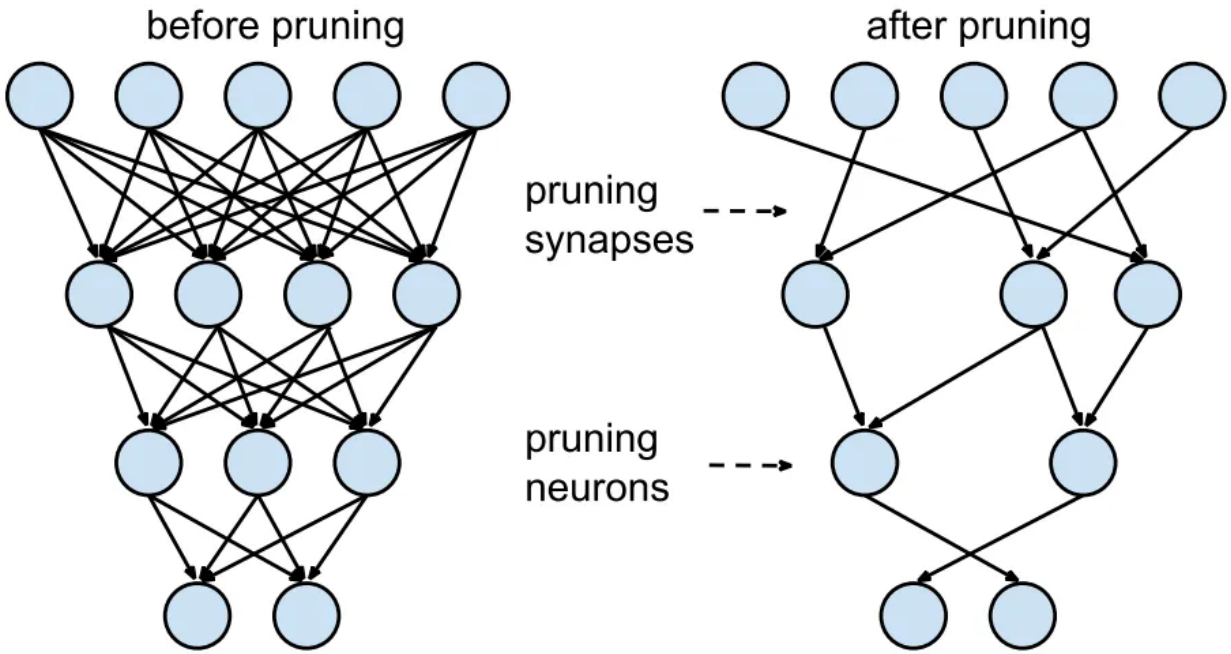
Modularity

- The models are increasing in size: typically, we fine-tune the entire language model for a specific task
- **Catastrophic interference**: neural networks have the tendency to drastically forget previously learned information upon learning new information
- Idea: we can use module composition to improve performance on multiple tasks
- Can we somehow **efficiently** update our models through added components?

Toward parameter-efficient learning

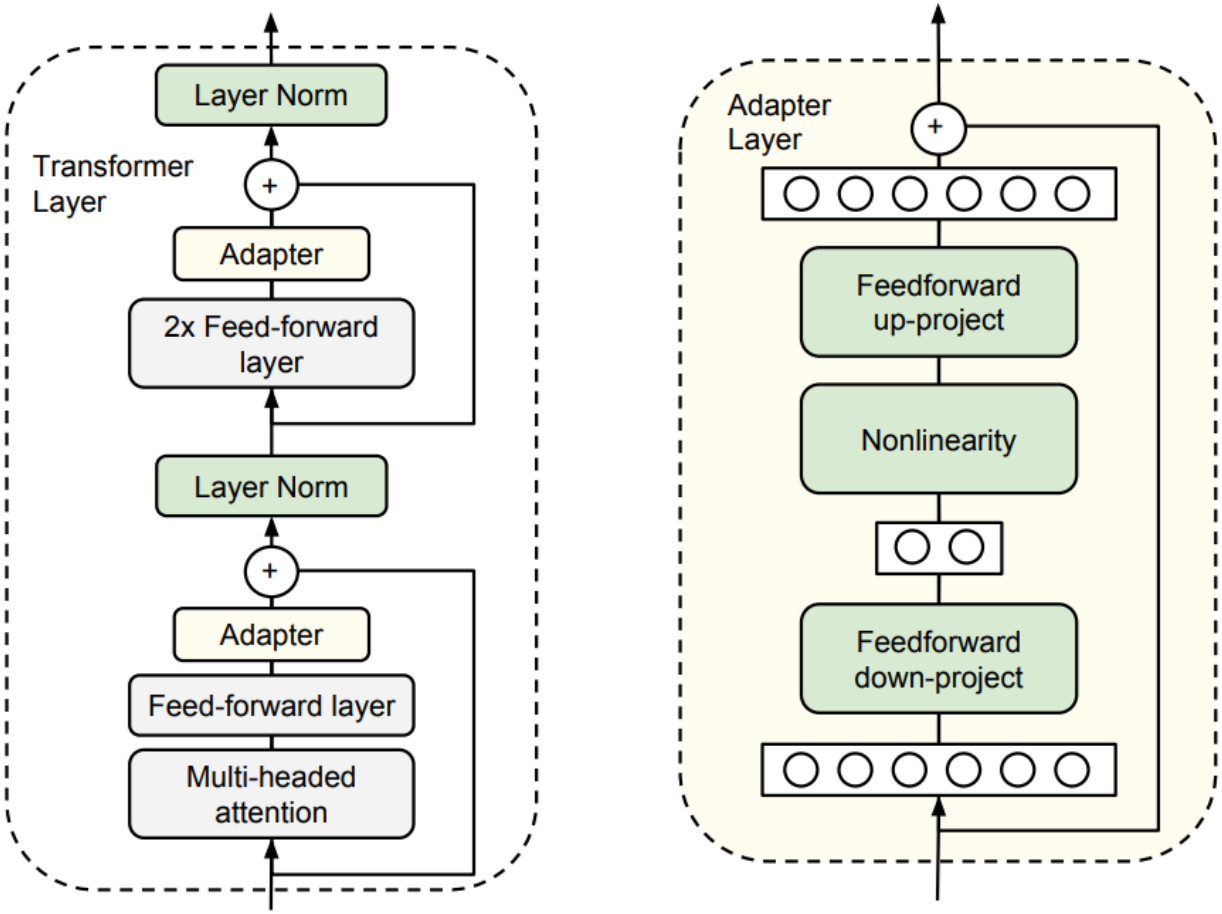
The lottery ticket hypothesis

Randomly-initialized, feed-forward networks contain subnetworks ("winning tickets") that perform comparably to the original network.

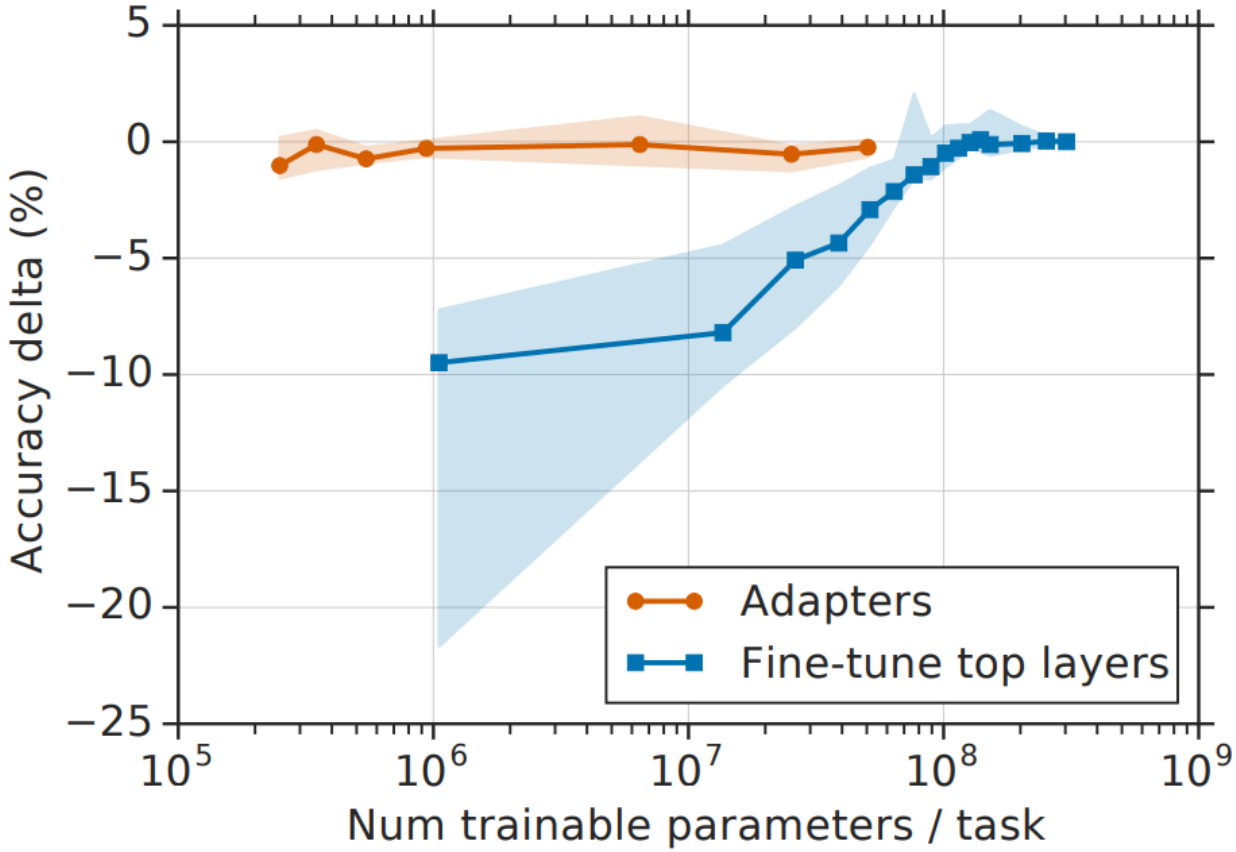


Adapters

- Integrate a module that contains **significantly fewer** parameters than the original Transformers
- **Freeze** the rest of the parameters



Efficiency of adapters



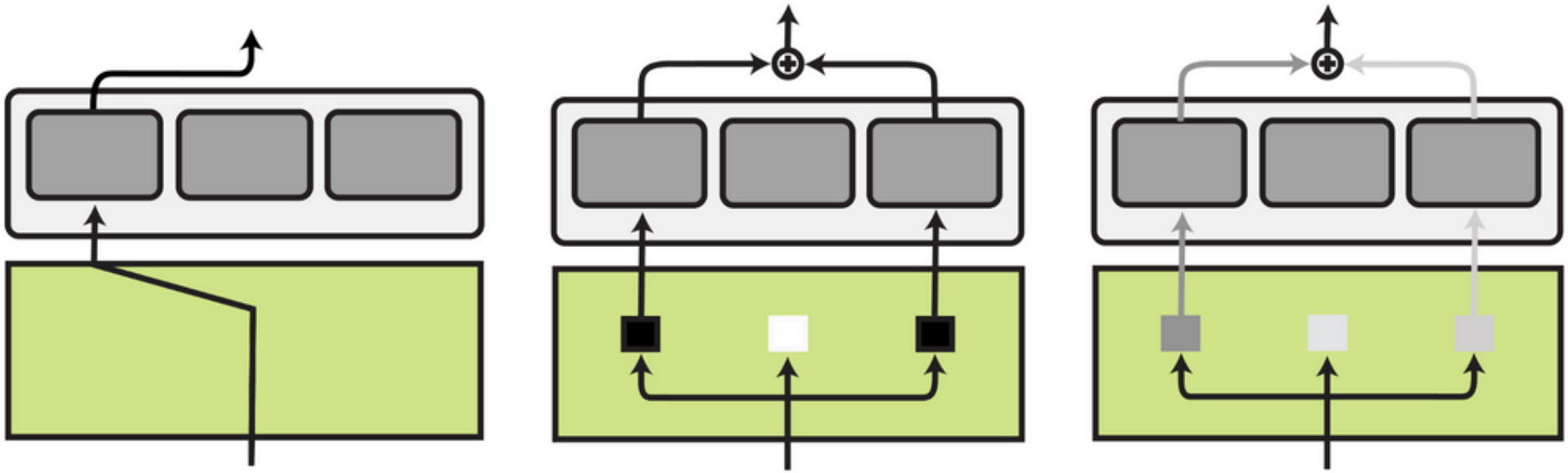
GLUE benchmark

Routing

- New adapter for each task
- How do we decide which one to use in a multi-task scenario?

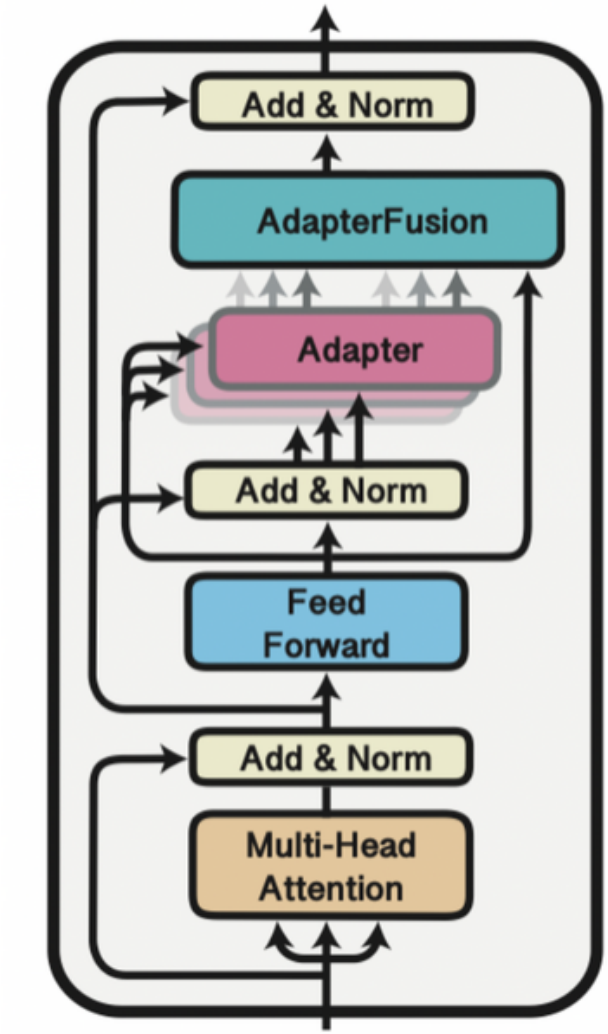
Routing

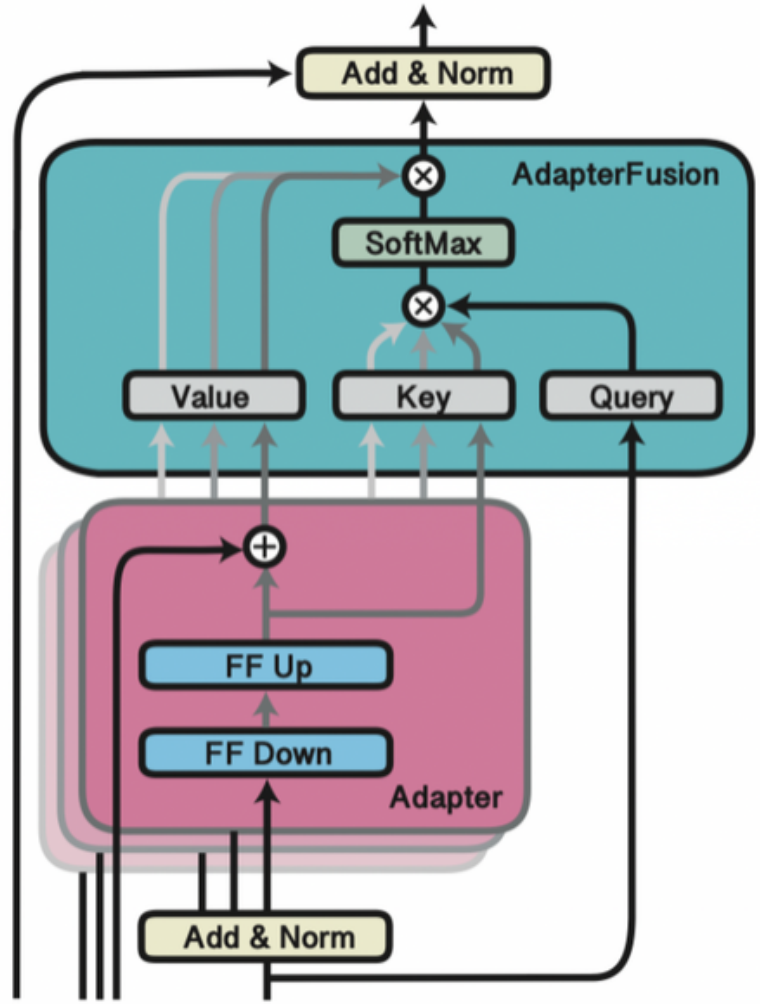
- New adapter for each task
- How do we decide which one to use in a multi-task scenario?



(a) Fixed Routing (b) Learned Routing (Hard) (c) Learned Routing (Soft)

Adapter fusion





Learning outcomes 3

1. Explain the motivation behind the modular approach to fine-tuning
2. Explain the idea behind adapters for parameter-efficient learning
3. Describe routing in modular learning and explain why we need it

Study assignment

1. Watch the second two parts of TAR "Neural NLP" video lectures
   - Video Part III
   - Video Part IV
2. Read the section on "Contextualized embeddings" (the third part of the reading material):
   - Reading
3. Read the OpenAI's blog on aligning language models:
   - https://openai.com/research/instruction-following
4. Read the article on Modular deep learning by Sebastian Ruder:
   - https://www.ruder.io/modular-deep-learning/
5. Self-check against learning outcomes!