BMI 702: Biomedical Informatics
(Large) Language Models

Jonathan Richard Schwarz jonathan_schwarz@hms.harvard.edu

Slides

03/21/2024

Outline for today

Part 1:

- N-Gram Language Models
- Transformers

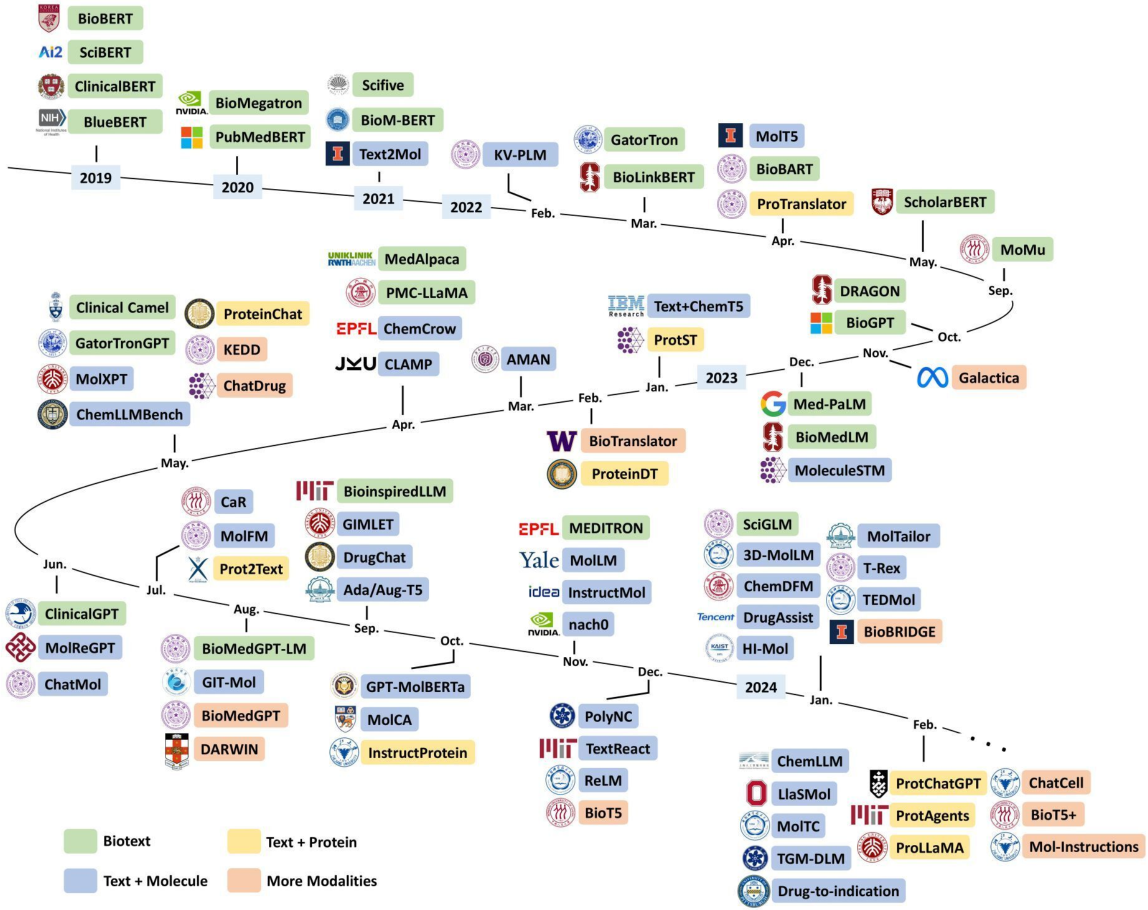
[break]

Part 2:

- In-Context Learning & Prompting
- Scaling Laws
- Parameter Efficient Fine-Tuning & Quantization
- Capabilities & Limitations

+ Glossary of new ideas (RLHF, RAG, Instruction Tuning), time permitting

This class



2019
BioBERT
SciBERT
ClinicalBERT
BlueBERT

2020
BioMegatron
PubMedBERT

2021
Scifive
BioM-BERT
Text2Mol

2022

Feb.
KV-PLM

Mar.
GatorTron
BioLinkBERT

Apr.
MolT5
BioBART
ProTranslator

May.
ScholarBERT

Sep.
MoMu

Oct.
DRAGON
BioGPT

Nov.
Galactica

Dec.
Med-PaLM
BioMedLM
MoleculeSTM

2023

Jan.
Text+ChemT5
ProtST

Feb.
BioTranslator
ProteinDT

Mar.
AMAN

Apr.
MedAlpaca
PMC-LLaMA
ChemCrow
CLAMP

May.
Clinical Camel
GatorTronGPT
MolXPT
ChemLLMBench
ProteinChat
KEDD
ChatDrug

Jun.
ClinicalGPT
MolReGPT
ChatMol

Jul.
CaR
MolFM
Prot2Text

Aug.
BioMedGPT-LM
GIT-Mol
BioMedGPT
DARWIN

Sep.
BioinspiredLLM
GIMLET
DrugChat
Ada/Aug-T5

Oct.
GPT-MolBERTa
MolCA
InstructProtein

Nov.
MEDITRON
MolLM
InstructMol
nach0

Dec.
PolyNC
TextReact
ReLM
BioT5

2024

Jan.
SciGLM
3D-MolLM
ChemDFM
DrugAssist
HI-Mol
MolTailor
T-Rex
TEDMol
BioBRIDGE
ChemLLM
LlaSMol
MolTC
TGM-DLM
Drug-to-indication

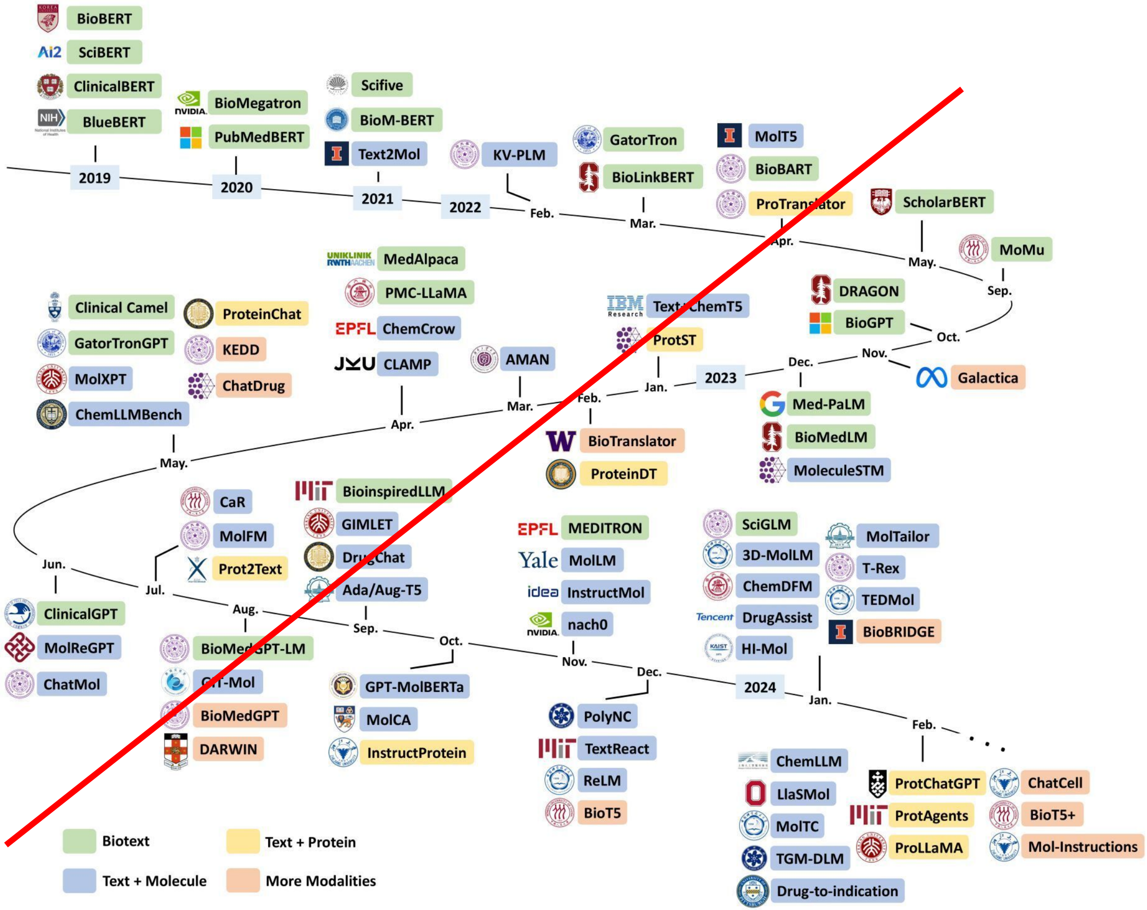
Feb.
ProtChatGPT
ProtAgents
ProLLaMA
ChatCell
BioT5+
Mol-Instructions

. . .

🟩 Biotext
🟨 Text + Protein
🟦 Text + Molecule
🟧 More Modalities

[Leveraging Biomolecule and Natural Language through Multi-Modal Learning: A Survey]

This class



[Leveraging Biomolecule and Natural Language through Multi-Modal Learning: A Survey]

Outline for today

Part 1:

- **N-Gram Language Models**
- Transformers

[break]

Part 2:

- In-Context Learning & Prompting
- Scaling Laws
- Parameter Efficient Fine-Tuning & Quantization
- Capabilities & Limitations

A famous quote

It must be recognized that the notion "probability of a sentence" is an entirely useless one, under any known interpretation of this term.

- Noam Chomsky, 1969

[Based on Edoardo Ponti's slides]

Intuitive interpretation

"Probability of a sentence" = how likely is it to occur in natural language

Example 1: Grammatical knowledge

$$p(the\, cat\, purrs) > p(cat\, purrs\, the)$$

Example 2: World knowledge

$$p(the\, cat\, purrs) > p(the\, cat\, smokes)$$

[Based on Edoardo Ponti's slides]

Intuitive interpretation

“Probability of a sentence” = how likely is it to occur in natural language

Example 1: Grammatical knowledge

$$p(the\, cat\, purrs) > p(cat\, purrs\, the)$$

Example 2: World knowledge

$$p(the\, cat\, purrs) > p(the\, cat\, smokes)$$

What about the probability of `"the Archaeopteryx winged jaggedly amidst foliage"`?

[Based on Edoardo Ponti's slides]

Intuitive interpretation

“Probability of a sentence” = how likely is it to occur in natural language

Example 1: Grammatical knowledge

$$p(the\,cat\,purrs) > p(cat\,purrs\,the)$$

Example 2: World knowledge

$$p(the\,cat\,purrs) > p(the\,cat\,smokes)$$

What about the probability of `"the Archaeopteryx winged jaggedly amidst foliage"`?

→ Useless measure to decide whether a sentence is grammatical

[Based on Edoardo Ponti's slides]

Probabilistic Models of Language

→ A vocabulary Σ is a (finite, non-empty) set of symbols (result of tokenization).

[Based on Edoardo Ponti's slides]

Probabilistic Models of Language

→ A vocabulary Σ is a (finite, non-empty) set of symbols (result of tokenization).

→ Kleene closure $\Sigma^\star$ of a vocabulary: Set of all possible (finite-length) sequences including the empty sequence.

→ Language $L \subseteq \Sigma^\star$: Subset of the Kleene closure.

[Based on Edoardo Ponti's slides]

Probabilistic Models of Language

→ A vocabulary Σ is a (finite, non-empty) set of symbols (result of tokenization).

→ Kleene closure $\Sigma^\star$ of a vocabulary: Set of all possible (finite-length) sequences including the empty sequence.

→ Language $L \subseteq \Sigma^\star$: Subset of the Kleene closure.

Example: $\Sigma = \{0, 1\}$ then $\Sigma^* = \{\epsilon, 0, 1, 00, 01, 10, 11, 000, \ldots\}$

[Based on Edoardo Ponti's slides]

Probabilistic Models of Language

→ A vocabulary Σ is a (finite, non-empty) set of symbols (result of tokenization).

→ Kleene closure $\Sigma^\star$ of a vocabulary: Set of all possible (finite-length) sequences including the empty sequence.

→ Language $L \subseteq \Sigma^\star$: Subset of the Kleene closure.

Probability model:

1. $p(L) = 1$
2. $p(\bigcup_{i=1}^{n} \mathcal{E}_i) = \sum_i^n p(\mathcal{E}_i)$ if $\mathcal{E}_1, \mathcal{E}_2, \ldots$ is a countable sequence of disjoint sets of $\mathcal{P}(L)$, the power set (=set of all subsets) of L.

[Based on Edoardo Ponti's slides]

Probabilistic Models of Language

→ A vocabulary $\sum$ is a (finite, non-empty) set of symbols (result of tokenization).

→ Kleene closure $\sum^\star$ of a vocabulary: Set of all possible (finite-length) sequences including the empty sequence.

→ Language $L \subseteq \Sigma^\star$: Subset of the Kleene closure.

Probability model:

1. $p(L) = 1$
2. $p(\bigcup_{i=1}^{n} \mathcal{E}_i) = \sum_i^n p(\mathcal{E}_i)$ if $\mathcal{E}_1, \mathcal{E}_2, \ldots$ is a countable sequence of disjoint sets of $\mathcal{P}(L)$, the power set (=set of all subsets) of L.
3. (Conditional probability) $$p(\mathbf{x}) = p(x_0) \prod_{i=1}^{L} p(x_i | x_1, \ldots, x_{i-1})$$

[Based on Edoardo Ponti's slides]

Probabilistic Models of Language

→ A vocabulary Σ is a (finite, non-empty) set of symbols (result of tokenization).

→ Kleene closure $\Sigma^\star$ of a vocabulary: Set of all possible (finite-length) sequences including the empty sequence.

→ Language $L \subseteq \Sigma^\star$: Subset of the Kleene closure.

Probability model:

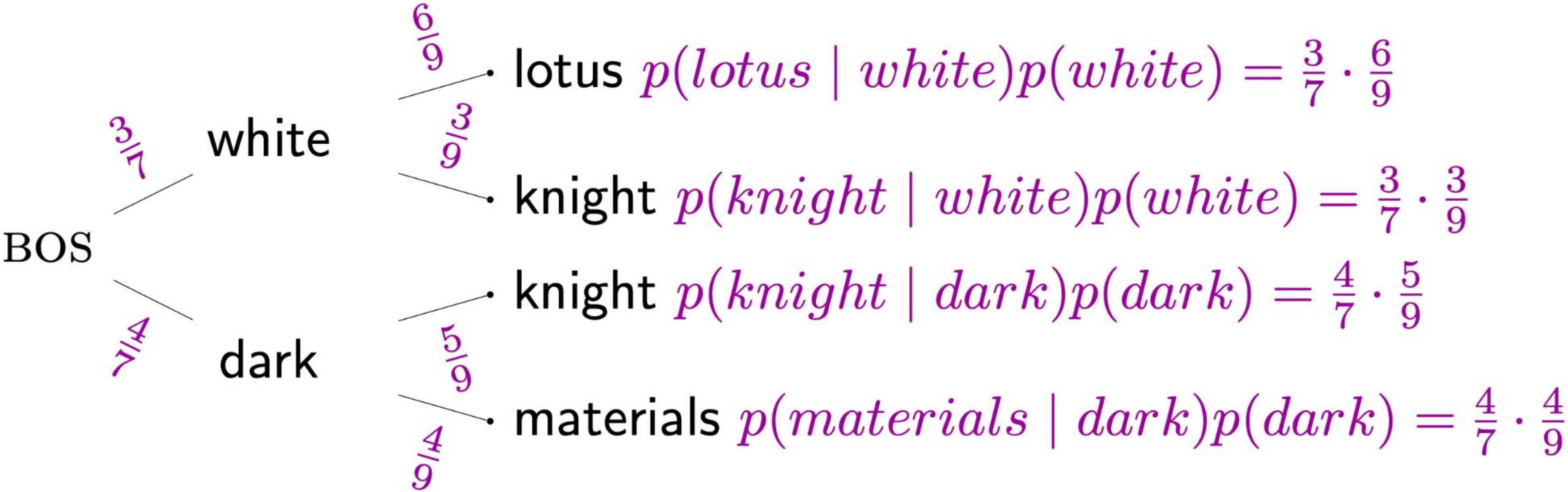
1. $p(L) = 1$

2. $p(\bigcup_{i=1}^{n} \mathcal{E}_i) = \sum_i^n p(\mathcal{E}_i)$ if $\mathcal{E}_1, \mathcal{E}_2, \ldots$ is a countable sequence of disjoint sets of $\mathcal{P}(L)$, the power set (=set of all subsets) of L.

3. (Conditional probability) $\log p(\mathbf{x}) = \log p(x_0) \sum_{i=1}^{L} \log p(x_i | x_1, \ldots, x_{i-1})$

[Based on Edoardo Ponti's slides]

Example



BOS

$\frac{3}{7}$ white

$\frac{4}{7}$ dark

$\frac{6}{9}$ lotus $p(lotus \mid white)p(white) = \frac{3}{7} \cdot \frac{6}{9}$

$\frac{3}{9}$ knight $p(knight \mid white)p(white) = \frac{3}{7} \cdot \frac{3}{9}$

$\frac{5}{9}$ knight $p(knight \mid dark)p(dark) = \frac{4}{7} \cdot \frac{5}{9}$

$\frac{4}{9}$ materials $p(materials \mid dark)p(dark) = \frac{4}{7} \cdot \frac{4}{9}$

[Based on Edoardo Ponti's slides]

Estimation

We assume there is some true $p^\star$ which we estimate/approximate with a (parametric) estimator) $\hat{p}$ which is an element of $\{p_\theta \mid \theta \in \Theta\}$.

[Based on Edoardo Ponti's slides]

Estimation

We assume there is some true $p^\star$ which we estimate/approximate with a (parametric) estimator) $\hat{p}$ which is an element of $\{p_\theta \mid \theta \in \Theta\}$.

This is done by learning from data $\mathcal{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \subseteq L$, e.g. by minimizing some loss:

$$\hat{\theta} \triangleq \arg\min_{\theta \in \Theta} \ell(\theta, \theta^\star)$$

[Based on Edoardo Ponti's slides]

Estimation

We assume there is some true $p^\star$ which we estimate/approximate with a (parametric) estimator) $\hat{p}$ which is an element of $\{p_\theta \mid \theta \in \Theta\}$.

This is done by learning from data $\mathcal{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \subseteq L$, e.g. by minimizing some loss:

$$\hat{\theta} \triangleq \arg\min_{\theta\in\Theta} \ell(\theta, \theta^\star)$$

Since the optimal model is unknown, we use the data as an estimate:

$$p_{\theta^\star} \approx \frac{1}{|\mathcal{D}|}\sum_{i=1}^{\mathcal{D}} \delta_{\mathbf{x}_i}(\mathbf{x}) \qquad \delta_{\mathbf{x}_i}(\mathbf{x}) \triangleq \begin{cases} 1 & \text{if } \mathbf{x}_i = \mathbf{x} \\ 0 & \text{else} \end{cases}$$

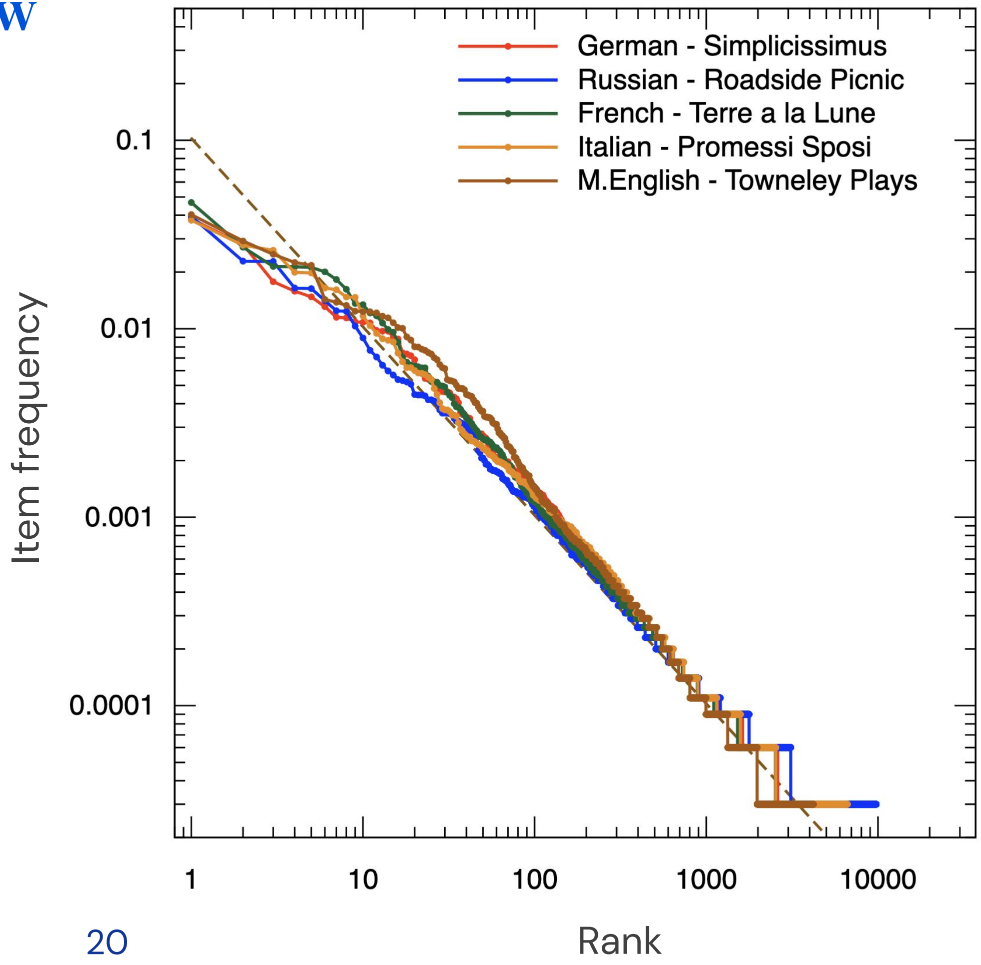
[Based on Edoardo Ponti's slides]

A note about data - Zipf's Law

Word frequency approximately inversely proportional to its rank:

$$\text{frequency} \propto \frac{1}{(\text{rank} + b)^a}$$

with **a, b** fitted. (Zipf-Mandelbrot law)

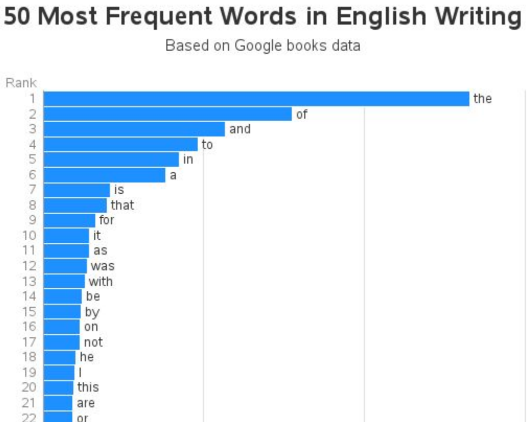


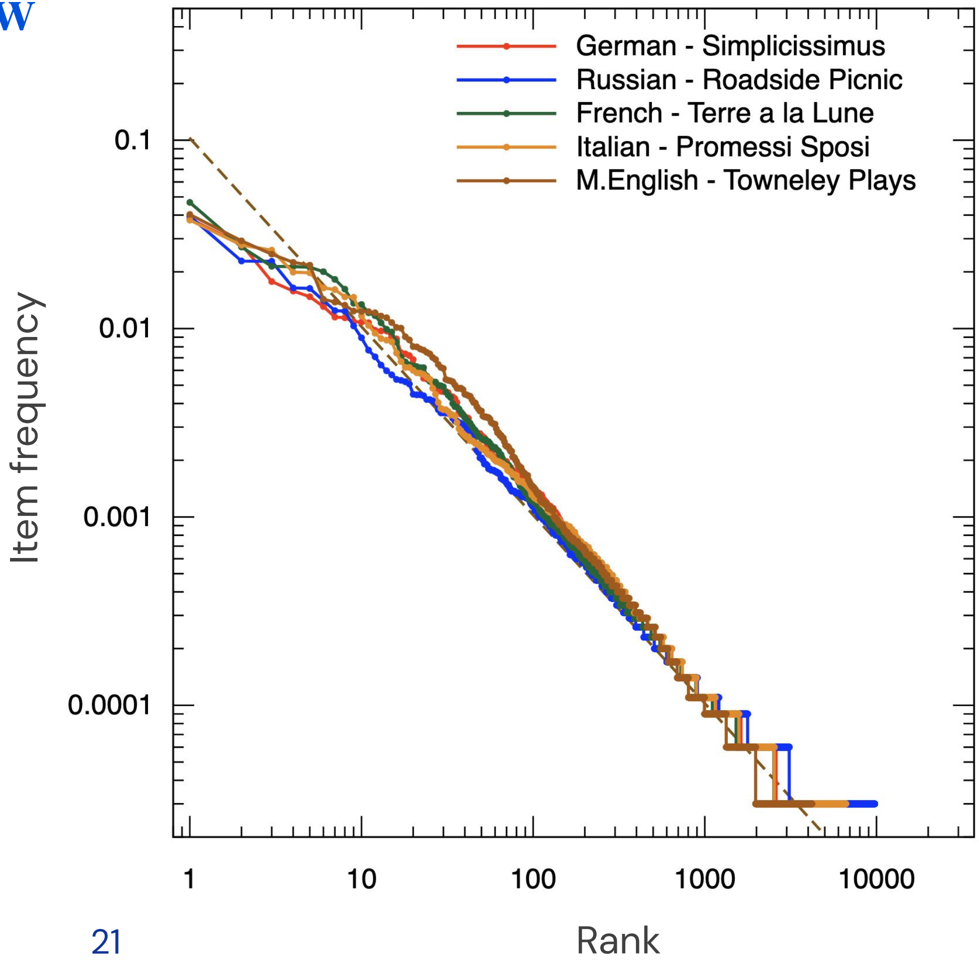
A note about data - Zipf's Law

Word frequency approximately inversely proportional to its rank:

$$\text{frequency} \propto \frac{1}{(\text{rank} + b)^a}$$

with **a, b** fitted. (Zipf-Mandelbrot law)





Cross-Entropy

A suitable loss function is the KL-Divergence (divergence between prob. distributions):

$$\mathbb{KL}(p_{\theta^\star}, p_{\hat{\theta}}) \triangleq -\sum_{\mathbf{x}\in L} p_{\theta^\star}(\mathbf{x}) \log p_{\hat{\theta}}(\mathbf{x}) + \underbrace{p_{\theta^\star}(\mathbf{x}) \log p_{\theta^\star}(\mathbf{x})}_{-H(p_{\theta^\star})}$$

[Based on Edoardo Ponti's slides]

Cross-Entropy

A suitable loss function is the KL-Divergence (divergence between prob. distributions):

$$\mathbb{KL}(p_{\theta^\star}, p_{\hat{\theta}}) \triangleq \underset{\text{Cross-Entropy}}{\underline{-\sum_{\mathbf{x}\in L} p_{\theta^\star}(\mathbf{x}) \log p_{\hat{\theta}}(\mathbf{x})}} + \underset{\text{constant wrt. model param.}}{\cancel{\underbrace{p_{\theta^\star}(\mathbf{x}) \log p_{\theta^\star}(\mathbf{x})}_{-H(p_{\theta^\star})}}}$$

Justification:

From Information Theory: Measures the excess number of bits we pay by encoding our data with a sub-optimal model. The optimum is just the entropy (Shannon, 1948).

[Based on Edoardo Ponti's slides]

N-Gram models

We can obtain a very simple form for $\{p_\theta \mid \theta \in \Theta\}$ by making the Markov assumption:

$$\begin{aligned} p(\mathbf{x}) &= p(x_1, \ldots, x_n) \\ &= p(x_n | x_1, x_2, \ldots, x_{n-1}) p(x_{n-1} | x_1, x_2, \ldots, x_{n-2}) \ldots p(x_1) \\ &\approx p(x_n | x_{n-2}, x_{n-1}) p(x_{n-1} | x_{n-3}, x_{n-2}) \ldots p(x_1) \end{aligned}$$

[Based on Edoardo Ponti's slides]

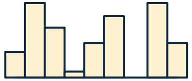
N-Gram models

We can obtain a very simple form for $\{p_\theta \mid \theta \in \Theta\}$ by making the Markov assumption:

$$\begin{aligned} p(\mathbf{x}) &= p(x_1, \ldots, x_n) \\ &= p(x_n | x_1, x_2, \ldots, x_{n-1}) p(x_{n-1} | x_1, x_2, \ldots, x_{n-2}) \ldots p(x_1) \\ &\approx p(x_n | x_{n-2}, x_{n-1}) p(x_{n-1} | x_{n-3}, x_{n-2}) \ldots p(x_1) \end{aligned}$$

This is a tri-gram model (history of two). Straightforwardly estimated using the Maximum-Likelihood Estimate of a categorical distribution:

$$p_{\mathrm{ML}}(x_3 | x_1, x_2) = \frac{C(x_1, x_2, x_3)}{C(x_1, x_2)}$$



What's the problem with this model?

[Based on Edoardo Ponti's slides]

N-Gram models

We can obtain a very simple form for $\{p_\theta \mid \theta \in \Theta\}$ by making the Markov assumption:

$$\begin{aligned} p(\mathbf{x}) &= p(x_1, \dots, x_n) \\ &= p(x_n | x_1, x_2, \dots, x_{n-1}) p(x_{n-1} | x_1, x_2, \dots, x_{n-2}) \dots p(x_1) \\ &\approx p(x_n | x_{n-2}, x_{n-1}) p(x_{n-1} | x_{n-3}, x_{n-2}) \dots p(x_1) \end{aligned}$$

This is a tri-gram model (history of two). Assumes all of these are equal:

- $p(\text{slept}|\text{the cat})$
- $p(\text{slept}|\text{after lunch the cat})$
- $p(\text{slept}|\text{the dog chased the cat})$
- $p(\text{slept}|\text{except for the cat})$

[Based on Edoardo Ponti's slides]

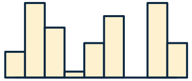
N-Gram models

We can obtain a very simple form for $\{p_\theta \mid \theta \in \Theta\}$ by making the Markov assumption:

$$\begin{aligned} p(\mathbf{x}) &= p(x_1, \ldots, x_n) \\ &= p(x_n | x_1, x_2, \ldots, x_{n-1}) p(x_{n-1} | x_1, x_2, \ldots, x_{n-2}) \ldots p(x_1) \\ &\approx p(x_n | x_{n-2}, x_{n-1}) p(x_{n-1} | x_{n-3}, x_{n-2}) \ldots p(x_1) \end{aligned}$$

This is a tri-gram model (history of two). Straightforwardly estimated using the Maximum-Likelihood Estimate of a categorical distribution:

$$p_{\mathrm{ML}}(x_3 | x_1, x_2) = \frac{C(x_1, x_2, x_3)}{C(x_1, x_2)}$$



Zero-Probability events!

[Based on Edoardo Ponti's slides]

Bayesian N-Gram models

Likelihood (Categorical)

Prior (Dirichlet)

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})}$$

Evidence

[Based on Edoardo Ponti's slides]

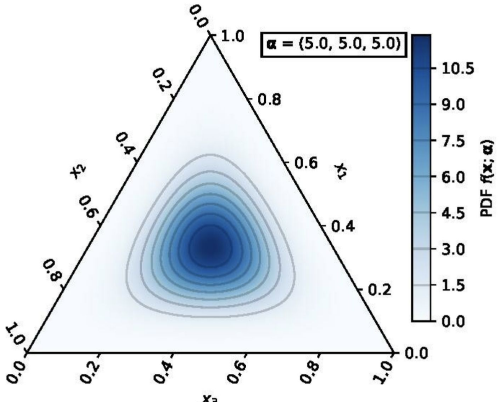
Bayesian N-Gram models

Likelihood (Categorical)

Prior (Dirichlet)

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})}$$

Evidence



Turns out this is an example of a "conjugate prior". A choice of prior for which the posterior has the same shape as the prior.

$$\begin{aligned} p_1, \dots, p_K &\sim \mathrm{Dir}(\alpha_1, \dots, \alpha_K) \\ y &\sim \mathrm{Cat}(p_1, \dots, p_K) \end{aligned}$$

[Based on Edoardo Ponti's slides]

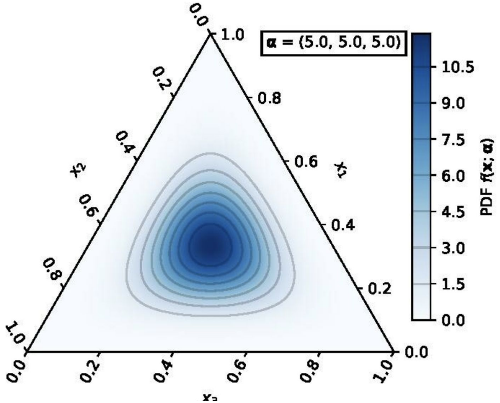
Bayesian N-Gram models

Likelihood (Categorical)

Prior (Dirichlet)

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})}$$

Evidence



Turns out this is an example of a "conjugate prior". A choice of prior for which the posterior has the same shape as the prior.

$$p_1, \ldots, p_K \sim \mathrm{Dir}(\alpha_1, \ldots, \alpha_K)$$

$$y \sim \mathrm{Cat}(p_1, \ldots, p_K)$$

$$p(\theta|\mathcal{D}) = \mathrm{Dir}(\alpha'_1, \ldots, \alpha'_K)$$

$$\alpha'_j = \boxed{\alpha_j} + \sum_{y_i \in D} \mathbb{1}\{y_i = j\}$$

You can think of those as "pseudo counts"

[Based on Edoardo Ponti's slides]

Evaluation

Two popular evaluation metrics evaluated on a held-out/test set:

(1) Cross entropy (per word):

$$H(p_{\theta^*}, p_{\hat{\theta}}) = -\frac{1}{|\mathcal{D}|} \sum_{\mathbf{x} \in \mathcal{D}} \frac{1}{n} \log_2 p_{\hat{\theta}}(x_1, \ldots, x_n)$$

Evaluation

Two popular evaluation metrics evaluated on a held-out/test set:

(1) Cross entropy (per word):

$$H(p_{\theta^*}, p_{\hat{\theta}}) = -\frac{1}{|\mathcal{D}|} \sum_{\mathbf{x} \in \mathcal{D}} \frac{1}{n} \log_2 p_{\hat{\theta}}(x_1, \ldots, x_n)$$

(2) Perplexity (captures a notion of surprise):

$$PPL(p_{\hat{\theta}}) = 2^{H(p_{\theta^*}, p_{\hat{\theta}})}$$

Evaluation

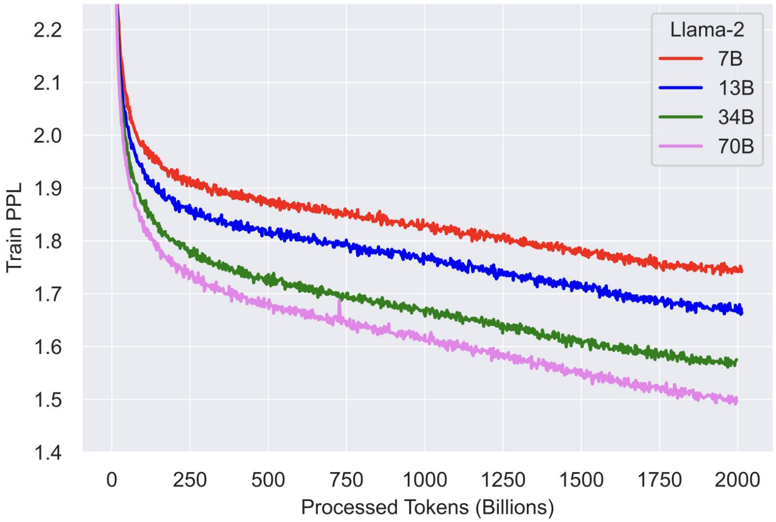
Two popular evaluation metrics evaluated on a held-out/test set:

(1) Cross entropy (per word):

$$H(p_{\theta^*}, p_{\hat{\theta}}) = -\frac{1}{|\mathcal{D}|} \sum_{\mathbf{x} \in \mathcal{D}} \frac{1}{n} \log_2 p_{\hat{\theta}}(x_1, \ldots, x_n)$$

(2) Perplexity (captures a notion of surprise):

$$PPL(p_{\hat{\theta}}) = 2^{H(p_{\theta^*}, p_{\hat{\theta}})}$$



[Touvron et al. "Llama 2: Open Foundation and Fine-Tuned Chat Models"]

Outline for today

Part 1:

- N-Gram Language Models
- **Transformers**

[break]

Part 2:

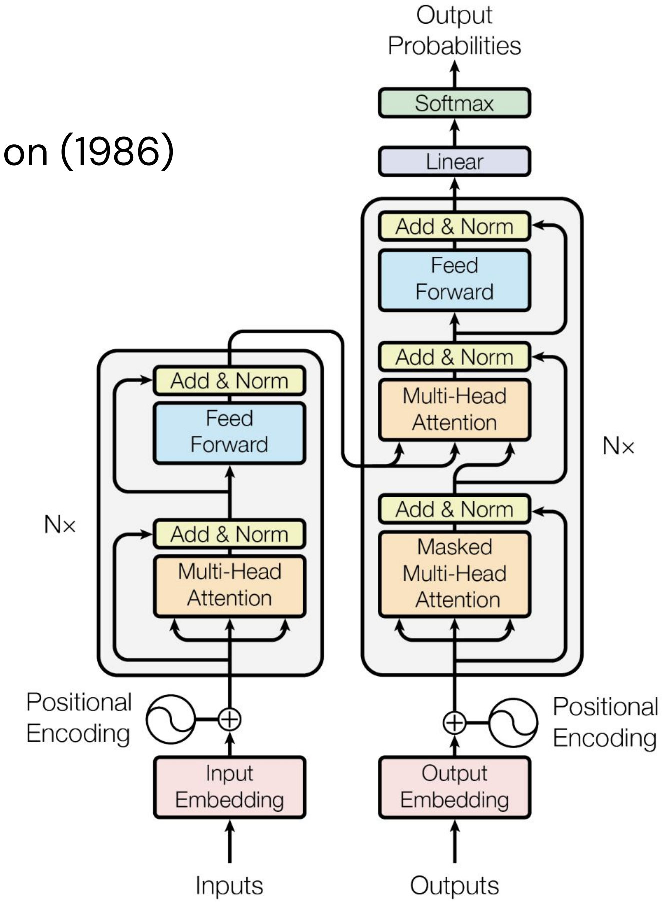
- In-Context Learning & Prompting
- Scaling Laws
- Parameter Efficient Fine-Tuning & Quantization
- Capabilities & Limitations

(Encoder-Decoder) Transformers

Probably the most influential ML paper since Backpropagation (1986)

→ Over 112k citations since 2017

→ Essentially replaced RNNs for most purposes



[Vaswani et al. "Attention is all you need", 2017]

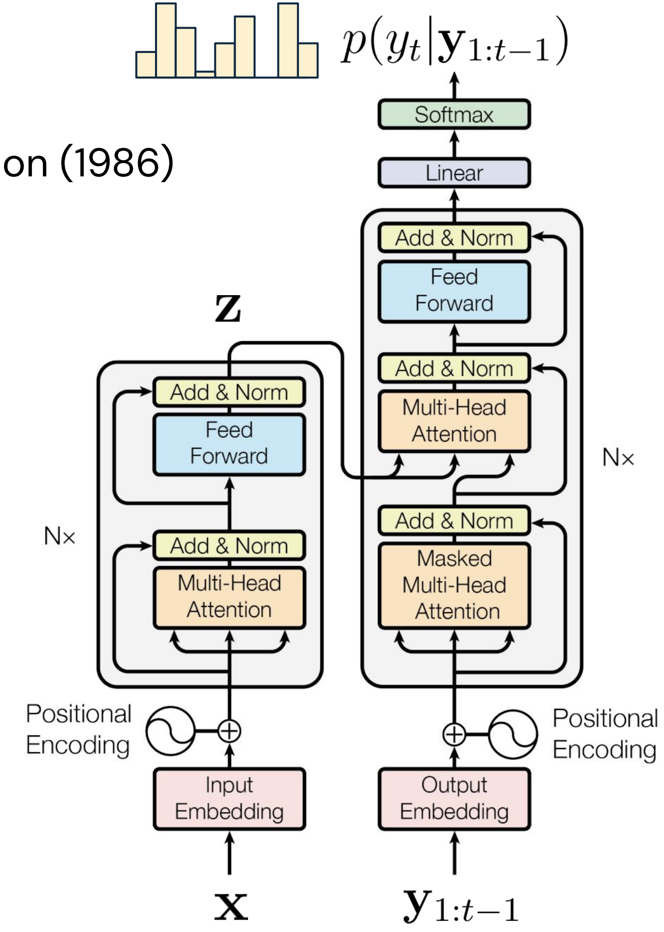
(Encoder-Decoder) Transformers

Probably the most influential ML paper since Backpropagation (1986)

→ Over 112k citations since 2017

→ Essentially replaced RNNs for most purposes

A simple sequence to sequence model mapping an input $(x_1, \ldots, x_n)$ (tokenized and "embedded") into a continuous representation $\mathbf{z} = (z_1, \ldots, z_n)$ based on which the decoder produces $(y_1, \ldots, y_m)$ autoregressively, i.e. one symbol at a time.



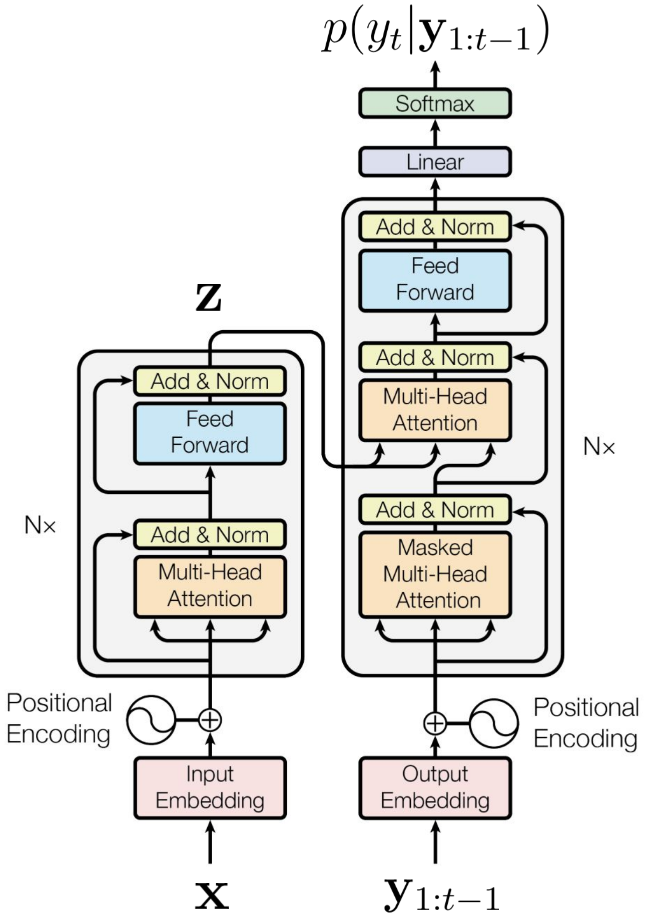
[Vaswani et al. "Attention is all you need", 2017]

The Transformer Building Blocks

1. Multi-head Attention

2. Position Encodings

3. Residual connections + Normalization



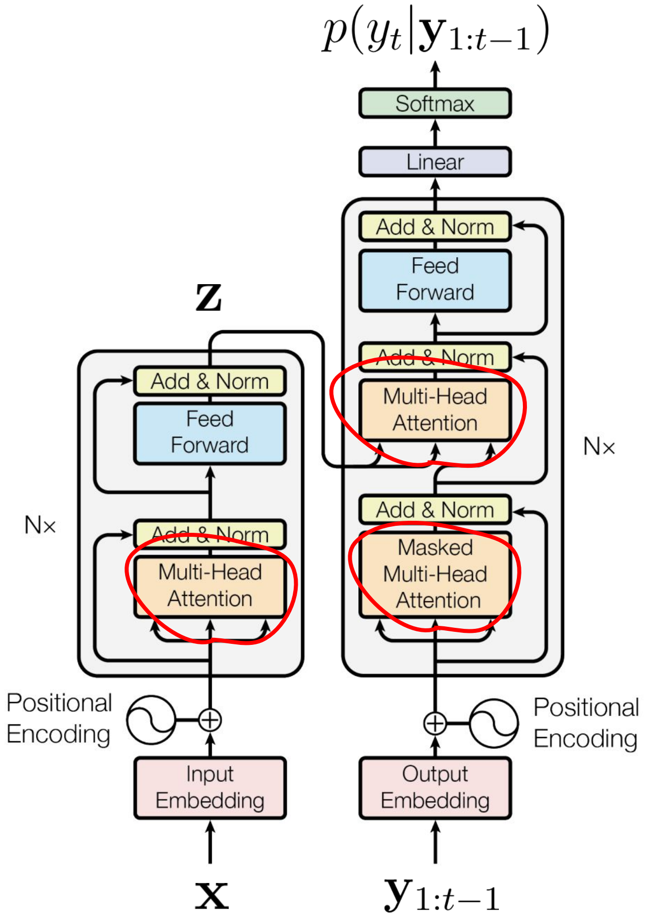
[Vaswani et al. "Attention is all you need", 2017]

The Transformer Building Blocks

1. **Multi-head Attention**

2. **Position Encodings**

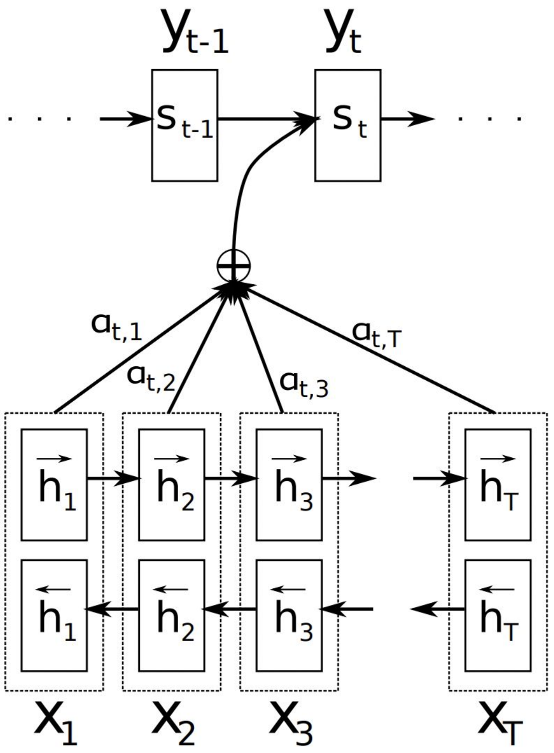
3. **Residual connections + Normalization**



[Vaswani et al. "Attention is all you need", 2017]

Attention: An idea from Machine Translation

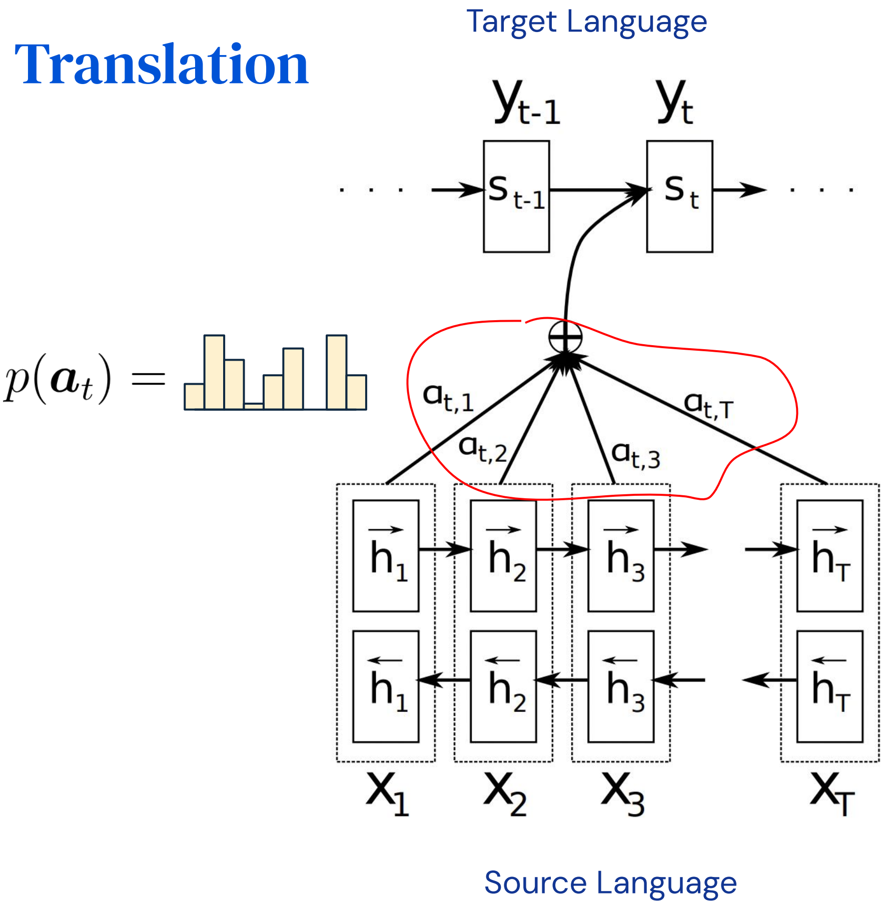
Target Language



Source Language

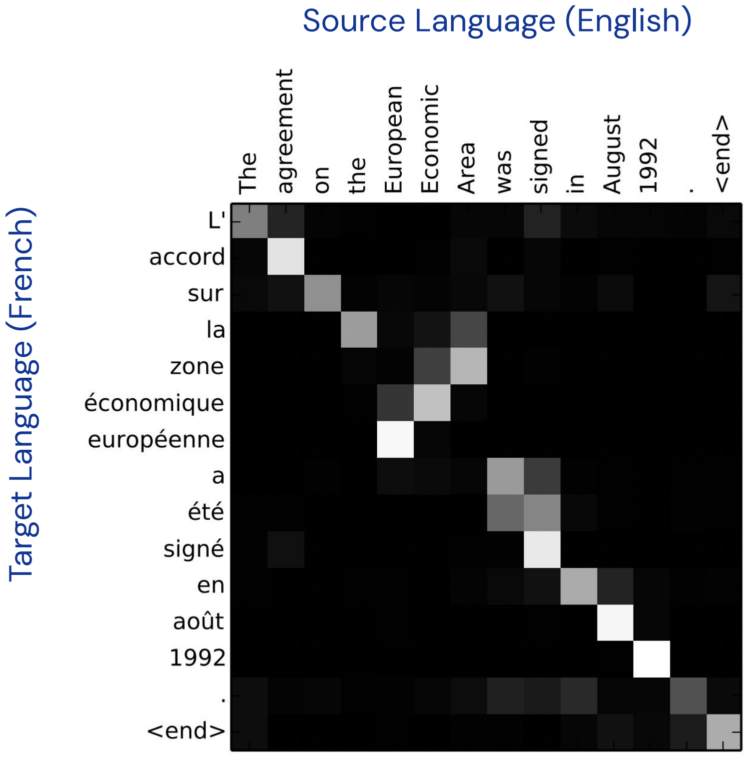
[Bahdanau et al. "Neural machine translation by jointly learning to align and translate", 2014]

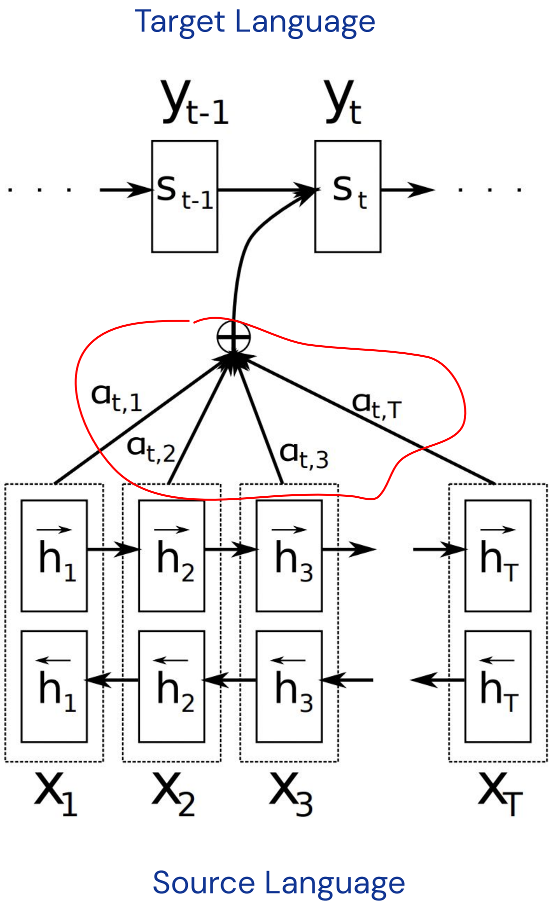
Attention: An idea from Machine Translation



[Bahdanau et al. "Neural machine translation by jointly learning to align and translate", 2014]

Attention: An idea from Machine Translation





[Bahdanau et al. "Neural machine translation by jointly learning to align and translate", 2014]

Scaled Dot-Product Attention

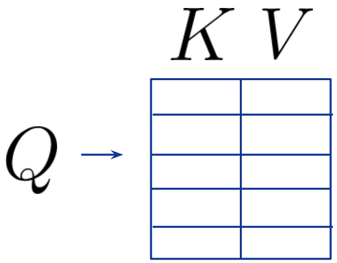
$$H = \mathrm{Attention}(QW^Q, KW^K, VW^V)$$

$$W^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}, W^K \in \mathbb{R}^{d_{\text{model}} \times d_k}, W^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$$

[Vaswani et al. "Attention is all you need", 2017]

Scaled Dot-Product Attention

$$H = \boxed{\text{Attention}}(QW^Q, KW^K, VW^V)$$



Think of this as a soft "look-up" operation in an associative memory using dot-products as a similarity measure.

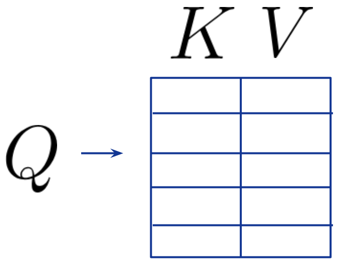
$$W^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}, W^K \in \mathbb{R}^{d_{\text{model}} \times d_k}, W^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$$

[Vaswani et al. "Attention is all you need", 2017]

Scaled Dot-Product Attention

$$H = \text{Attention}(QW^Q, KW^K, VW^V)$$

Where do they come from?

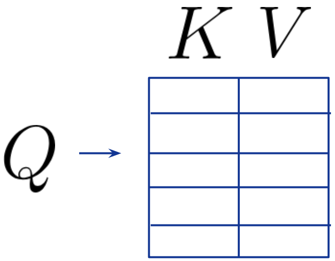


$$W^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}, W^K \in \mathbb{R}^{d_{\text{model}} \times d_k}, W^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$$

[Vaswani et al. "Attention is all you need", 2017]

Scaled Dot-Product Attention

$$H = \text{Attention}(QW^Q, KW^K, VW^V)$$



Where do they come from?

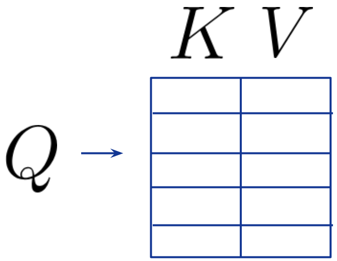
In Machine Translation, Keys and Values come from the source language, queries from the target language processed so far

$$W^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}, W^K \in \mathbb{R}^{d_{\text{model}} \times d_k}, W^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$$

[Vaswani et al. "Attention is all you need", 2017]

Scaled Dot-Product Attention

$$H = \text{Attention}(QW^Q, KW^K, VW^V)$$



Where do they come from?

For now, let's think of them as equal, i.e. the input (or previous hidden layer) sequence:

$$Q = K = V = X$$

$$W^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}, W^K \in \mathbb{R}^{d_{\text{model}} \times d_k}, W^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$$

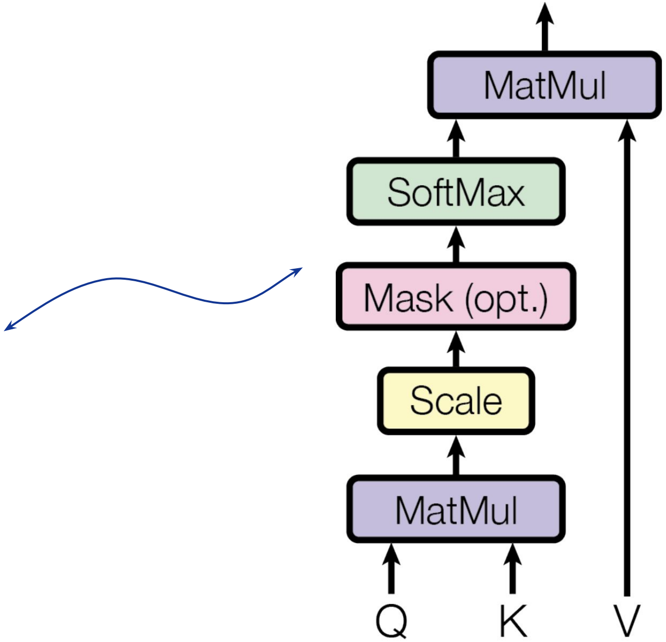
[Vaswani et al. "Attention is all you need", 2017]

Scaled Dot-Product Attention

$$H = \text{Attention}(QW^Q, KW^K, VW^V)$$

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

$$\text{softmax}(x)_i = \frac{e^{x_i}}{\sum_{j=1}^{\dim(x)} e^{x_j}}$$



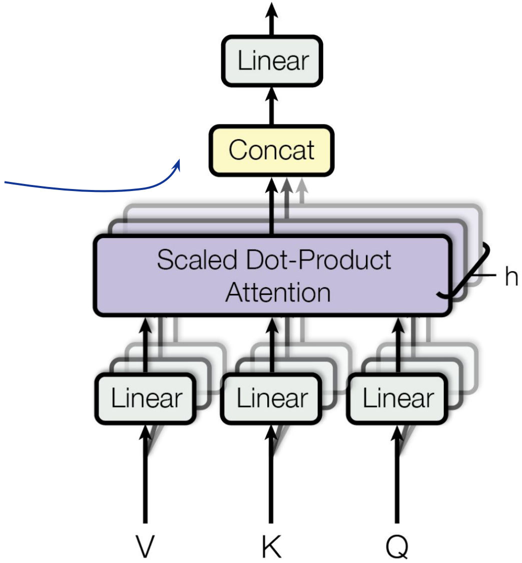
$$W^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}, W^K \in \mathbb{R}^{d_{\text{model}} \times d_k}, W^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$$

[Vaswani et al. "Attention is all you need", 2017]

Multi-Head Attention

$$H_i = \mathrm{Attention}(QW_i^Q, K_iW^K, V_iW^V), i = 1, \ldots, H$$

$$\mathrm{MultiHead}(Q, K, V) = \mathrm{Concat}(H_1, \ldots, H_H)W^O$$



$$W_i^Q \in \mathbb{R}^{d_{\mathrm{model}} \times d_k}, W_i^K \in \mathbb{R}^{d_{\mathrm{model}} \times d_k}, W_i^V \in \mathbb{R}^{d_{\mathrm{model}} \times d_v}$$

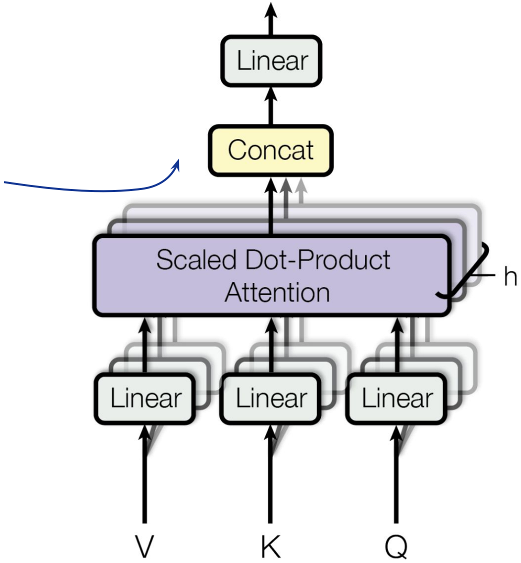
[Vaswani et al. "Attention is all you need", 2017]

Multi-Head Attention

$$H_i = \text{Attention}(QW_i^Q, K_iW^K, V_iW^V), i = 1, \ldots, H$$

$$\text{MultiHead}(Q, K, V) = \text{Concat}(H_1, \ldots, H_H)W^O$$

What's the shape of the output weights?



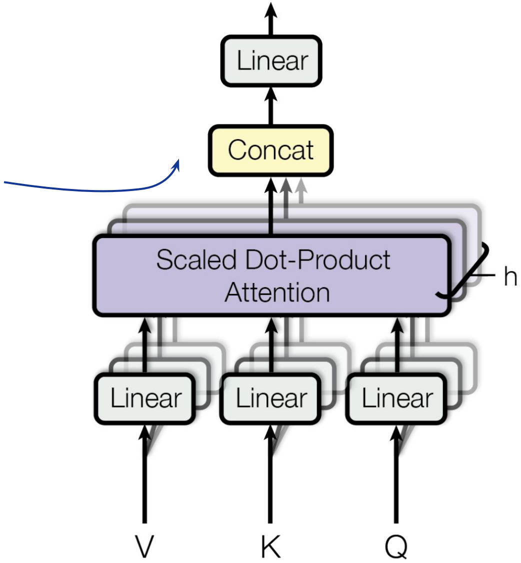
$$W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}, W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}, W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$$

[Vaswani et al. "Attention is all you need", 2017]

Multi-Head Attention

$$H_i = \text{Attention}(QW_i^Q, K_iW^K, V_iW^V), i = 1, \ldots, H$$

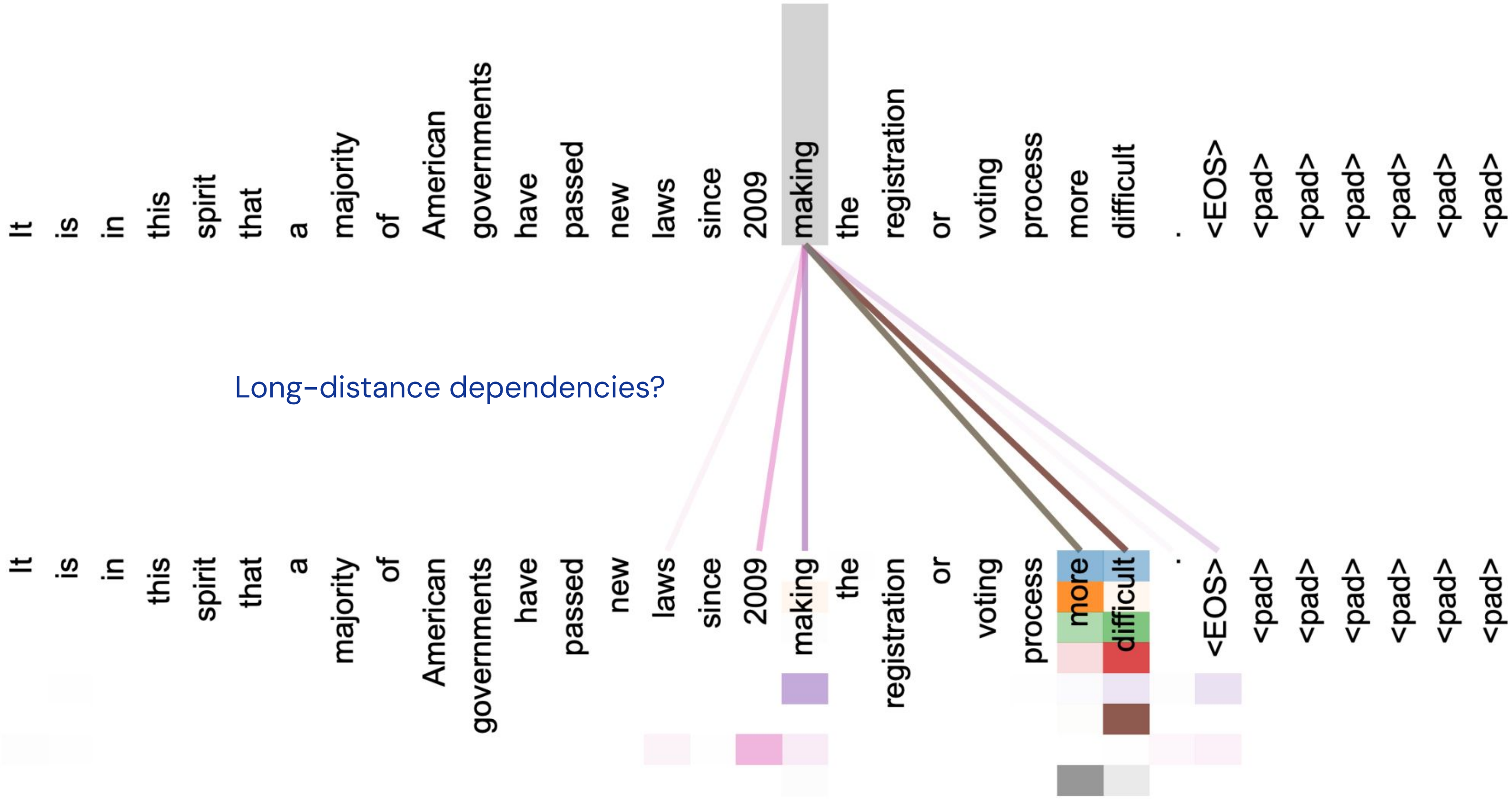
$$\text{MultiHead}(Q, K, V) = \text{Concat}(H_1, \ldots, H_H)W^O$$



$$W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}, W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}, W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v} \text{ and } W^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$$

[Vaswani et al. "Attention is all you need", 2017]

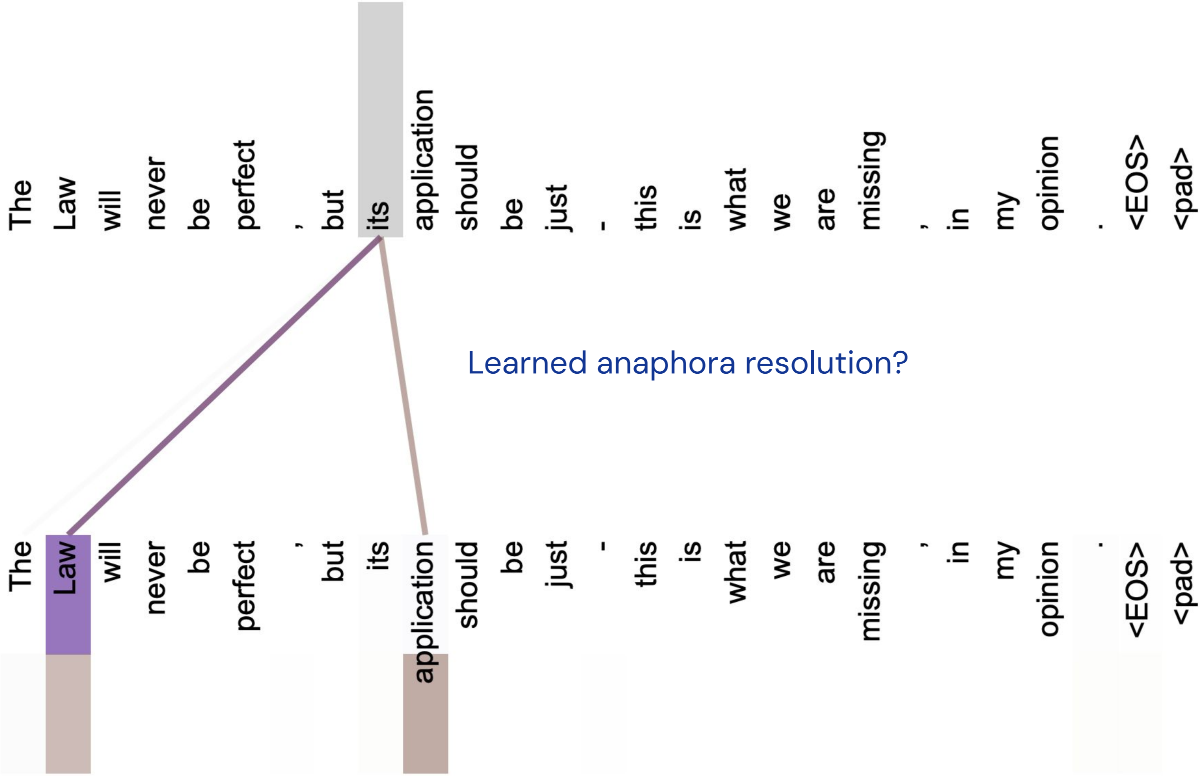
Scaled Dot-Product Attention


It is in this spirit that a majority of American governments have passed new laws since 2009 making the registration or voting process more difficult . <EOS> <pad> <pad> <pad> <pad> <pad> <pad>
It is in this spirit that a majority of American governments have passed new laws since 2009 making the registration or voting process more difficult . <EOS> <pad> <pad> <pad> <pad> <pad> <pad>

Long-distance dependencies?

[Vaswani et al. "Attention is all you need", 2017]

Scaled Dot-Product Attention

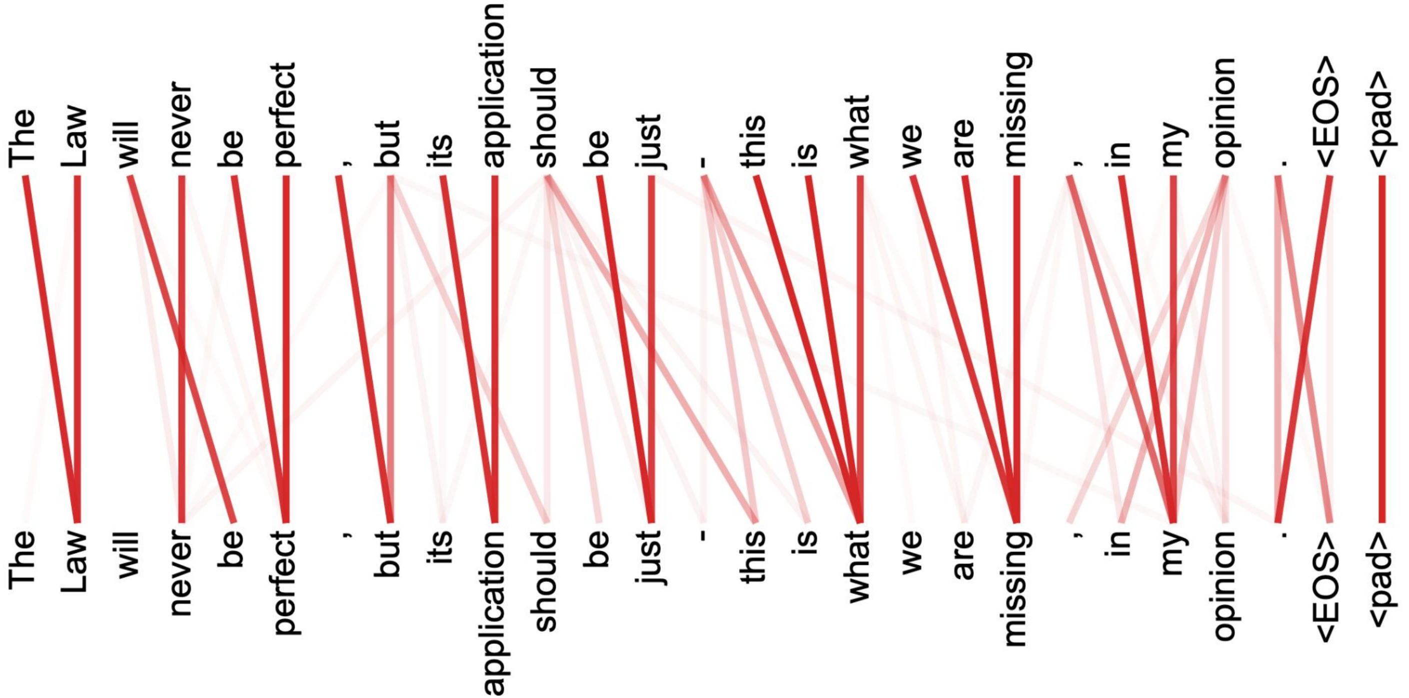

The Law will never be perfect , but its application should be just - this is what we are missing , in my opinion . <EOS> <pad>
The Law will never be perfect , but its application should be just - this is what we are missing , in my opinion . <EOS> <pad>

Learned anaphora resolution?

[Vaswani et al. "Attention is all you need", 2017]

Scaled Dot-Product Attention

Sequence and structure information?

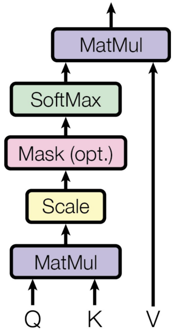


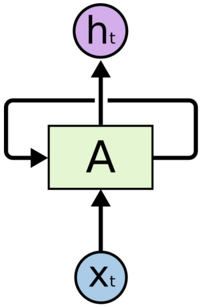
[Vaswani et al. "Attention is all you need", 2017]

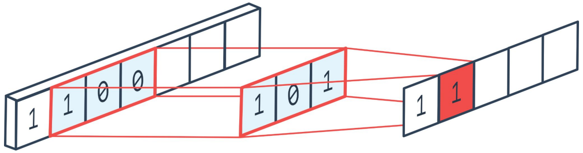
The complexity of self-attention

For full dependency between all elements

Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention			
Recurrent			
Convolutional			



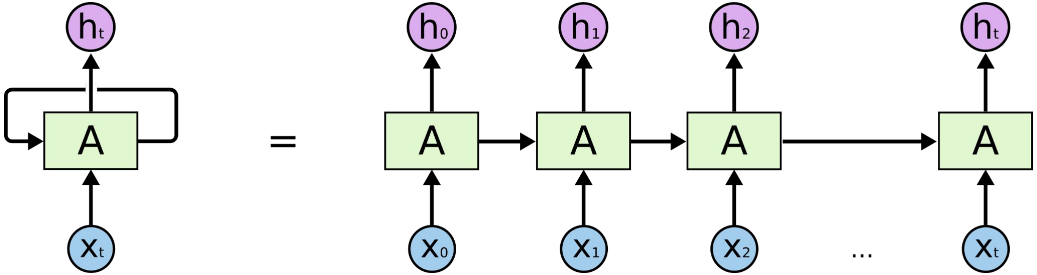




[Vaswani et al. "Attention is all you need", 2017]

The complexity of self-attention

Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Recurrent	$O(\overset{\text{sequence length}}{n} \cdot \underset{\text{dimensionality}}{d}^2)$	$O(n)$	$O(n)$



[Vaswani et al. "Attention is all you need", 2017]

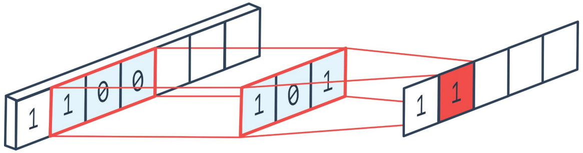
The complexity of self-attention

Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(log_k(n))$

sequence length

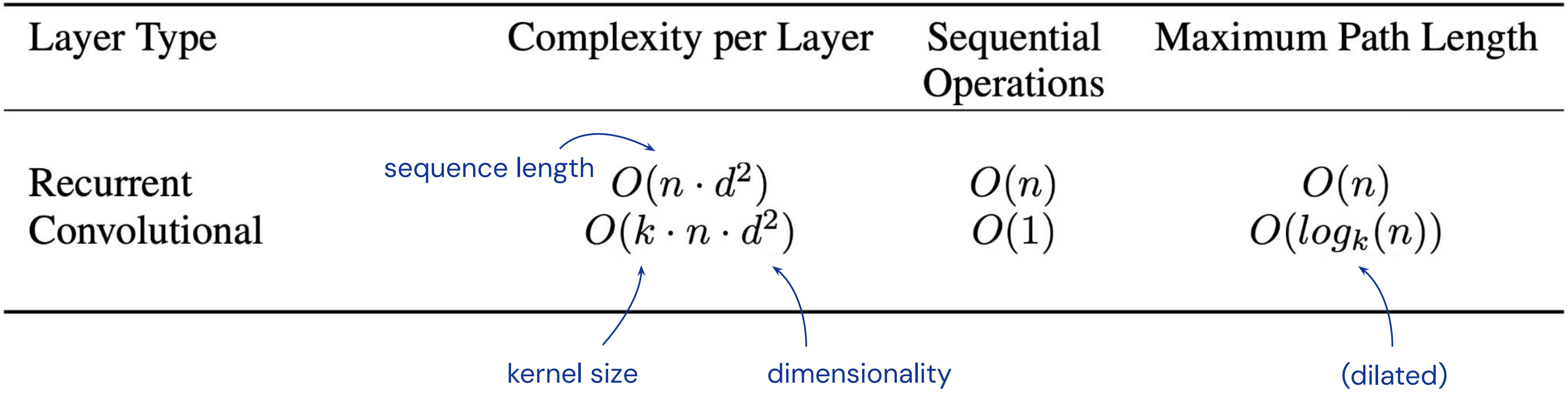
kernel size

dimensionality



[Vaswani et al. "Attention is all you need", 2017]

The complexity of self-attention



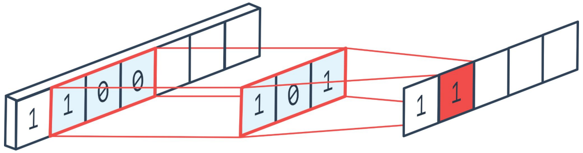
Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(log_k(n))$

sequence length

kernel size

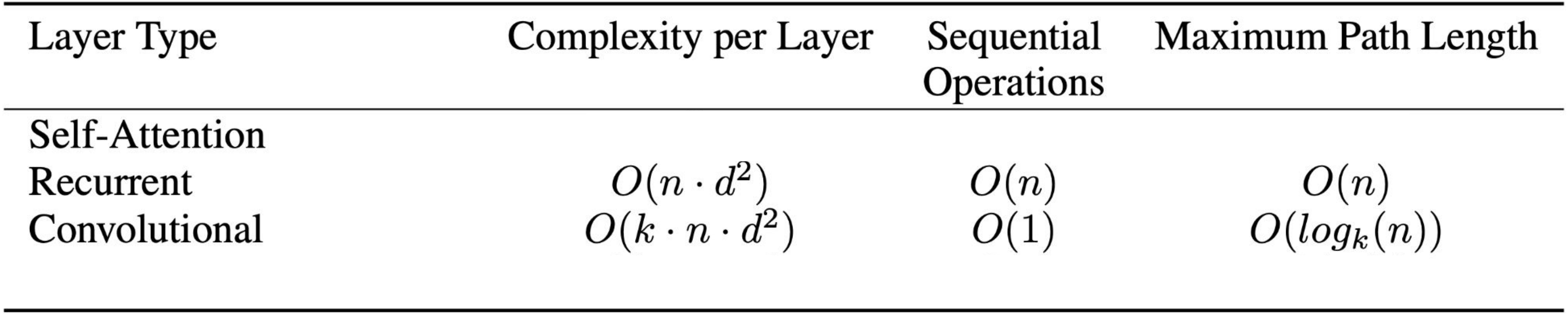
dimensionality

(dilated)

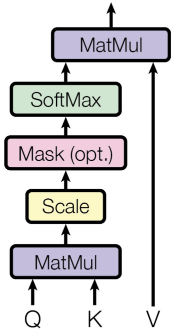


[Vaswani et al. "Attention is all you need", 2017]

The complexity of self-attention



Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention			
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(log_k(n))$

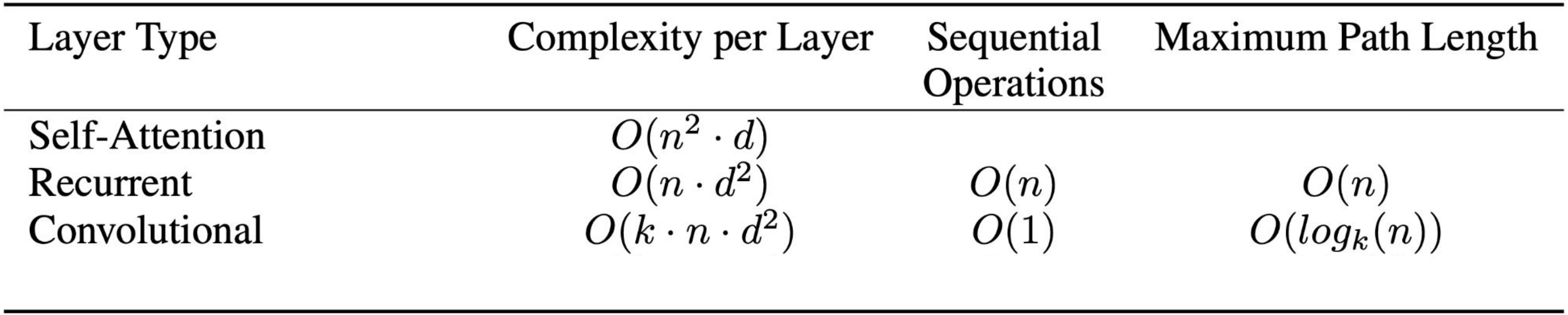


$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^\top}{\sqrt{d_k}})V$$

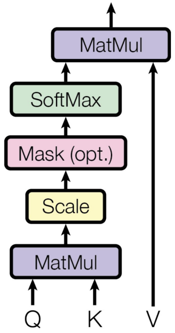
$$Q, K, V \in \mathbb{R}^{N \times d_k}$$

[Vaswani et al. "Attention is all you need", 2017]

The complexity of self-attention



Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O(n^2 \cdot d)$		
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(log_k(n))$

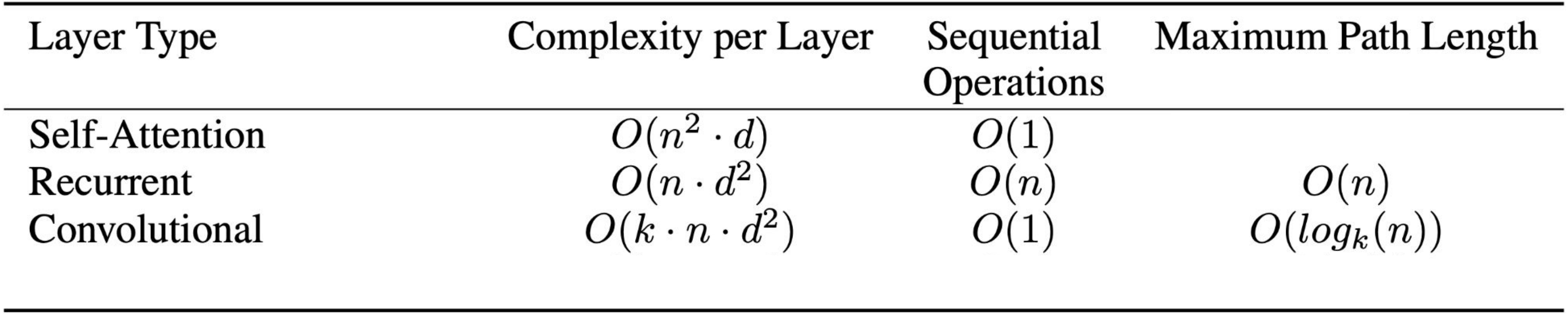


$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^\top}{\sqrt{d_k}})V$$

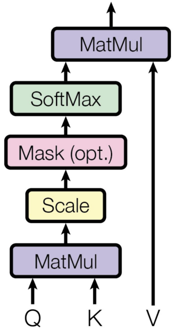
$$Q, K, V \in \mathbb{R}^{N \times d_k}$$

[Vaswani et al. "Attention is all you need", 2017]

The complexity of self-attention



Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(log_k(n))$

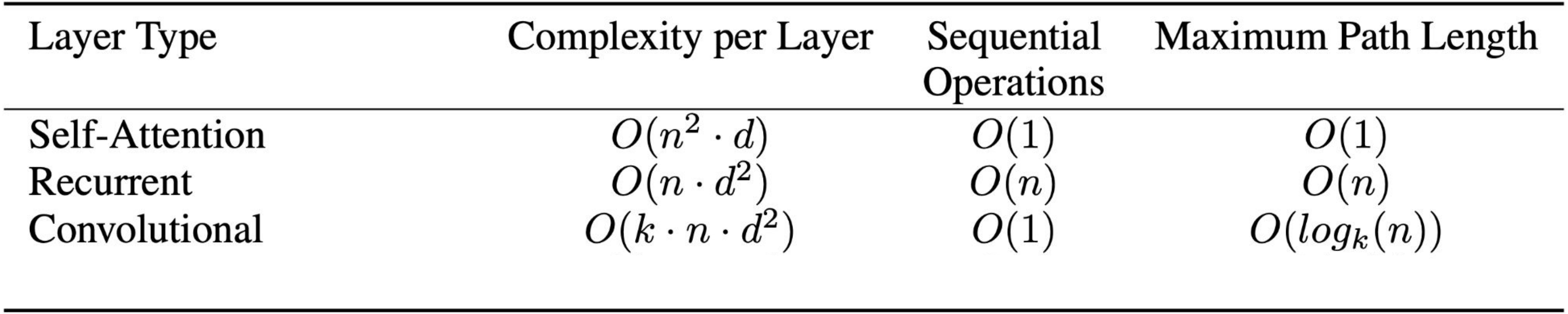


$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^\top}{\sqrt{d_k}})V$$

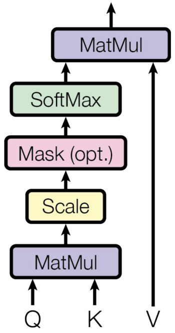
$$Q, K, V \in \mathbb{R}^{N \times d_k}$$

[Vaswani et al. "Attention is all you need", 2017]

The complexity of self-attention



Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(log_k(n))$

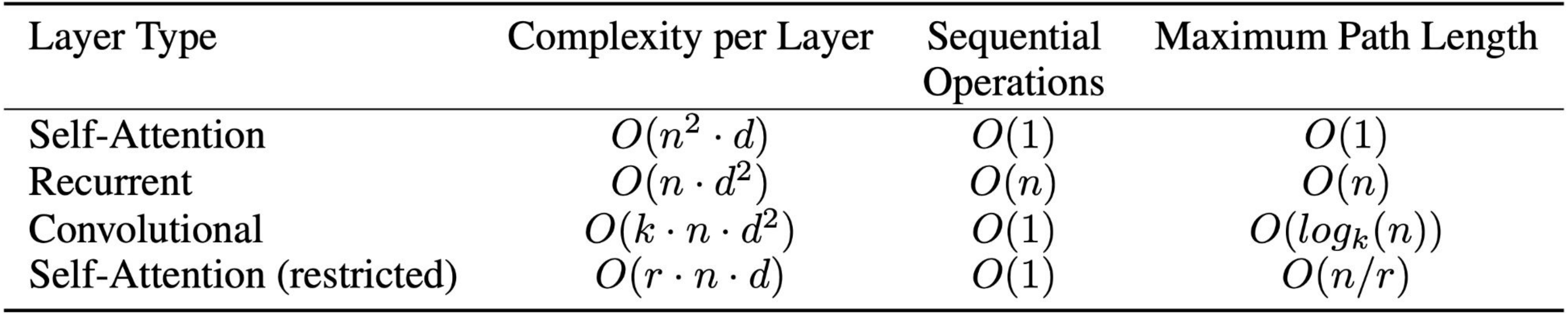


$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^\top}{\sqrt{d_k}})V$$

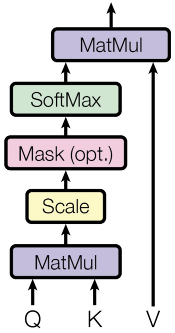
$$Q, K, V \in \mathbb{R}^{N \times d_k}$$

[Vaswani et al. "Attention is all you need", 2017]

The complexity of self-attention



Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(log_k(n))$
Self-Attention (restricted)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

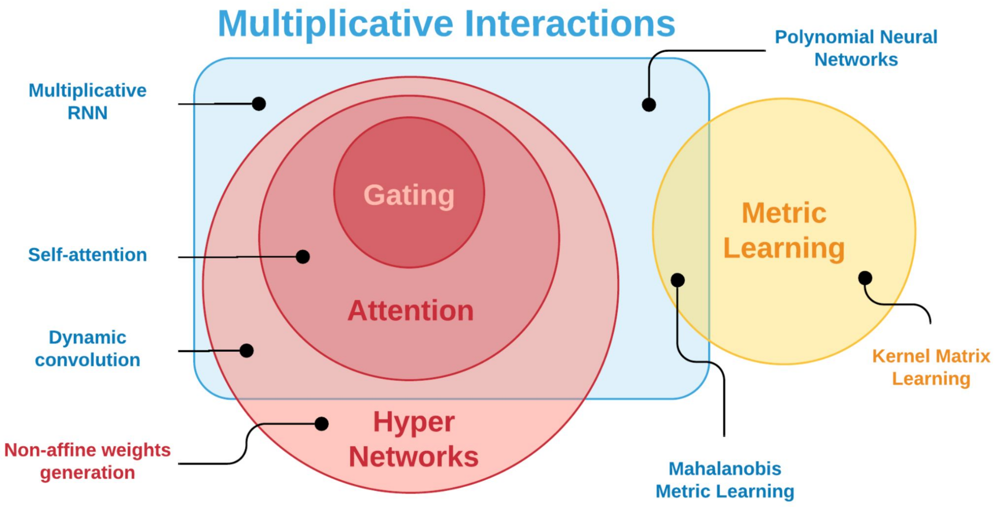


$$\mathrm{Attention}(Q, K, V) = \mathrm{softmax}(\frac{QK^\top}{\sqrt{d_k}})V$$

$$Q, K, V \in \mathbb{R}^{N \times d_k}$$

[Vaswani et al. "Attention is all you need", 2017]

Understanding Self-Attention

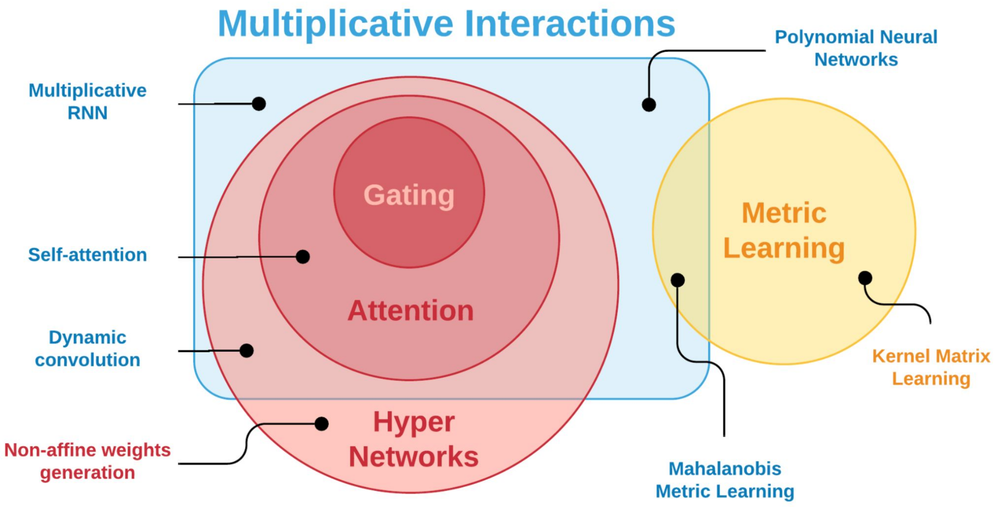


[Jayakumar et al. "Multiplicative Interactions and where to find them", 2019]

Understanding Self-Attention

General class of multiplicative Interactions:

$$f(\mathbf{x},\mathbf{z}) = \mathbf{z}^T \mathbb{W}\mathbf{x} + \mathbf{z}^T \mathbf{U} + \mathbf{V}\mathbf{x} + \mathbf{b}$$



[Jayakumar et al. "Multiplicative Interactions and where to find them", 2019]

Understanding Self-Attention

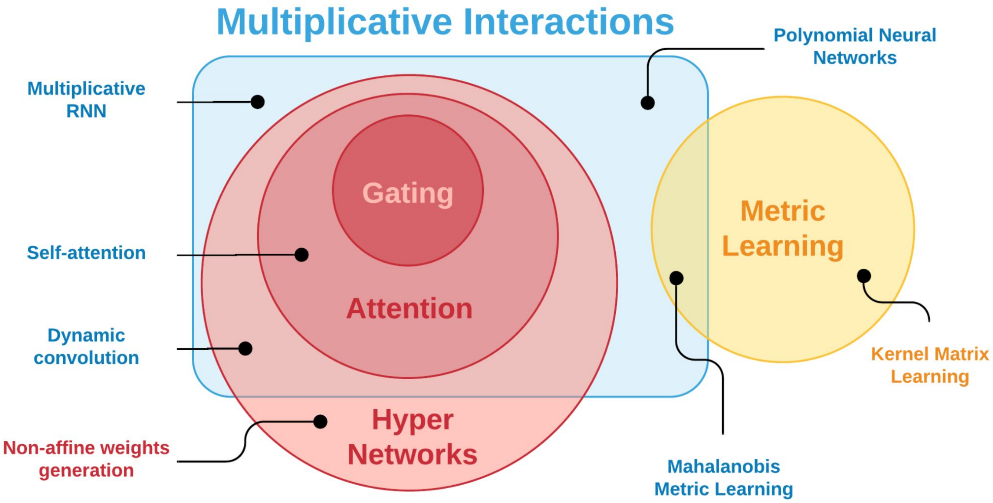
General class of multiplicative Interactions:

$$f(\mathbf{x}, \mathbf{z}) = \mathbf{z}^T \mathbb{W} \mathbf{x} + \mathbf{z}^T \mathbf{U} + \mathbf{V} \mathbf{x} + \mathbf{b}$$

can be written as:

$$\mathbf{W}' = \mathbf{z}^T \mathbb{W} + \mathbf{V} \qquad \mathbf{b}' = \mathbf{z}^T \mathbf{U} + \mathbf{b}$$

$$\mathbf{y} = \mathbf{W}' \mathbf{x} + \mathbf{b}'$$



[Jayakumar et al. "Multiplicative Interactions and where to find them", 2019]

Understanding Self-Attention

General class of multiplicative Interactions:

$$f(\mathbf{x}, \mathbf{z}) = \mathbf{z}^T \mathbb{W} \mathbf{x} + \mathbf{z}^T \mathbf{U} + \mathbf{V}\mathbf{x} + \mathbf{b}$$

can be written as:

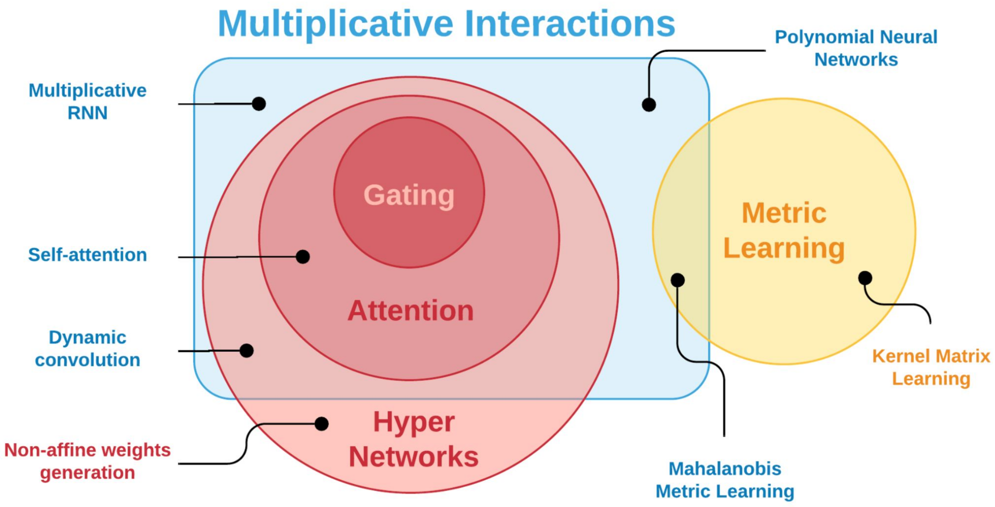
$$\mathbf{W}' = \mathbf{z}^T \mathbb{W} + \mathbf{V} \qquad \mathbf{b}' = \mathbf{z}^T \mathbf{U} + \mathbf{b}$$

$$\mathbf{y} = \mathbf{W}'\mathbf{x} + \mathbf{b}'$$

Consider diagonal approximation:

$$\mathbf{W}' = \mathrm{diag}(a_1, ..., a_n) \quad f = \mathbf{a} \odot \mathbf{x}$$

(similarly for biases)



[Jayakumar et al. "Multiplicative Interactions and where to find them", 2019]

Understanding Self-Attention

General class of multiplicative Interactions:

$$f(\mathbf{x},\mathbf{z}) = \mathbf{z}^T \mathbb{W}\mathbf{x} + \mathbf{z}^T\mathbf{U} + \mathbf{V}\mathbf{x} + \mathbf{b}$$

can be written as:

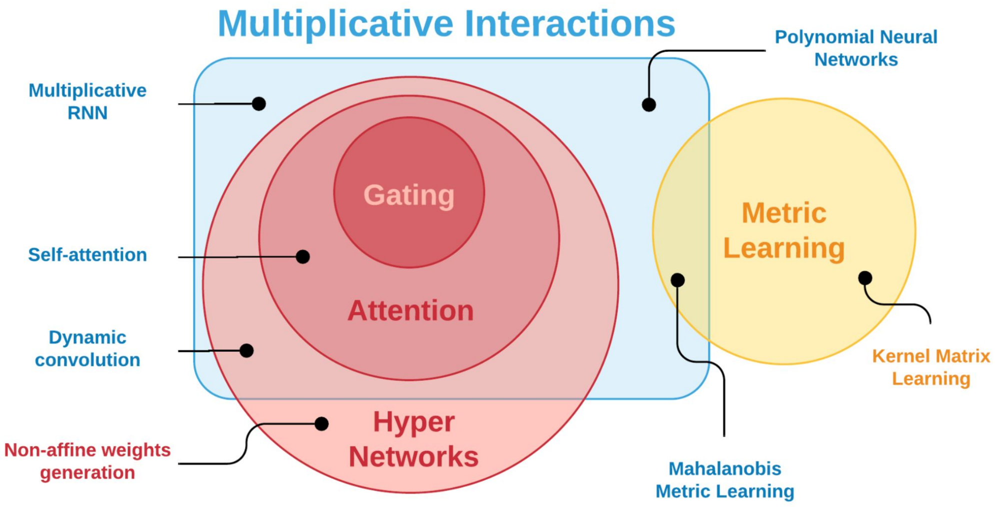
$$\mathbf{W}' = \mathbf{z}^T\mathbb{W} + \mathbf{V} \qquad \mathbf{b}' = \mathbf{z}^T\mathbf{U} + \mathbf{b}$$

$$\mathbf{y} = \mathbf{W}'\mathbf{x} + \mathbf{b}'$$

Consider diagonal approximation:

$$\mathbf{W}' = \mathrm{diag}(a_1, ..., a_n) \quad f = \mathbf{a} \odot \mathbf{x}$$

(similarly for biases)

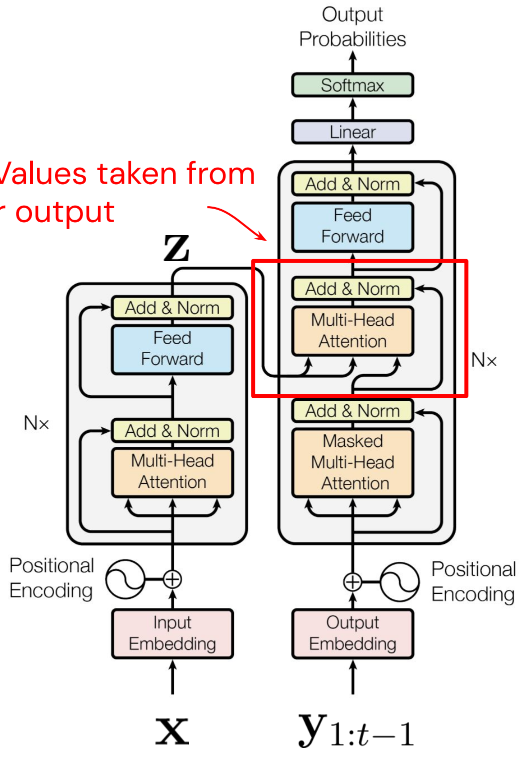


Then, Self-Attention is (with **m** bounded):

$$\mathbf{m} = f(\mathbf{x},\mathbf{z}) \quad \mathbf{y} = \mathbf{m} \odot \mathbf{x}$$

[Jayakumar et al. "Multiplicative Interactions and where to find them", 2019]

Encoder-Decoder v Decoder-Only Transformers



Keys & Values taken from Encoder output

$$H_i^{(l)} = \text{Attention}(QW_i^Q, K_iW^K, V_iW^V)$$

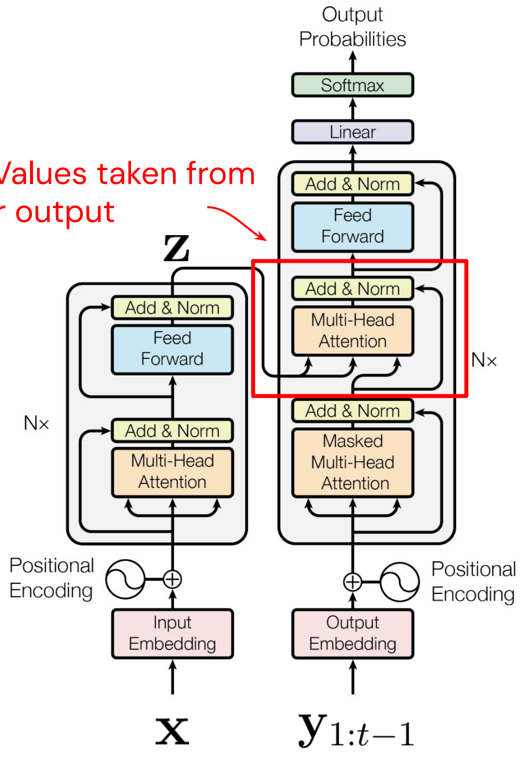
$$Q = Y^{(l-1)}$$

$$K = V = Z^{(l-1)}$$

Encoder – Decoder

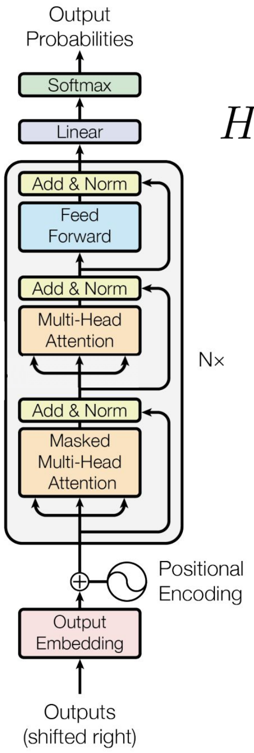
[Jayakumar et al. "Multiplicative Interactions and where to find them", 2019]

Encoder-Decoder v Decoder-Only Transformers



Keys & Values taken from Encoder output

Encoder – Decoder



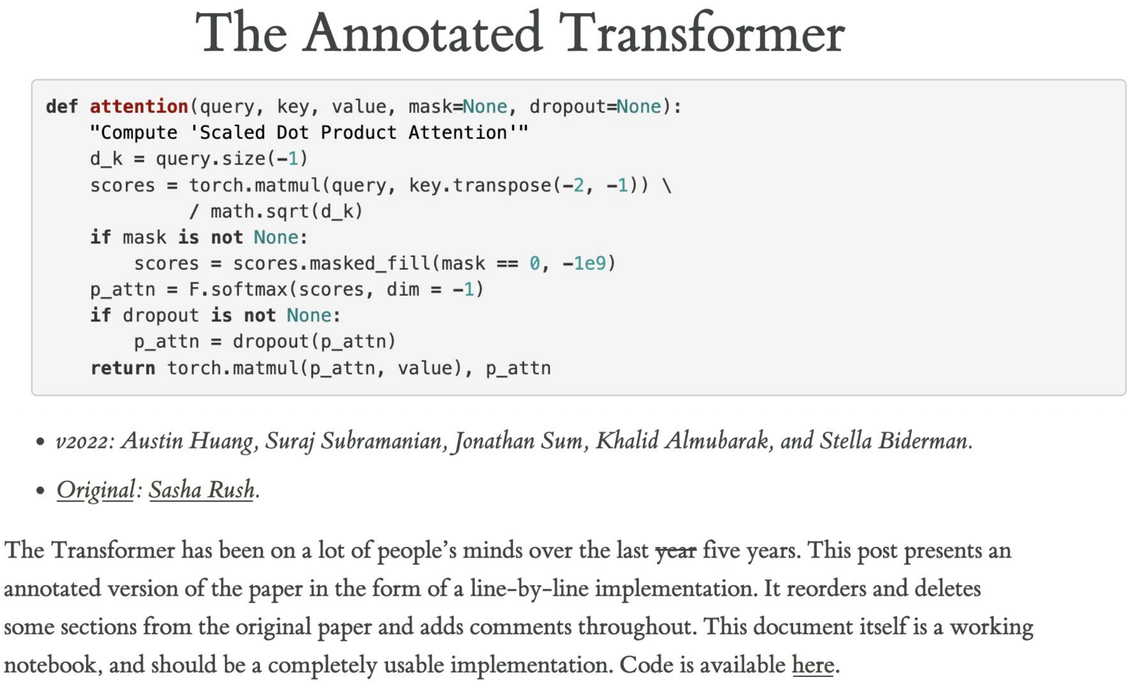
Decoder only

$$H_i^{(l)} = \text{Attention}(QW_i^Q, K_iW^K, V_iW^V)$$

$$Q = K = V = Y^{(l-1)}$$

[Jayakumar et al. "Multiplicative Interactions and where to find them", 2019]

Homework for Everyone (Credit to Harvard NLP)



The Annotated Transformer

```
def attention(query, key, value, mask=None, dropout=None):
    "Compute 'Scaled Dot Product Attention'"
    d_k = query.size(-1)
    scores = torch.matmul(query, key.transpose(-2, -1)) \
             / math.sqrt(d_k)
    if mask is not None:
        scores = scores.masked_fill(mask == 0, -1e9)
    p_attn = F.softmax(scores, dim = -1)
    if dropout is not None:
        p_attn = dropout(p_attn)
    return torch.matmul(p_attn, value), p_attn
```

- *v2022: Austin Huang, Suraj Subramanian, Jonathan Sum, Khalid Almubarak, and Stella Biderman.*
- *Original: Sasha Rush.*

The Transformer has been on a lot of people's minds over the last ~~year~~ five years. This post presents an annotated version of the paper in the form of a line-by-line implementation. It reorders and deletes some sections from the original paper and adds comments throughout. This document itself is a working notebook, and should be a completely usable implementation. Code is available here.

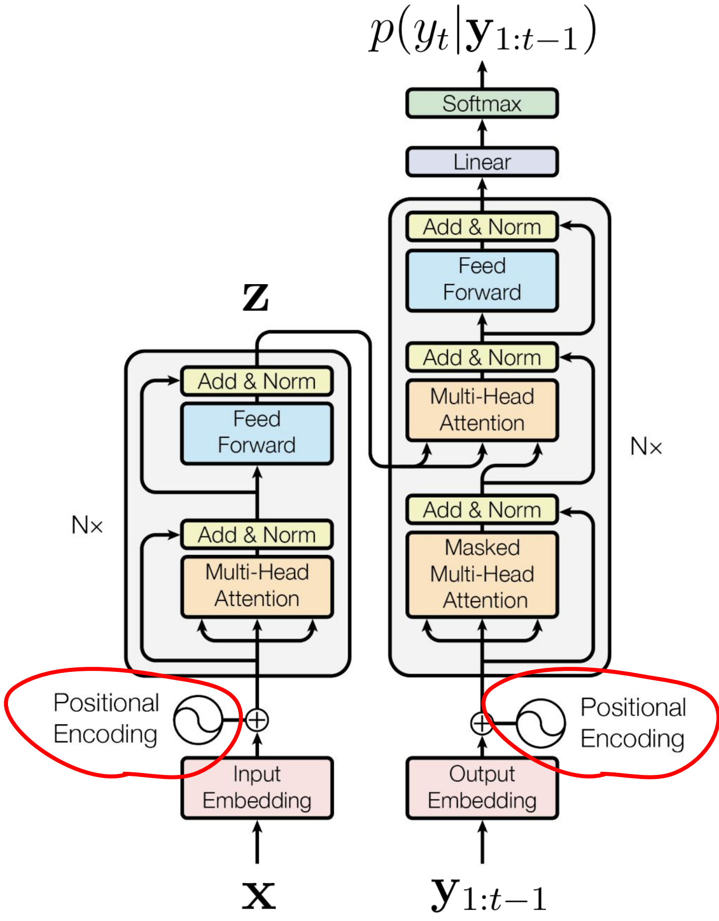
[The Annotated Transformer]

The Transformer Building Blocks

1. **Multi-head Attention**

2. **Position Encodings**

3. **Residual connections + Normalization**



[Vaswani et al. "Attention is all you need", 2017]

Positional Encodings

In order to allow the model to distinguish between sequence positions, we use positional encodings.

[Vaswani et al. "Attention is all you need", 2017,
More details on positional encodings, Stanford XCS224U]

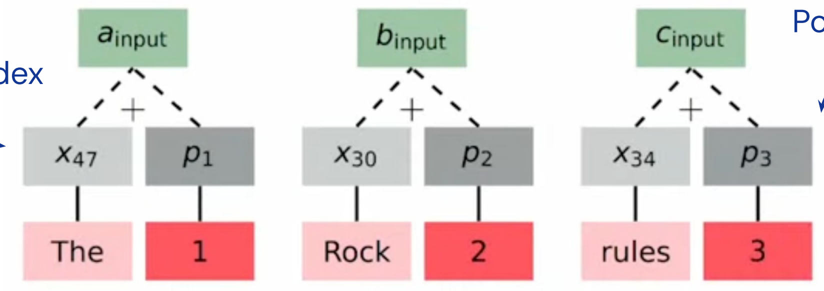
Positional Encodings

In order to allow the model to distinguish between sequence positions, we use positional encodings. Key questions:

1. Does the set of positions need to be decided ahead of time?
2. Does the scheme hinder generalization to new positions?

[Vaswani et al. "Attention is all you need", 2017,
More details on positional encodings, Stanford XCS224U]

Absolute Positional Encodings

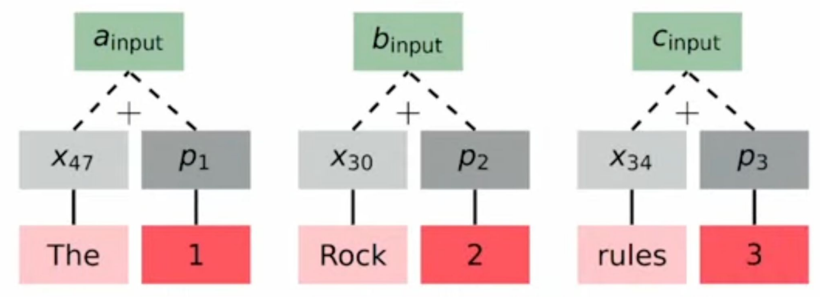


Vocabulary index

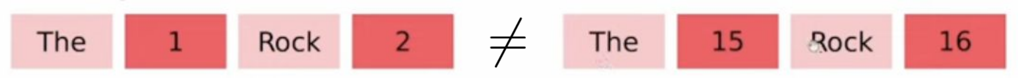
Positional encoding

[Vaswani et al. "Attention is all you need", 2017,
More details on positional encodings, Stanford XCS224U]

Absolute Positional Encodings



1. Set of positions need to be decided ahead of time (to normalize).
2. Scheme hinders generalization to new positions:



[Vaswani et al. "Attention is all you need", 2017,
More details on positional encodings, Stanford XCS224U]

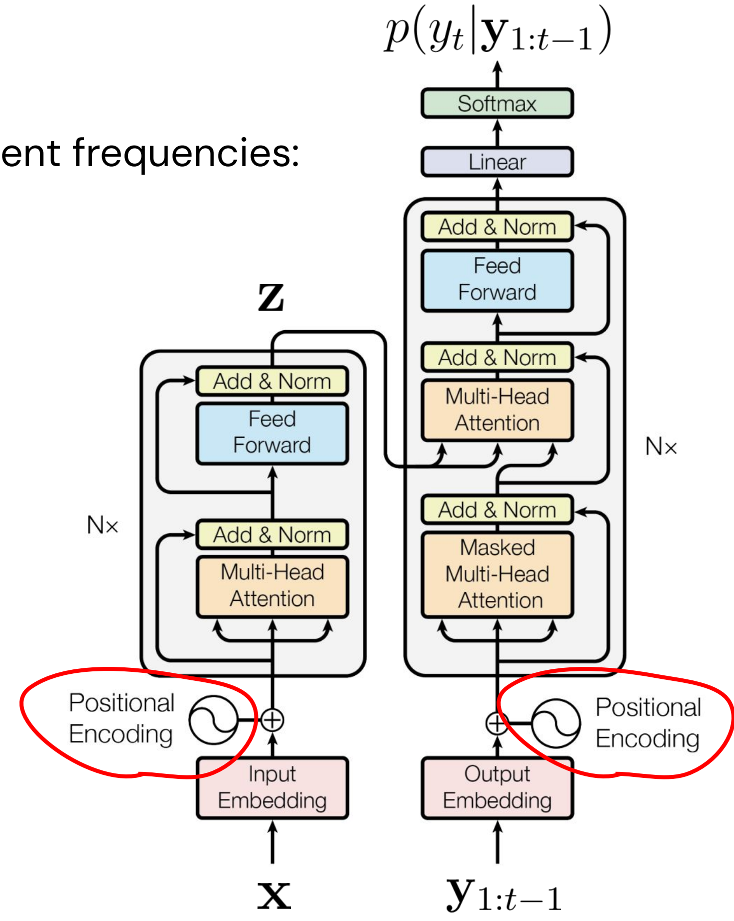
Frequency Positional Encodings

Instead, the original transformer used additive values at different frequencies:

$$PE_{t,2i} = \sin\left(\frac{t}{10{,}000^{\frac{2i}{d_{model}}}}\right)$$

$$PE_{t,2i+1} = \cos\left(\frac{t}{10{,}000^{\frac{2i+1}{d_{model}}}}\right)$$

where PE has the same **dimensionality** as our embeddings and we have a wavelength from $(2\pi, 10000 \cdot 2\pi)$



[Vaswani et al. "Attention is all you need", 2017, More details on positional encodings, Stanford XCS224U]

Frequency Positional Encodings

Instead, the original transformer used additive values at different frequencies:

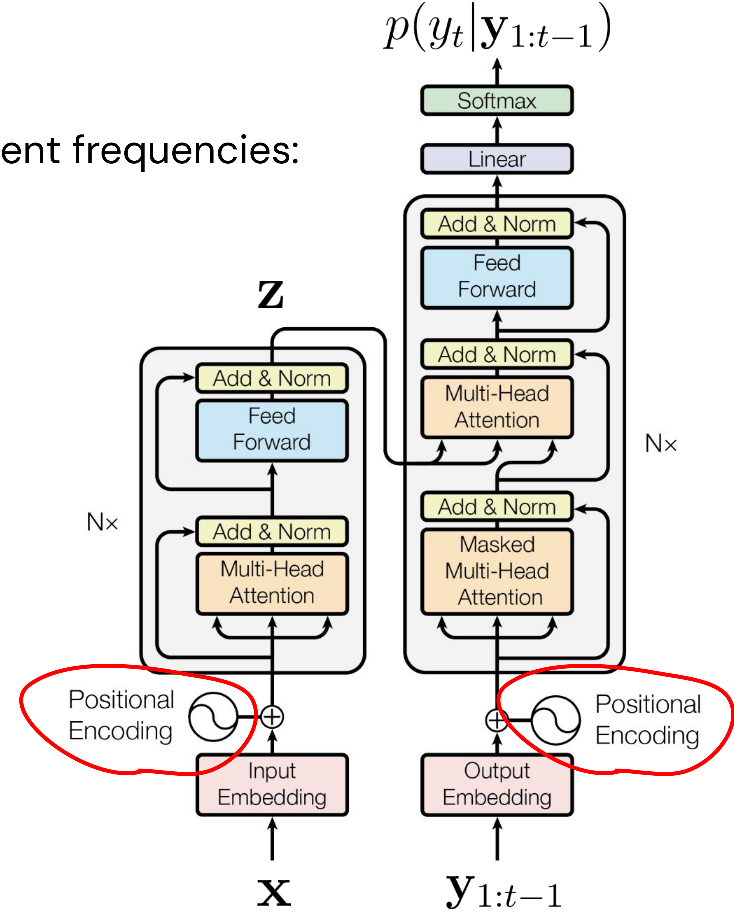
$$PE_{\underline{t,2i}} = \sin\left(\frac{t}{10{,}000^{\frac{2i}{d_{model}}}}\right)$$

Position, embedding id

$$PE_{t,2i+1} = \cos\left(\frac{t}{10{,}000^{\frac{2i+1}{d_{model}}}}\right)$$

where PE has the same **dimensionality** as our embeddings

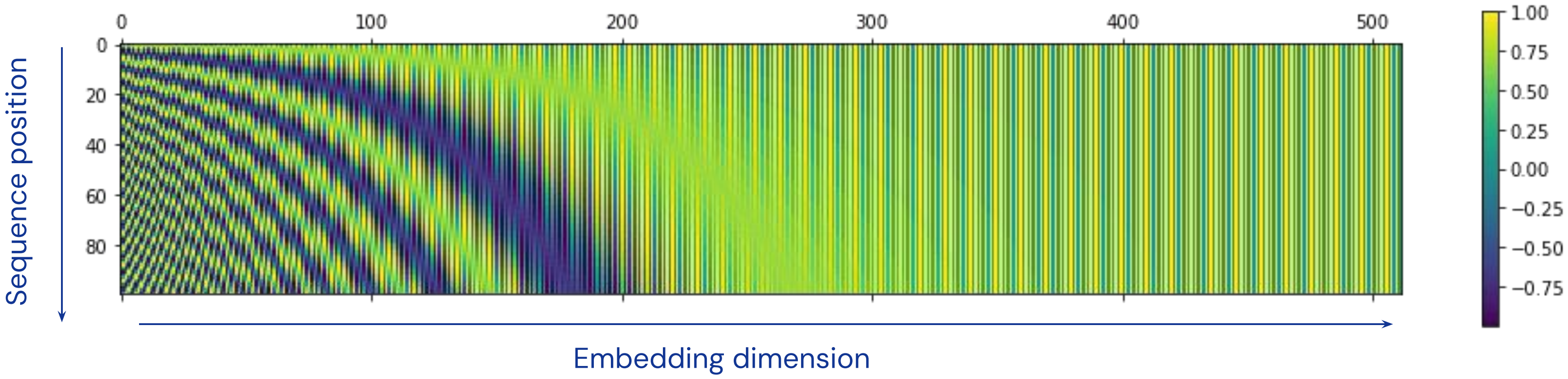
and we have a wavelength from $(2\pi, 10000 \cdot 2\pi)$



[Vaswani et al. "Attention is all you need", 2017,
More details on positional encodings, Stanford XCS224U]

Frequency Positional Encodings

Positional encodings for a sequence of 100 items with an embedding dimensionality of 512



1. Set of positions need to be decided ahead of time
2. Scheme hinders generalization to new positions:

[Vaswani et al. "Attention is all you need", 2017,
More details on positional encodings, Stanford XCS224U,
A gentle introduction into positional encodings]

[Vaswani et al. "Attention is all you need", 2017,
More details on positional encodings, Stanford XCS224U]

Modern Positional Encodings

Frequency Positional Encodings have essentially been replaced in modern Transformers.
Popular alternatives

(1) Relative positional encodings (Example with window size 1):

[Vaswani et al. "Attention is all you need", 2017,
More details on positional encodings, Stanford XCS224U,
A gentle introduction into positional encodings]

Modern Positional Encodings

Frequency Positional Encodings have essentially been replaced in modern Transformers.

Popular alternatives

(1) Relative positional encodings (Example with window size 1):

$$\mathbf{v}_i^{(l)} = \sum_{j=1}^{N} (\alpha_{i,j} \cdot \mathbf{v}_j^{l-1}) + \mathbf{p}_{i,j}^{v}$$

[Vaswani et al. "Attention is all you need", 2017,
More details on positional encodings, Stanford XCS224U,
A gentle introduction into positional encodings]

Modern Positional Encodings

Frequency Positional Encodings have essentially been replaced in modern Transformers. Popular alternatives

(1) Relative positional encodings (Example with window size 1):

y1	y2	y3	y4	y5
p(3,1) = w(-1)	p(3,2) = w(-1)	p(3,3) = w(0)	p(3,4) = w(1)	p(3,4) = w(1)
p(2,1) = w(-1)	p(2,2) = w(0)	p(2,3) = w(1)	p(2,4) = w(1)	p(2,5) = w(1)

$$\mathbf{v}_i^{(l)} = \sum_{j=1}^{N} (\alpha_{i,j} \cdot \mathbf{v}_j^{l-1}) + \mathbf{p}_{i,j}^{v}$$

[Vaswani et al. "Attention is all you need", 2017,
More details on positional encodings, Stanford XCS224U,
A gentle introduction into positional encodings]

Modern Positional Encodings

Frequency Positional Encodings have essentially been replaced in modern Transformers.
Popular alternatives

(1) Relative positional encodings (Example with window size 1):

y1	y2	y3	y4	y5
p(3,1) = w(-1)	p(3,2) = w(-1)	p(3,3) = w(0)	p(3,4) = w(1)	p(3,4) = w(1)
p(2,1) = w(-1)	p(2,2) = w(0)	p(2,3) = w(1)	p(2,4) = w(1)	p(2,5) = w(1)

$$\mathbf{v}_i^{(l)} = \sum_{j=1}^{N} (\alpha_{i,j} \cdot \mathbf{v}_j^{l-1}) + \mathbf{p}_{i,j}^{v}$$

(2) Rotary positional encodings (RoPE, Su et al., 2021)

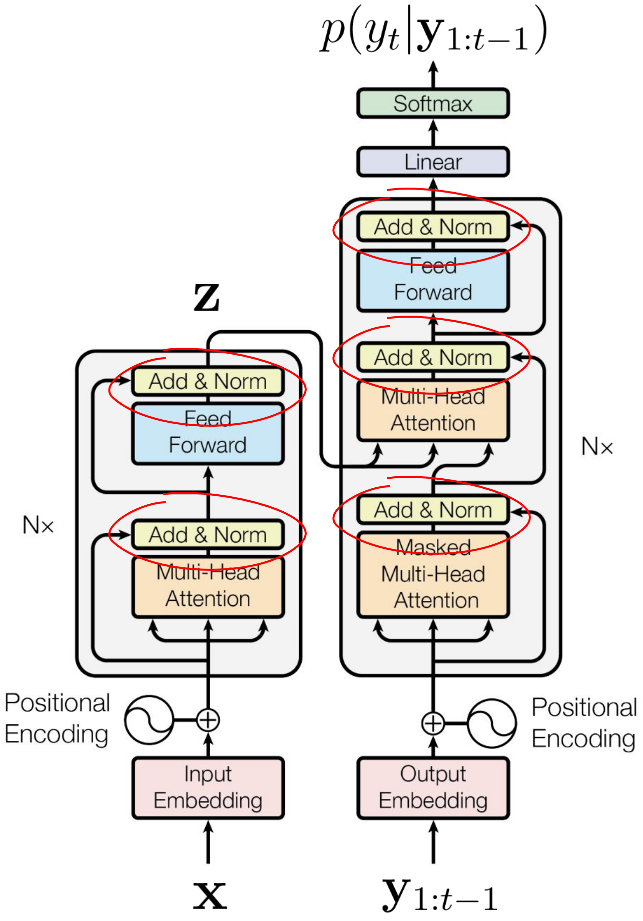
[Vaswani et al. "Attention is all you need", 2017,
More details on positional encodings, Stanford XCS224U,
A gentle introduction into positional encodings]

The Transformer Building Blocks

1. Multi-head Attention

2. Position Encodings

3. Residual connections + Normalization

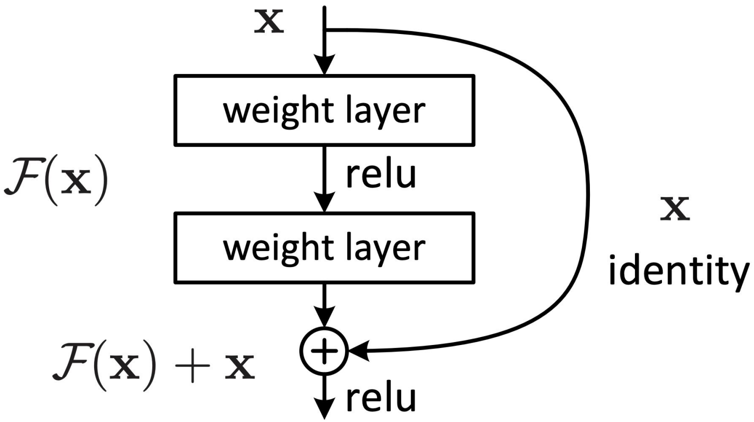


[Vaswani et al. "Attention is all you need", 2017]

Residual Connections + Normalization

$\mathcal{H}(\mathbf{x})$ → desired mapping

$\mathcal{F}(\mathbf{x}) := \mathcal{H}(\mathbf{x}) - \mathbf{x}$ → chosen mapping

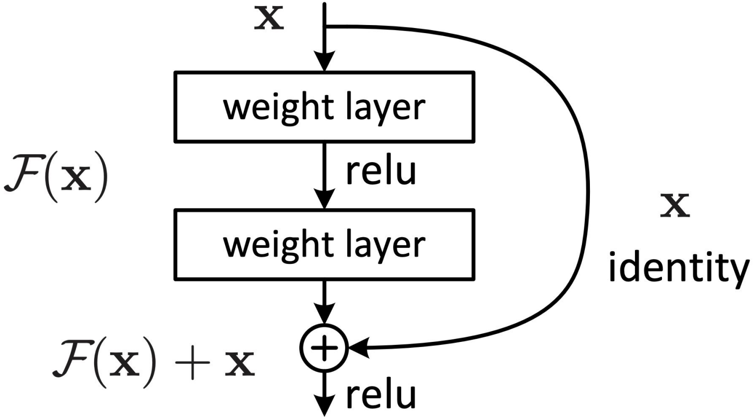


[Deep residual learning for image recognition, CVPR 2016]

Residual Connections + Normalization

$\mathcal{H}(\mathbf{x})$ → desired mapping

$\mathcal{F}(\mathbf{x}) := \mathcal{H}(\mathbf{x}) - \mathbf{x}$ → chosen mapping

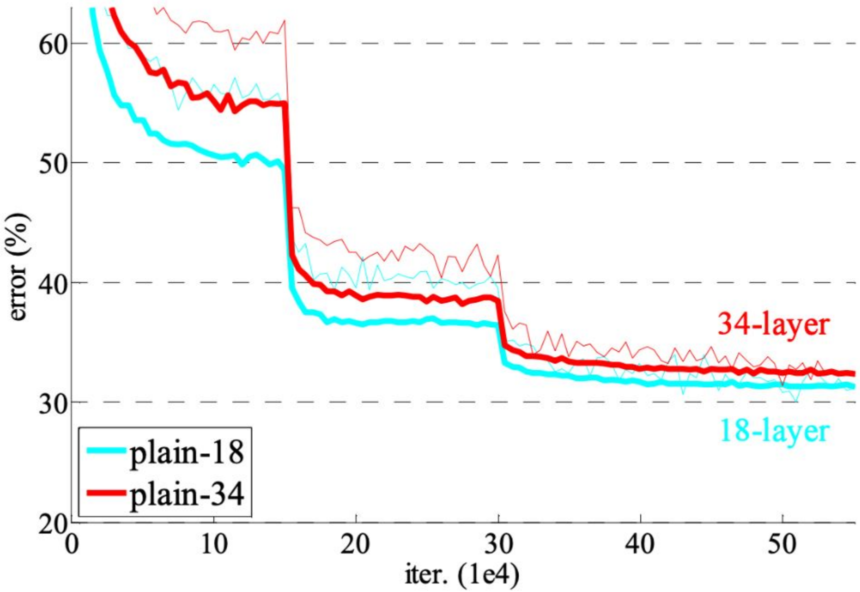


The motivation for this are "skip-connections", which had empirically been observed the help train deeper networks in many previous studies (see vanishing gradient problem (LSTM, Hochreiter et al., 1997)).

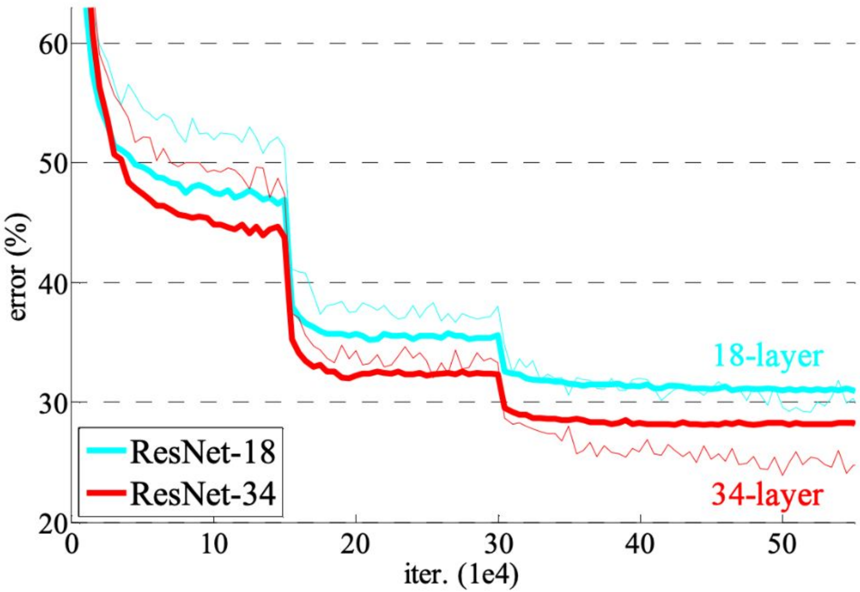
[Deep residual learning for image recognition, CVPR 2016]

Residual Connections + Normalization

Plain network

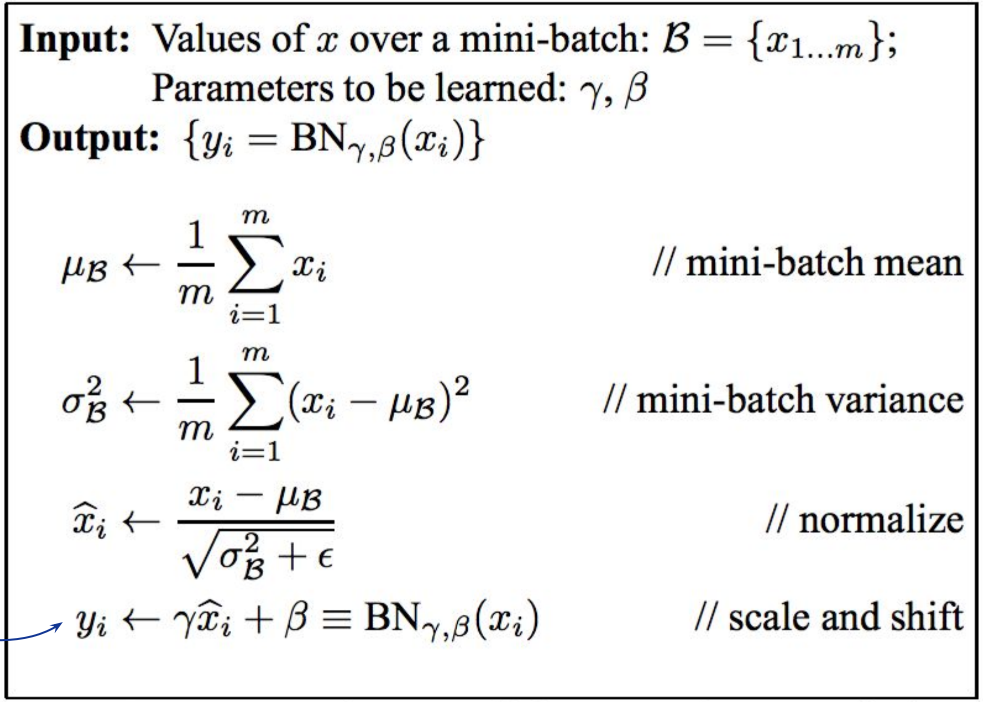


Residual Network



[Deep residual learning for image recognition, CVPR 2016]

Batch Normalization

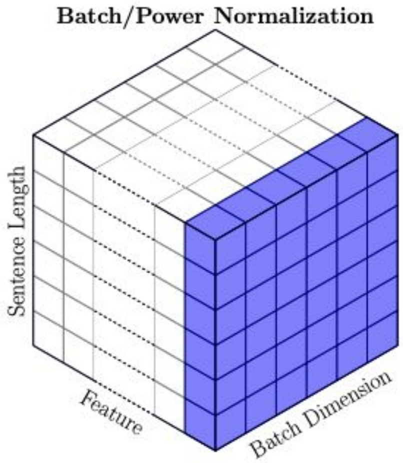


Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1 \ldots m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

```
$\mu_{\mathcal{B}} \leftarrow \frac{1}{m} \sum_{i=1}^{m} x_i$    // mini-batch mean
$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m} \sum_{i=1}^{m} (x_i - \mu_{\mathcal{B}})^2$    // mini-batch variance
$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}}$    // normalize
$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i)$    // scale and shift
```

Pre-Activations

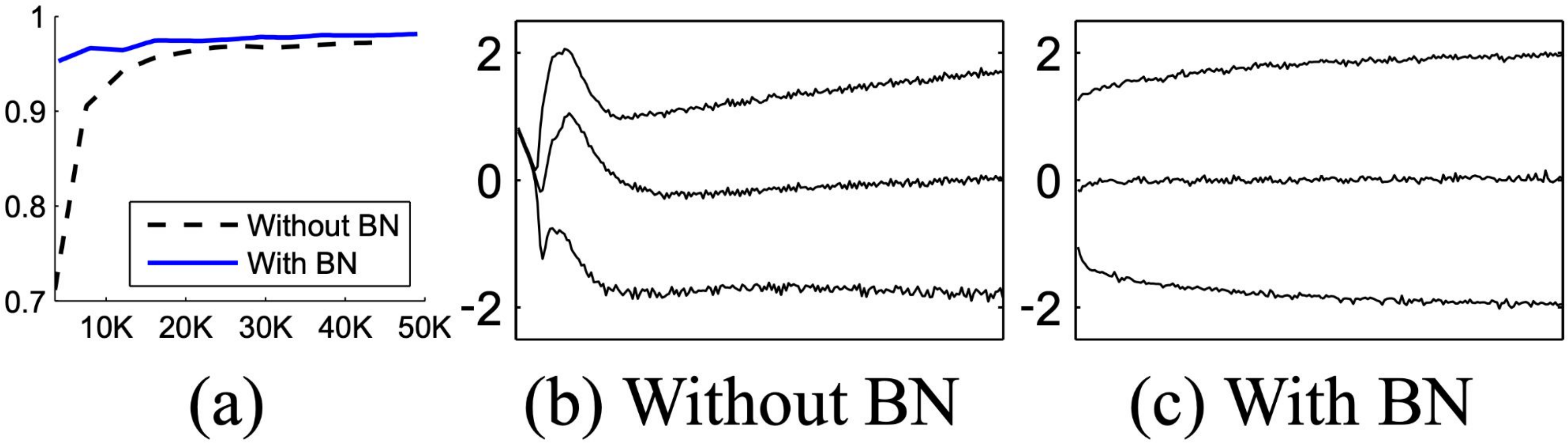


[Batch Normalization, Ioffe. et al, 2015]

Batch Normalization

Task Performance

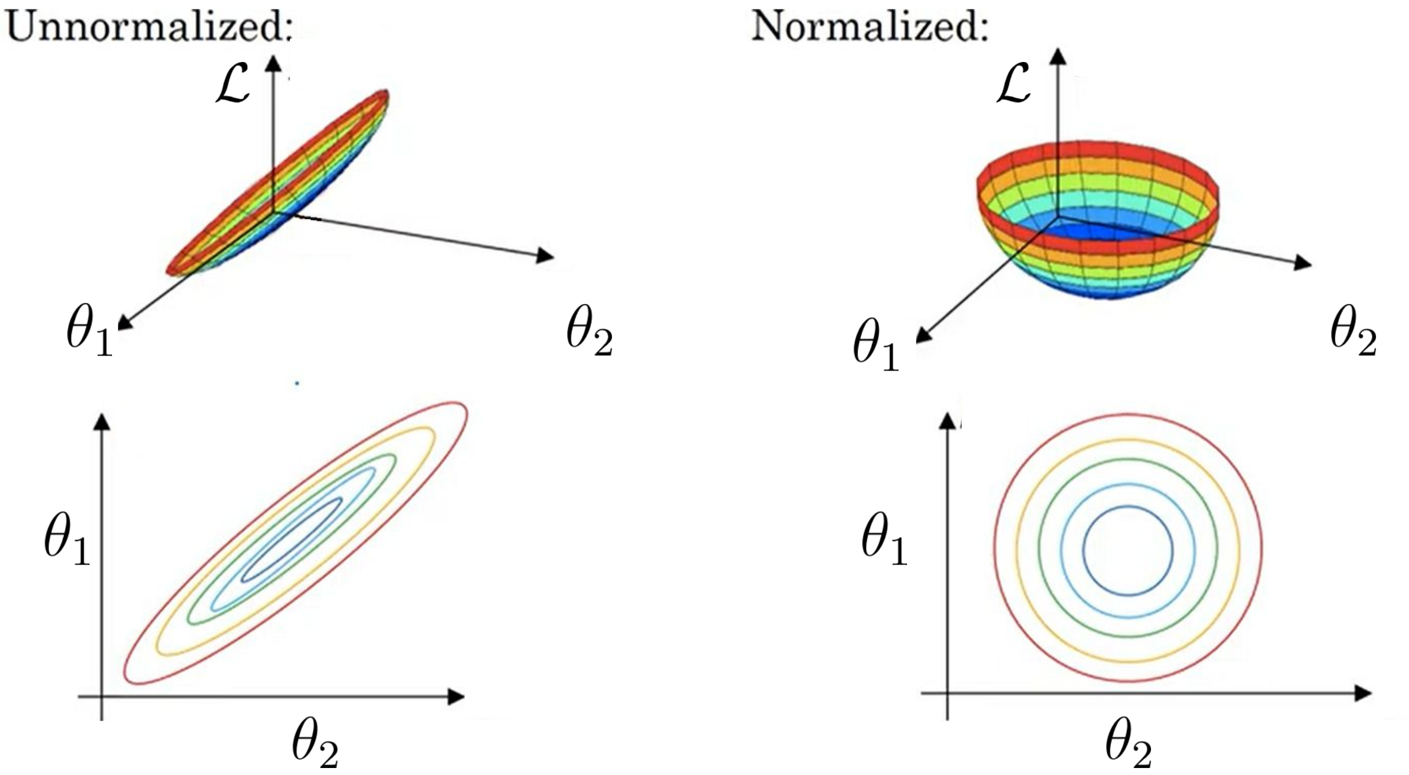
Pre-activation distribution (15%, 50%, 80%) quantile



(a) (b) Without BN (c) With BN

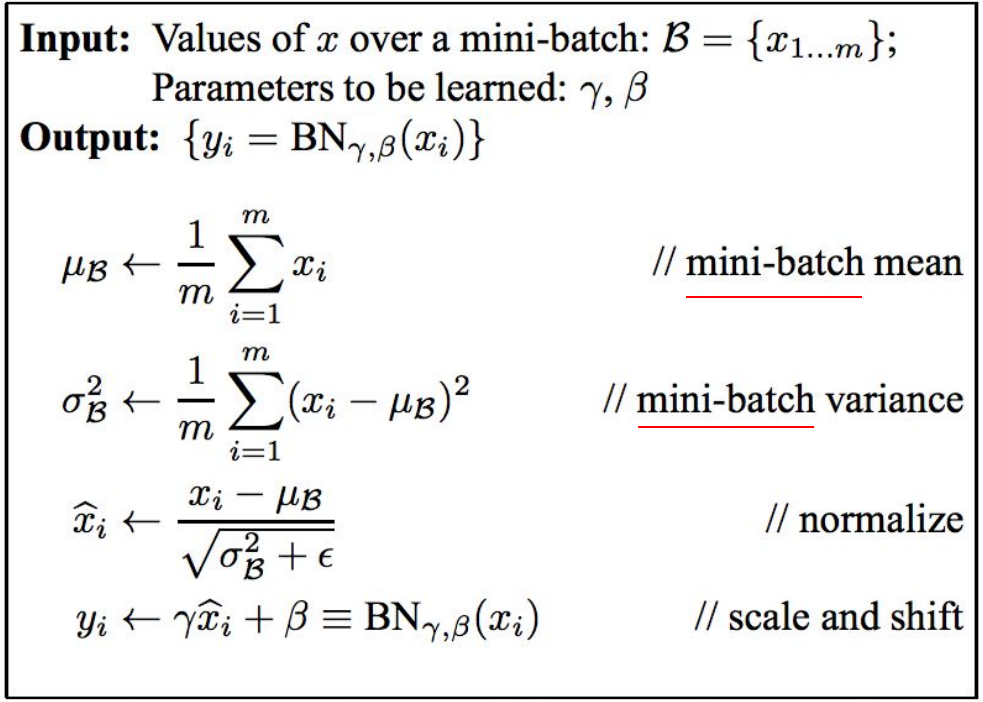
[Batch Normalization, Ioffe. et al, 2015]

Batch Normalization Intuition



[Batch Normalization, Ioffe. et al, 2015, Image Source]

Batch Normalization



Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1\ldots m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \text{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \text{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

This is a bug magnet:

1. Introduces a dependency between examples in the batch
2. Implementations are stateful, to track the statistics over the course of training

This can lead to:

⟶ Information leakage (e.g. in autoregressive % contrastive models)

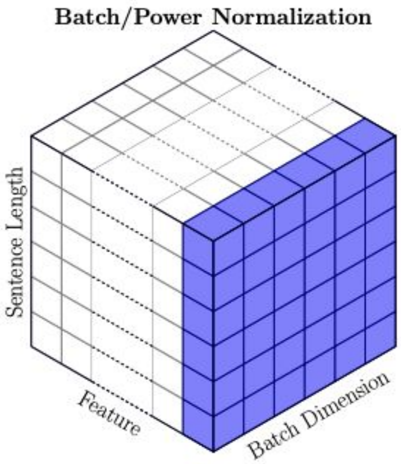
⟶ Unattractive batch size effects

⟶ Affects optimization (improving gradient propagation, but adding randomness in a tightly coupled way)

[Batch Normalization, Ioffe. et al, 2015]

Layer Normalization

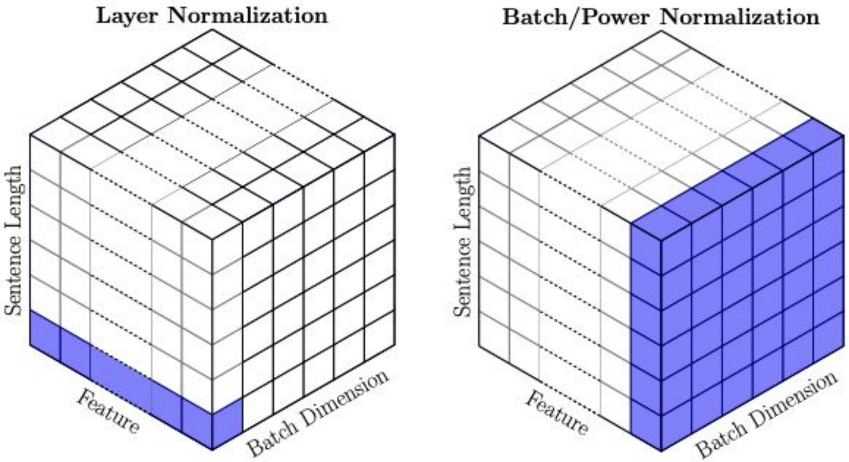
$$\mathbf{h}^t = f\left[\frac{\mathbf{g}}{\sigma^t} \odot \left(\mathbf{a}^t - \mu^t\right) + \mathbf{b}\right] \qquad \mu^t = \frac{1}{H}\sum_{i=1}^{H} a_i^t \qquad \sigma^t = \sqrt{\frac{1}{H}\sum_{i=1}^{H}\left(a_i^t - \mu^t\right)^2}$$



[Layer Normalization, Ba. et al, 2016]

Layer Normalization

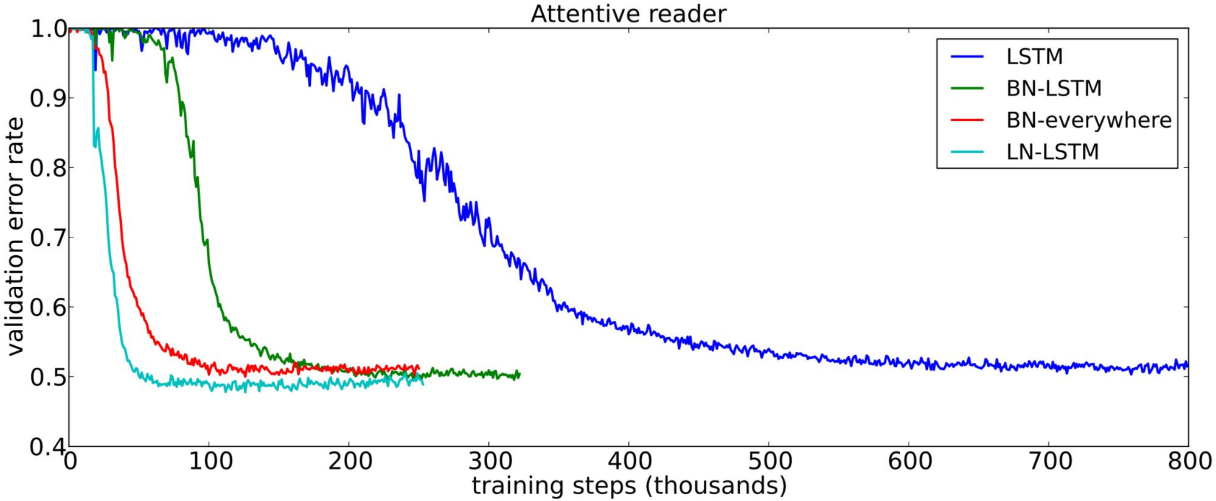
$$\mathbf{h}^t = f\left[\frac{\mathbf{g}}{\sigma^t} \odot \left(\mathbf{a}^t - \mu^t\right) + \mathbf{b}\right] \qquad \mu^t = \frac{1}{H}\sum_{i=1}^{H} a_i^t \qquad \sigma^t = \sqrt{\frac{1}{H}\sum_{i=1}^{H}\left(a_i^t - \mu^t\right)^2}$$



[Layer Normalization, Ba. et al, 2016]

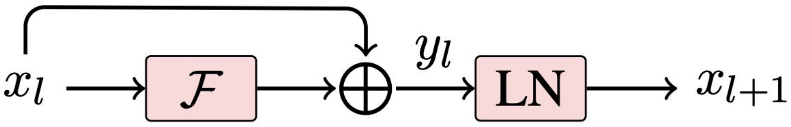
Layer Normalization

$$\mathbf{h}^t = f\left[\frac{\mathbf{g}}{\sigma^t} \odot \left(\mathbf{a}^t - \mu^t\right) + \mathbf{b}\right] \qquad \mu^t = \frac{1}{H}\sum_{i=1}^{H} a_i^t \qquad \sigma^t = \sqrt{\frac{1}{H}\sum_{i=1}^{H}\left(a_i^t - \mu^t\right)^2}$$



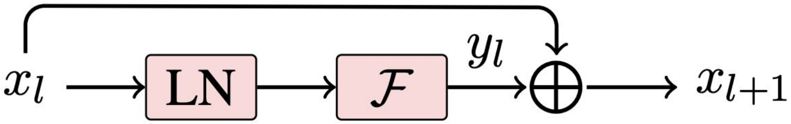
[Layer Normalization, Ba. et al, 2016]

Layer Normalization in Transformers



(a) post-norm residual unit

Modern Transformers



(b) pre-norm residual unit

Figure 1: Examples of pre-norm residual unit and post-norm residual unit. $\mathcal{F}$ = sub-layer, and LN = layer normalization.

[Learning deep transformer models for machine translation, Wang. et al, 20169]

Modern Transformers: Architecture & Training Tricks

Training:

→ Dropout (Srivastava et al., 2014) during at every layer just before adding residual

→ AdamW optimizer with warmup and cosine decay (Loshchilov & Hutter, 2017)

→ Label smoothing (Müller et al, 2019

→ Auto-regressive decoding with beam search and length penalties (Graves, 2012)

→ Checkpoint-averaging (Izmailov et al., 2018)

[Tensor2Tensor Transformers, Kaiser, 2024]

Modern Transformers: Architecture & Training Tricks

Training:

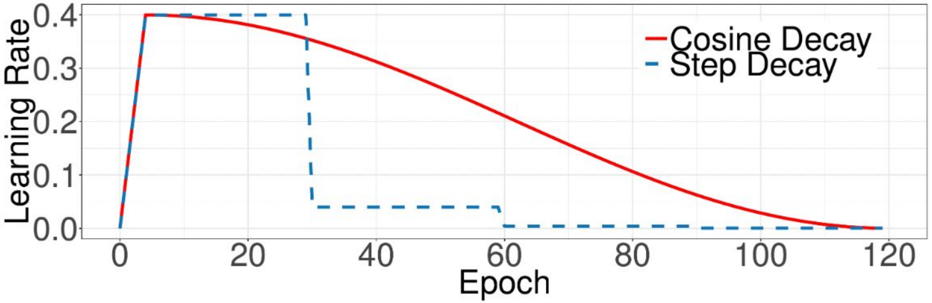
→ Dropout (Srivastava et al., 2014) during at every layer just before adding residual

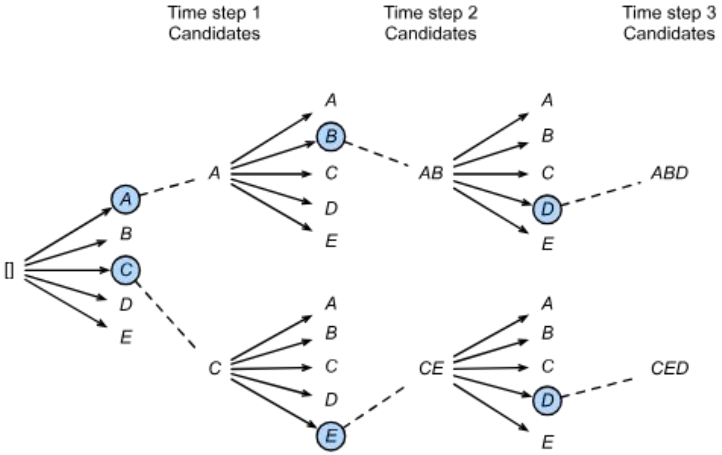
→ AdamW optimizer with warmup and cosine decay (Loshchilov & Hutter, 2017)

→ Label smoothing (Müller et al, 2019

→ Auto-regressive decoding with beam search and length penalties (Graves, 2012)

→ Checkpoint-averaging (Izmailov et al., 2018)





[Tensor2Tensor Transformers, Kaiser, 2024]

Modern Transformers: Architecture & Training Tricks

Architecture:

→ Pre-normalization using RMSNorm (Zhang and Sennrich, 2019)

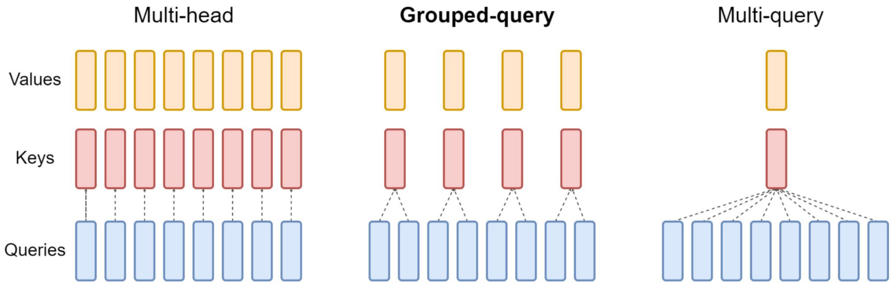
→ SwiGLU activation function (Shazeer, 2020)

→ Rotary positional embeddings (RoPE, Su et al. 2022)

→ Grouped-query attention (GQA, Ainslie et al., 2023)

→ Flash or Ring Attention (Dao et al, 2022; Liu et al, 2023)

→ Mixture of Experts (MoE, Mistral AI., 2023)



[Llama 2: Open Foundation and Fine-Tuned Chat Models, 2023, Tensor2Tensor Transformers, Kaiser, 2024]

Outline for today

Part 1:

- N-Gram Language Models
- Transformers

[break]

Part 2:

- In-Context Learning & Prompting
- Scaling Laws
- Parameter Efficient Fine-Tuning & Quantization
- Capabilities & Limitations

Outline for today

Part 1:

- N-Gram Language Models
- Transformers

[break]

Part 2:

- **In-Context Learning & Prompting**
- Scaling Laws
- Parameter Efficient Fine-Tuning & Quantization
- Capabilities & Limitations

Language Models as Few-Shot Learners (GPT-3)

Prior to GPT-2: Rich literature on so called few-shot learning algorithms

→ Goal: Develop ML methods that can perform well on a novel, unseen task for which we only have a small number of labeled examples.

[Language Models are Few-Shot Learners, Brown et al., 2020,
Matching Networks for One Shot Learning, Vinyals et al., 2017]

Language Models as Few-Shot Learners (GPT-3)

Prior to GPT-2: Rich literature on so called few-shot learning algorithms

→ Goal: Develop ML methods that can perform well on a novel, unseen task for which we only have a small number of labeled examples.

→ Main strategy: Episodic training

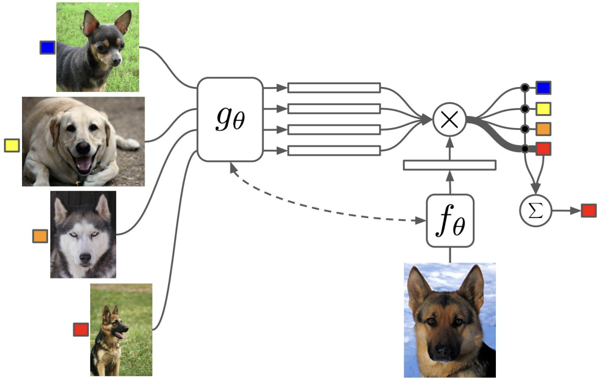


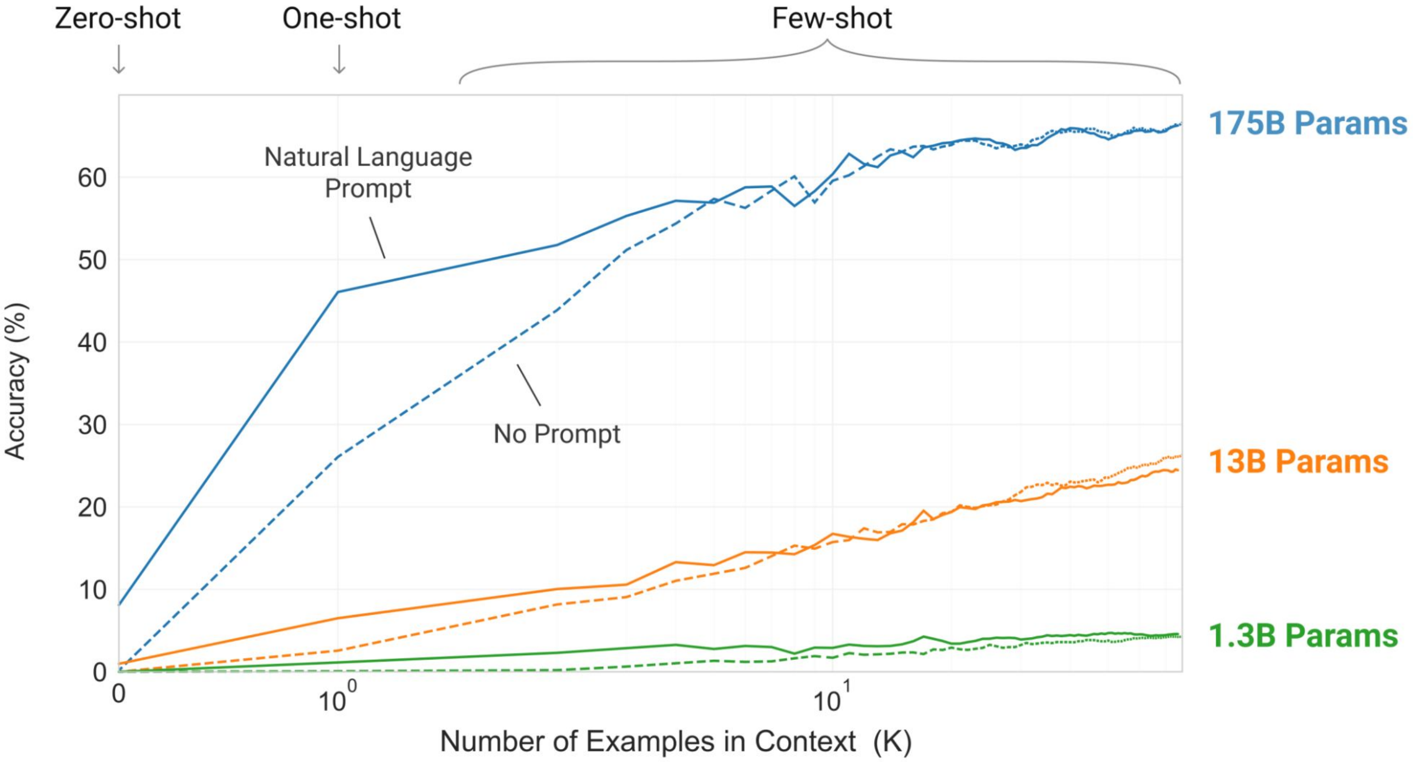
Figure 1: Matching Networks architecture

"our training procedure is based on a simple machine learning principle: test and train conditions must match"

(Vinyals et al., 2017)

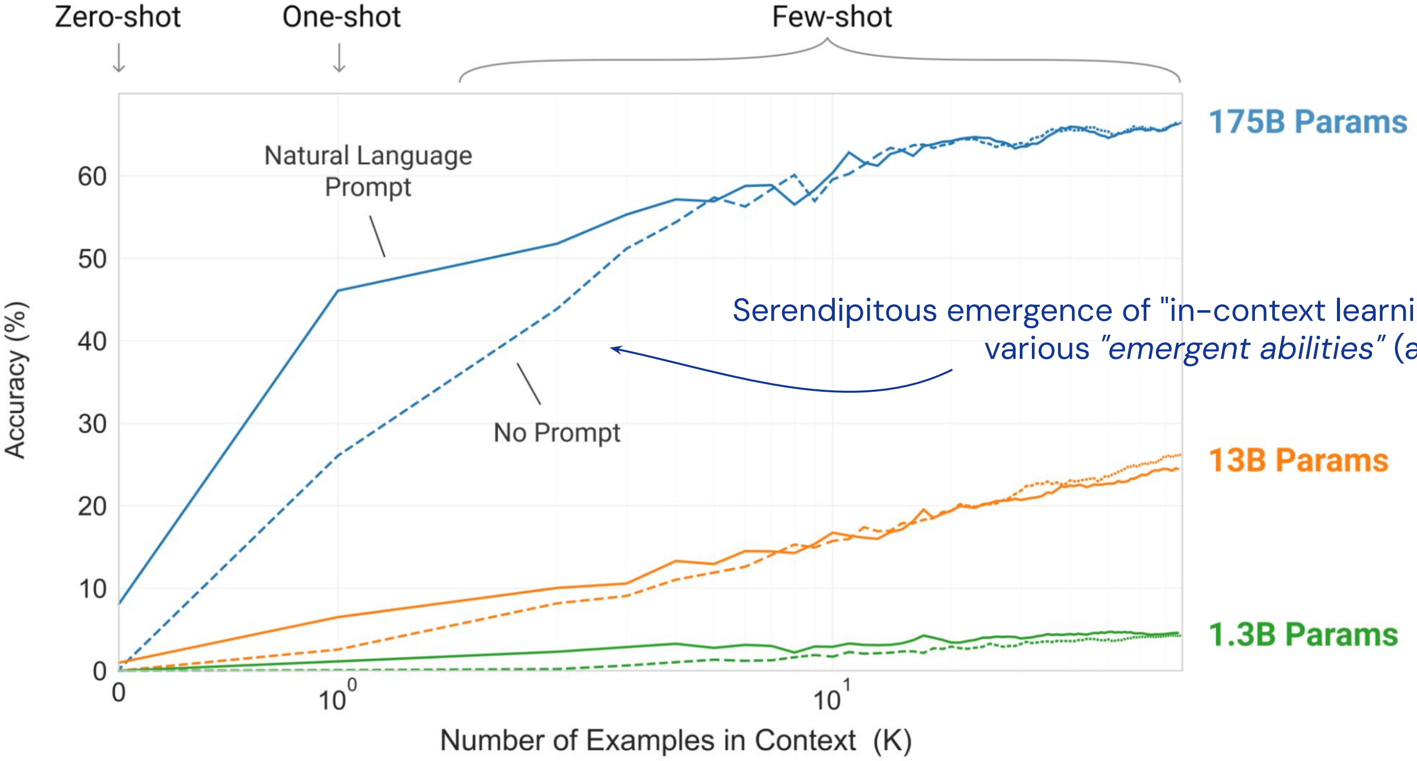
[Language Models are Few-Shot Learners, Brown et al., 2020, Matching Networks for One Shot Learning, Vinyals et al., 2017]

Language Models as Few-Shot Learners (GPT-3)



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

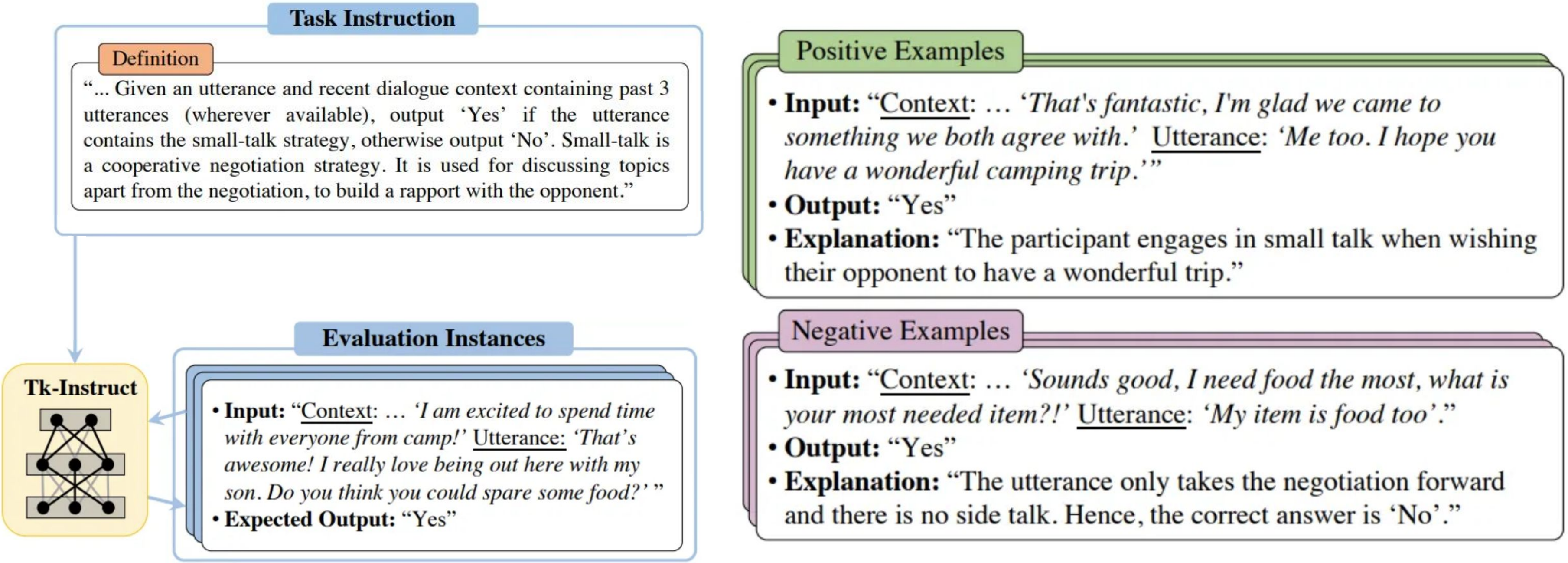
Language Models as Few-Shot Learners (GPT-3)



Serendipitous emergence of "in-context learning", one of various *"emergent abilities"* (and risks)

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

An In-Context Learning example



Task Instruction

Definition

"... Given an utterance and recent dialogue context containing past 3 utterances (wherever available), output 'Yes' if the utterance contains the small-talk strategy, otherwise output 'No'. Small-talk is a cooperative negotiation strategy. It is used for discussing topics apart from the negotiation, to build a rapport with the opponent."

Positive Examples

- **Input:** "Context: … *'That's fantastic, I'm glad we came to something we both agree with.'* Utterance: *'Me too. I hope you have a wonderful camping trip.'*"
- **Output:** "Yes"
- **Explanation:** "The participant engages in small talk when wishing their opponent to have a wonderful trip."

Negative Examples

- **Input:** "Context: … *'Sounds good, I need food the most, what is your most needed item?!'* Utterance: *'My item is food too'*."
- **Output:** "Yes"
- **Explanation:** "The utterance only takes the negotiation forward and there is no side talk. Hence, the correct answer is 'No'."

Evaluation Instances

- **Input:** "Context: … *'I am excited to spend time with everyone from camp!'* Utterance: *'That's awesome! I really love being out here with my son. Do you think you could spare some food?'* "
- **Expected Output:** "Yes"

[Language Models are Few-Shot Learners, Brown et al., 2020, Matching Networks for One Shot Learning, Vinyals et al., 2017]

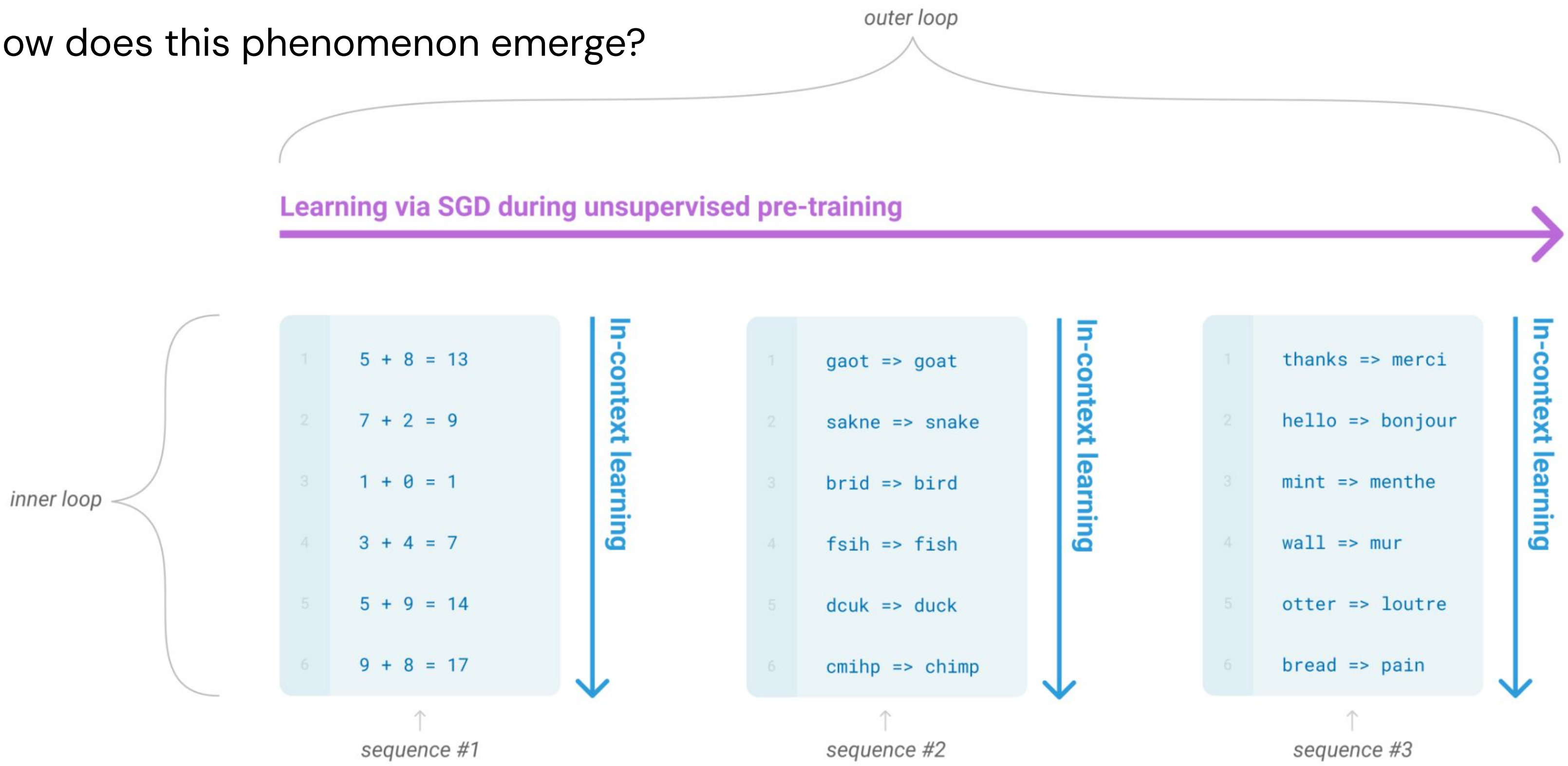
Language Models as Few-Shot Learners (GPT-3)

How does this phenomenon emerge?

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

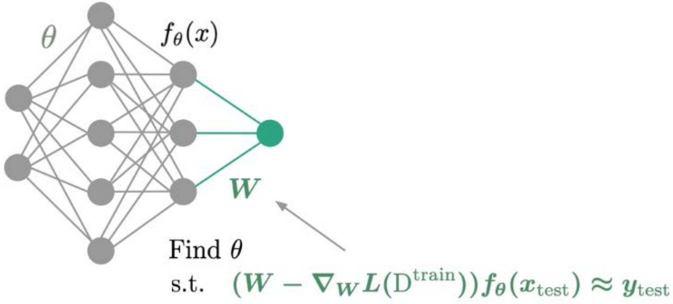
Language Models as Few-Shot Learners (GPT-3)

How does this phenomenon emerge?



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

Understanding In-Context Learning

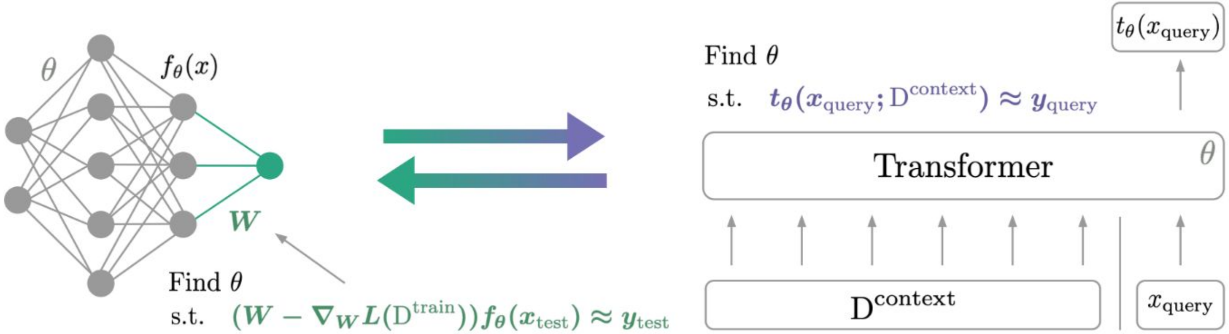


Find θ
s.t. $(W - \nabla_W L(\mathrm{D}^{\text{train}}))f_\theta(x_{\text{test}}) \approx y_{\text{test}}$

Could Self-Attention implement a gradient-based learning algorithm?

[Transformers Learn In-Context by Gradient Descent, von Oswald et al., 2023]

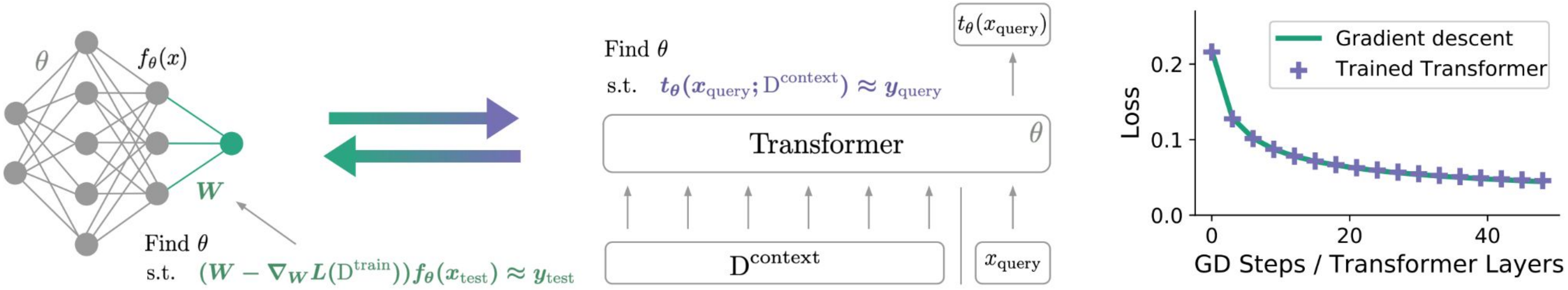
Understanding In-Context Learning


θ
$f_\theta(x)$
W
Find θ
s.t. $(W - \nabla_W L(\mathrm{D}^{\text{train}}))f_\theta(x_{\text{test}}) \approx y_{\text{test}}$
$t_\theta(x_{\text{query}})$
Find θ
s.t. $t_\theta(x_{\text{query}}; \mathrm{D}^{\text{context}}) \approx y_{\text{query}}$
Transformer
θ
$\mathrm{D}^{\text{context}}$
x_{query}

Could Self-Attention implement a gradient-based learning algorithm?

[Transformers Learn In-Context by Gradient Descent, von Oswald et al., 2023]

Understanding In-Context Learning


θ
$f_\theta(x)$
W
Find θ
s.t. $(W - \nabla_W L(D^{\text{train}}))f_\theta(x_{\text{test}}) \approx y_{\text{test}}$
$t_\theta(x_{\text{query}})$
Find θ
s.t. $t_\theta(x_{\text{query}}; D^{\text{context}}) \approx y_{\text{query}}$
Transformer
θ
D^{context}
x_{query}
Gradient descent
Trained Transformer
Loss
0.2
0.1
0.0
0
20
40
GD Steps / Transformer Layers

Could Self-Attention implement a gradient-based learning algorithm?

[Transformers Learn In-Context by Gradient Descent, von Oswald et al., 2023]

Understanding In-Context Learning in Linear Transformers

Squared error for a linear model:

$$L(W) = \frac{1}{2N} \sum_{i=1}^{N} \|Wx_i - y_i\|^2$$

Update with Gradient descent:

$$\Delta W = -\eta \nabla_W L(W) = -\frac{\eta}{N} \sum_{i=1}^{N} (Wx_i - y_i) x_i^T$$

[Transformers Learn In-Context by Gradient Descent, von Oswald et al., 2023]

Understanding In-Context Learning in Linear Transformers

Squared error for a linear model:

$$L(W) = \frac{1}{2N}\sum_{i=1}^{N}\|Wx_i - y_i\|^2$$

Update with Gradient descent:

$$\Delta W = -\eta\nabla_W L(W) = -\frac{\eta}{N}\sum_{i=1}^{N}(Wx_i - y_i)x_i^T$$

New Loss:

$$L(W+\Delta W) = \frac{1}{2N}\sum_{i=1}^{N}\|(W+\Delta W)x_i - y_i\|^2$$

$$= \frac{1}{2N}\sum_{i=1}^{N}\|Wx_i - (y_i - \Delta y_i)\|^2$$

"transformed targets"

[Transformers Learn In-Context by Gradient Descent, von Oswald et al., 2023]

Understanding In-Context Learning in Linear Transformers

Tokens

$$e_j \leftarrow e_j + \mathrm{SA}_\theta(j, \{e_1, \ldots, e_N\})$$

$$= e_j + \sum_h P_h V_h \mathrm{softmax}(K_h^T q_{h,j})$$

$$q_{h,j} = W_{h,Q} e_j$$

(similarly for Values and Keys)

[Transformers Learn In-Context by Gradient Descent, von Oswald et al.., 2023]

Understanding In-Context Learning in Linear Transformers

Tokens

$$e_j \leftarrow e_j + \mathrm{SA}_\theta(j, \{e_1, \ldots, e_N\})$$
$$= e_j + \sum_h P_h V_h \mathrm{softmax}(K_h^T q_{h,j})$$

$$q_{h,j} = W_{h,Q} e_j$$

(similarly for Values and Keys)

Let's consider a linear Transformer:

$$e_j \leftarrow e_j + \mathrm{LSA}_\theta(j, \{e_1, \ldots, e_N\}) = e_j + \sum_h P_h V_h K_h^T q_{h,j}$$

[Transformers Learn In-Context by Gradient Descent, von Oswald et al., 2023]

Understanding In-Context Learning in Linear Transformers

Tokens

$$e_j \leftarrow e_j + \mathbf{SA}_\theta(j, \{e_1, \ldots, e_N\})$$
$$= e_j + \sum_h P_h V_h \mathrm{softmax}(K_h^T q_{h,j})$$

$$q_{h,j} = W_{h,Q} e_j$$

(similarly for Values and Keys)

Let's consider a linear Transformer:

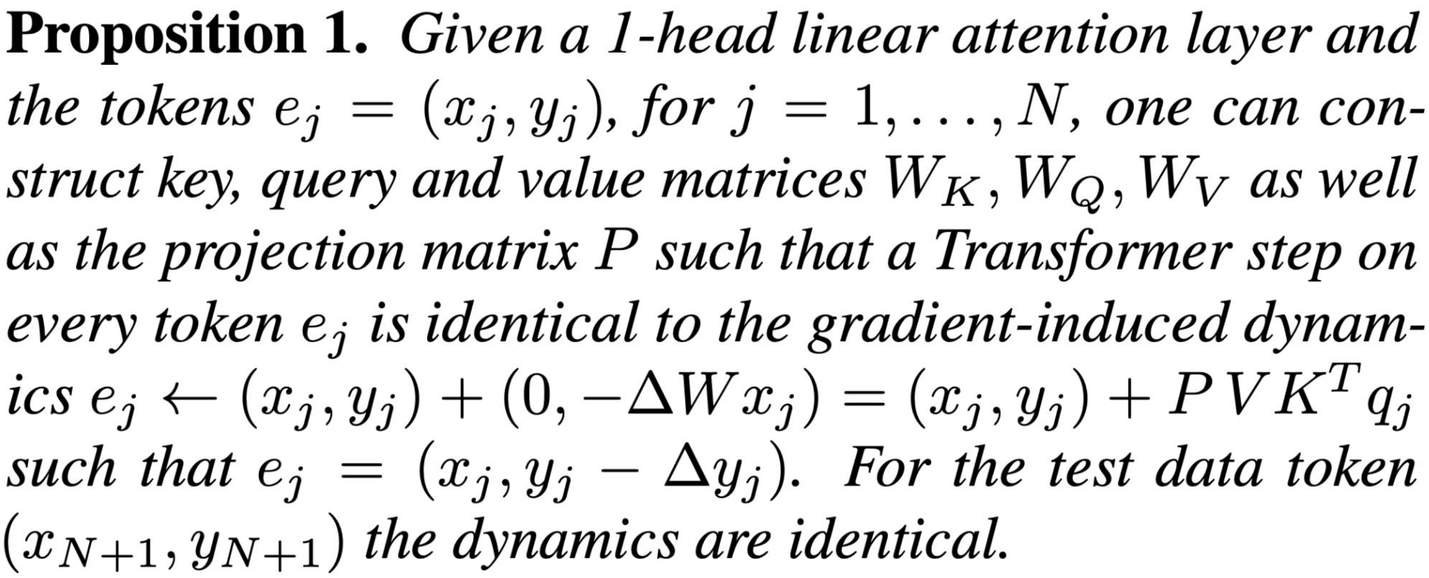
$$e_j \leftarrow e_j + \mathbf{LSA}_\theta(j, \{e_1, \ldots, e_N\}) = e_j + \textstyle\sum_h P_h V_h K_h^T q_{h,j}$$

Let's set up an in-context regression problem:

$$e_j = (x_j, y_j) \in \mathbb{R}^{N_x + N_y}$$
$$e_{N+1} = (x_{N+1}, y_{N+1}) = (x_{\text{test}}, \hat{y}_{\text{test}}) = e_{\text{test}}$$

[Transformers Learn In-Context by Gradient Descent, von Oswald et al., 2023]

Understanding In-Context Learning in Linear Transformers



Proposition 1. *Given a 1-head linear attention layer and the tokens* $e_j = (x_j, y_j)$, *for* $j = 1, \ldots, N$, *one can construct key, query and value matrices* W_K, W_Q, W_V *as well as the projection matrix* P *such that a Transformer step on every token* e_j *is identical to the gradient-induced dynamics* $e_j \leftarrow (x_j, y_j) + (0, -\Delta W x_j) = (x_j, y_j) + PVK^T q_j$ *such that* $e_j = (x_j, y_j - \Delta y_j)$. *For the test data token* (x_{N+1}, y_{N+1}) *the dynamics are identical.*

Details in the paper

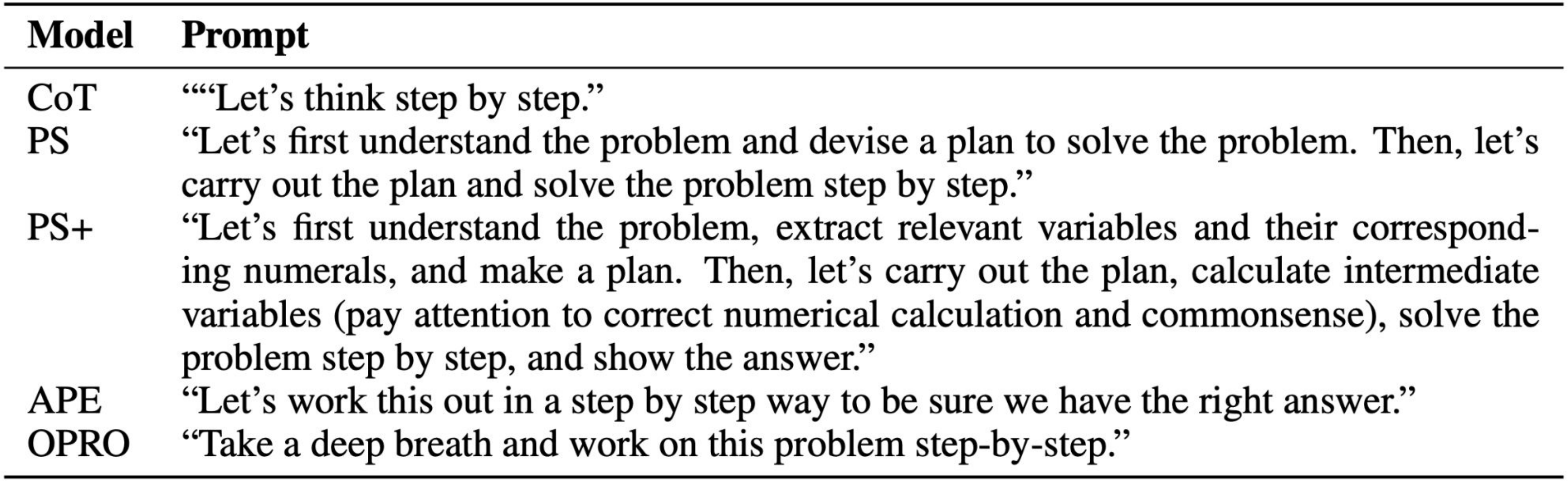
[Transformers Learn In-Context by Gradient Descent, von Oswald et al., 2023]

Prompting

In practice, it turns out that in-context learning is extremely sensitive to the way prompts are phrased:

Prompting

In practice, it turns out that in-context learning is extremely sensitive to the way prompts are phrased. These all give very different results using the same data:

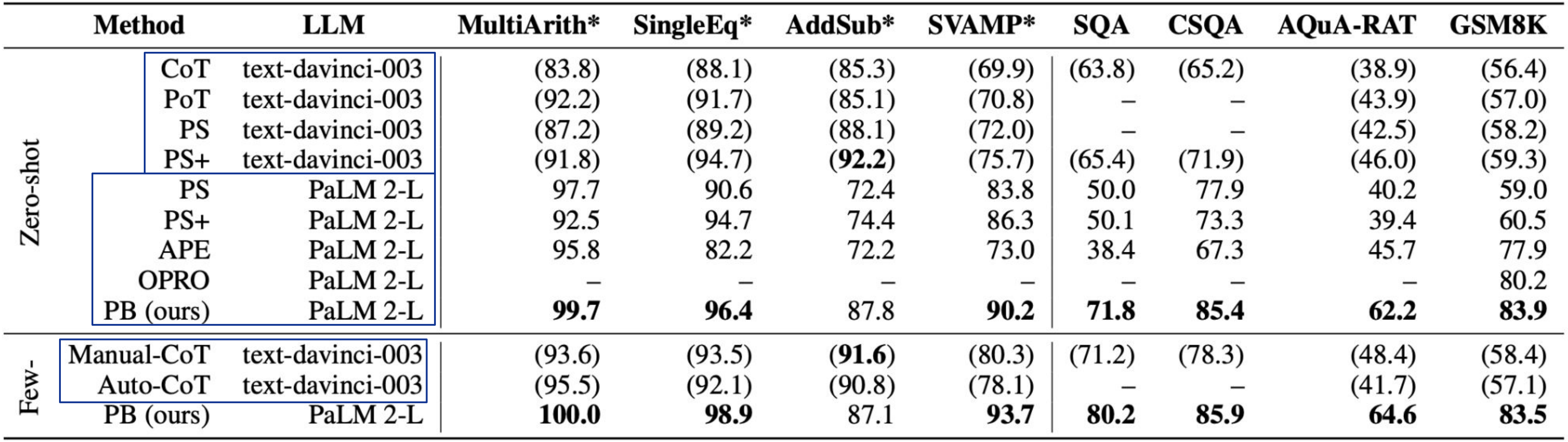


Model	**Prompt**
CoT	““Let’s think step by step.”
PS	“Let’s first understand the problem and devise a plan to solve the problem. Then, let’s carry out the plan and solve the problem step by step.”
PS+	“Let’s first understand the problem, extract relevant variables and their corresponding numerals, and make a plan. Then, let’s carry out the plan, calculate intermediate variables (pay attention to correct numerical calculation and commonsense), solve the problem step by step, and show the answer.”
APE	“Let’s work this out in a step by step way to be sure we have the right answer.”
OPRO	“Take a deep breath and work on this problem step-by-step.”

[PromptBreeder, Fernando et al., 2023]

Prompting

In practice, it turns out that in-context learning is extremely sensitive to the way prompts are phrased:



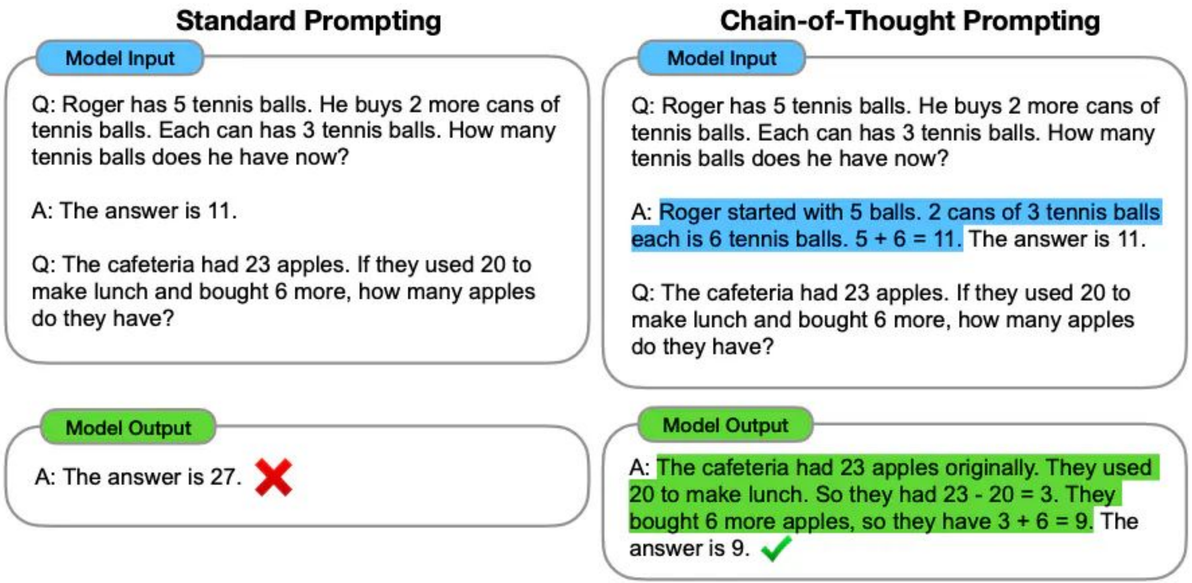
	Method	LLM	MultiArith*	SingleEq*	AddSub*	SVAMP*	SQA	CSQA	AQuA-RAT	GSM8K
Zero-shot	CoT	text-davinci-003	(83.8)	(88.1)	(85.3)	(69.9)	(63.8)	(65.2)	(38.9)	(56.4)
	PoT	text-davinci-003	(92.2)	(91.7)	(85.1)	(70.8)	–	–	(43.9)	(57.0)
	PS	text-davinci-003	(87.2)	(89.2)	(88.1)	(72.0)	–	–	(42.5)	(58.2)
	PS+	text-davinci-003	(91.8)	(94.7)	**(92.2)**	(75.7)	(65.4)	(71.9)	(46.0)	(59.3)
	PS	PaLM 2-L	97.7	90.6	72.4	83.8	50.0	77.9	40.2	59.0
	PS+	PaLM 2-L	92.5	94.7	74.4	86.3	50.1	73.3	39.4	60.5
	APE	PaLM 2-L	95.8	82.2	72.2	73.0	38.4	67.3	45.7	77.9
	OPRO	PaLM 2-L	–	–	–	–	–	–	–	80.2
	PB (ours)	PaLM 2-L	**99.7**	**96.4**	87.8	**90.2**	**71.8**	**85.4**	**62.2**	**83.9**
Few-	Manual-CoT	text-davinci-003	(93.6)	(93.5)	**(91.6)**	(80.3)	(71.2)	(78.3)	(48.4)	(58.4)
	Auto-CoT	text-davinci-003	(95.5)	(92.1)	(90.8)	(78.1)	–	–	(41.7)	(57.1)
	PB (ours)	PaLM 2-L	**100.0**	**98.9**	87.1	**93.7**	**80.2**	**85.9**	**64.6**	**83.5**

Same data, same model

[PromptBreeder, Fernando et al., 2023]

Chain-of-Thought Prompting

In practice, it turns out that in-context learning is extremely sensitive to the way prompts are phrased. One common trick that almost always works in adding explanations:



Standard Prompting

Model Input

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: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The answer is 27. ✗

Chain-of-Thought Prompting

Model Input

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: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The cafeteria had 23 apples originally. They used 20 to make lunch. So they had 23 - 20 = 3. They bought 6 more apples, so they have 3 + 6 = 9. The answer is 9. ✓

These can be even auto-generated (using another LLM!)

[Chain-of-Thought Prompting Elicits Reasoning in Large Language Models, Wei et al., 2022]

Outline for today

Part 1:

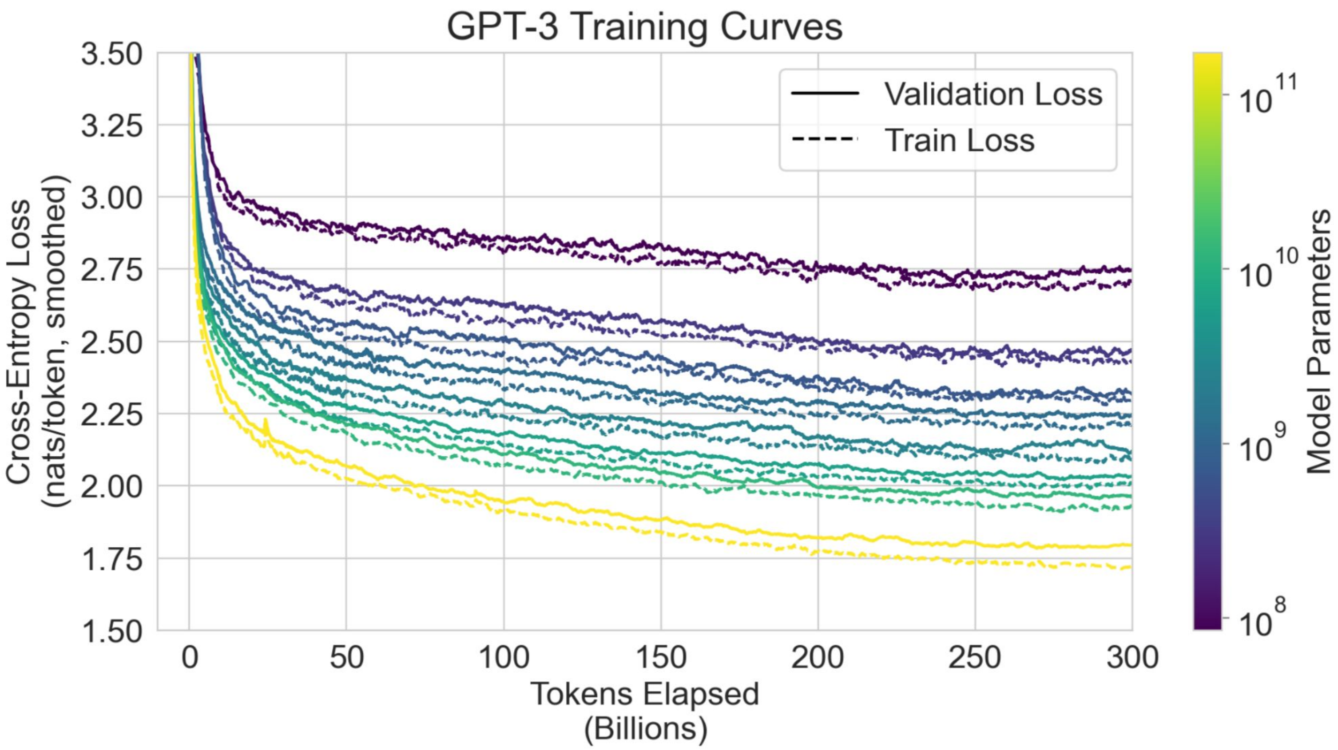
- N-Gram Language Models
- Transformers

[break]

Part 2:

- In-Context Learning & Prompting
- **Scaling Laws**
- Parameter Efficient Fine-Tuning & Quantization
- Capabilities & Limitations

Can we predict the expected improvement?



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

Training Compute Optimal Models

Number of datapoints

$$N_{opt}(C), D_{opt}(C) = \underset{N,D \text{ s.t. } \mathrm{FLOPs}(N,D)=C}{\operatorname{argmin}} L(N, D)$$

Number of parameters

FLOPS budget

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

Scaling Laws from Classical Risk Decomposition

$$N_{opt}(C), D_{opt}(C) = \underset{N,D \text{ s.t. } \mathrm{FLOPs}(N,D)=C}{\operatorname{argmin}} L(N, D)$$

Number of data seen so far

Estimated loss

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

Number of parameters

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

Scaling Laws from Classical Risk Decomposition

$$N_{opt}(C), D_{opt}(C) = \underset{N,D \text{ s.t. FLOPs}(N,D)=C}{\operatorname{argmin}} L(N, D)$$

Loss for an ideal generative process (entropy of natural text)

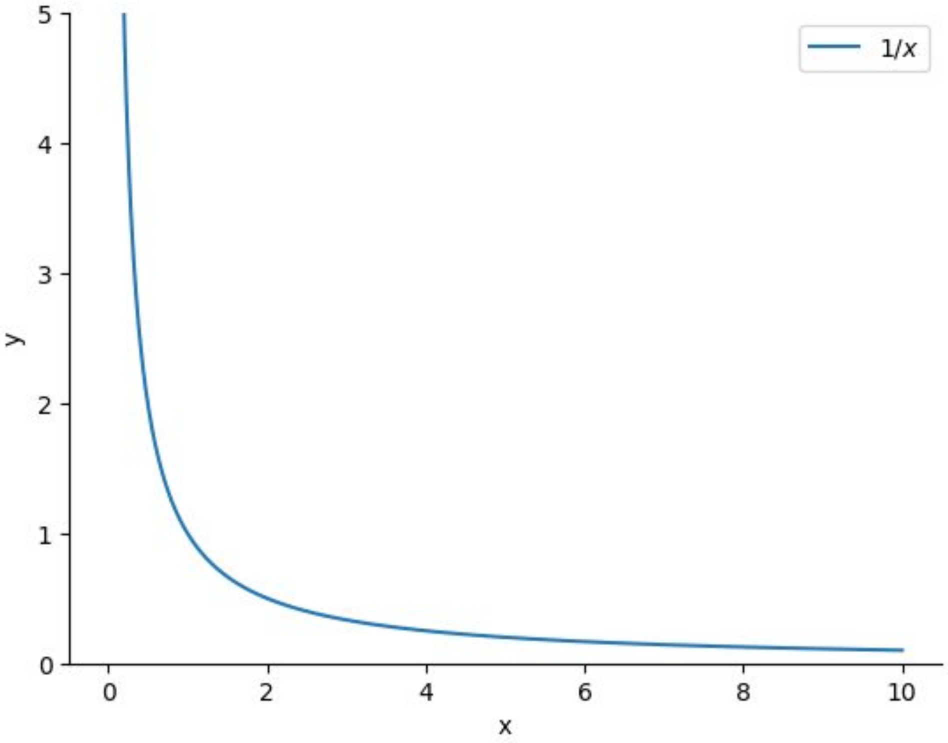
$$\hat{L}(N, D) \triangleq \boxed{E} + \boxed{\frac{A}{N^{\alpha}}} + \boxed{\frac{B}{D^{\beta}}}$$

Perfectly trained transformer with N parameters underperforms the ideal setting

Transformer is not trained to convergence, as we only make a finite number of optimisation steps, on a sample of the dataset distribution

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

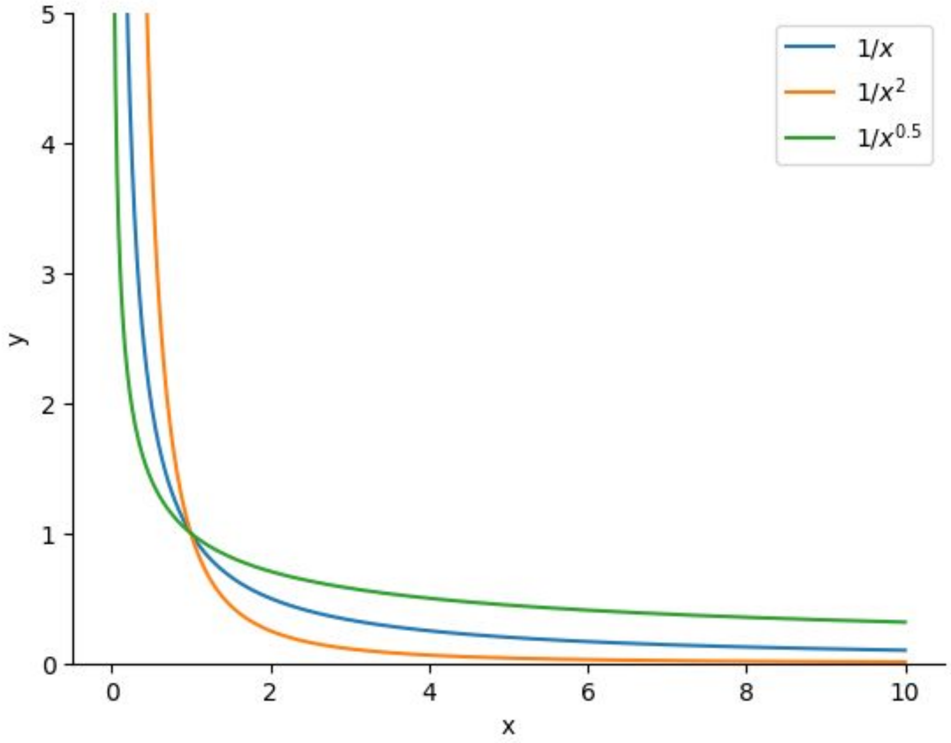
Scaling Laws from Classical Risk Decomposition



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

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

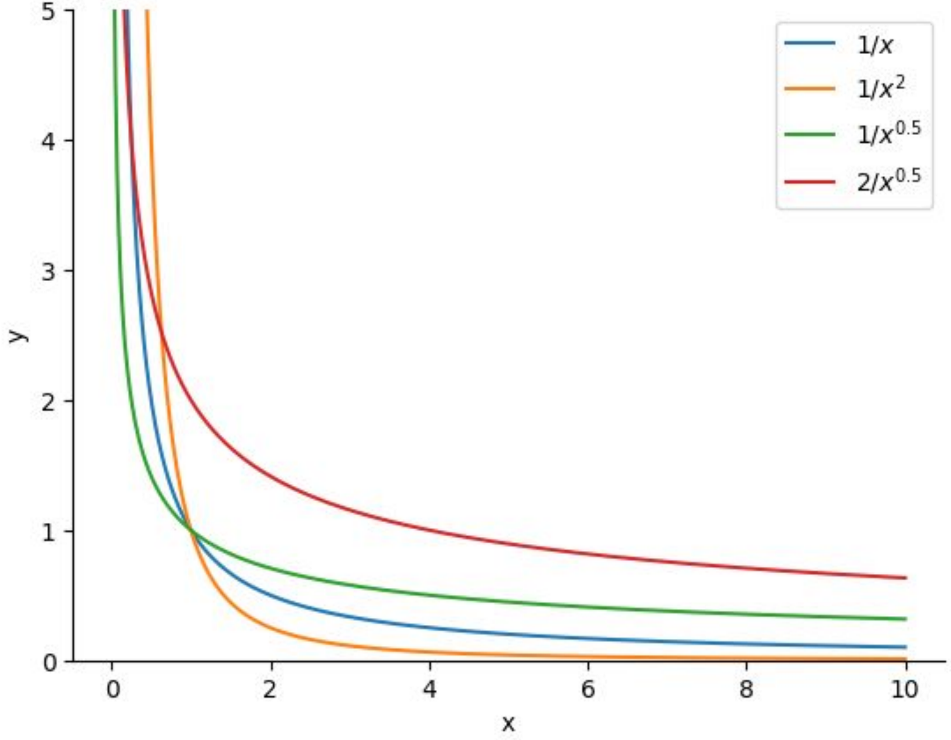
Scaling Laws from Classical Risk Decomposition



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

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

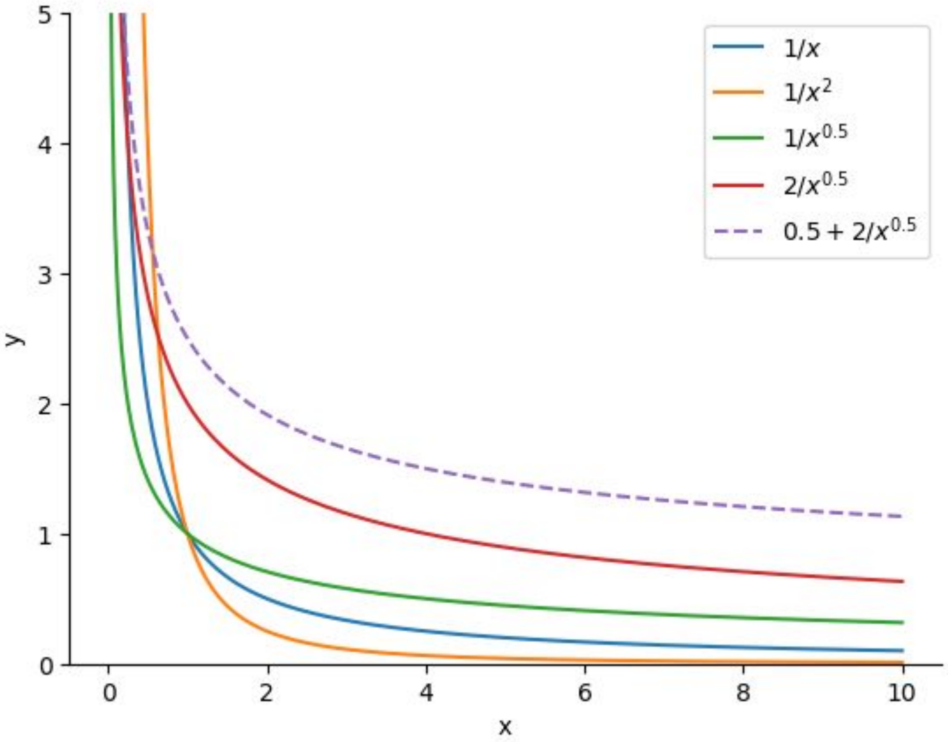
Scaling Laws from Classical Risk Decomposition



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

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

Scaling Laws from Classical Risk Decomposition



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

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

Scaling Laws from Classical Risk Decomposition

Loss for an ideal generative process (entropy of natural text)

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

Perfectly trained transformer with N parameters underperforms the ideal setting

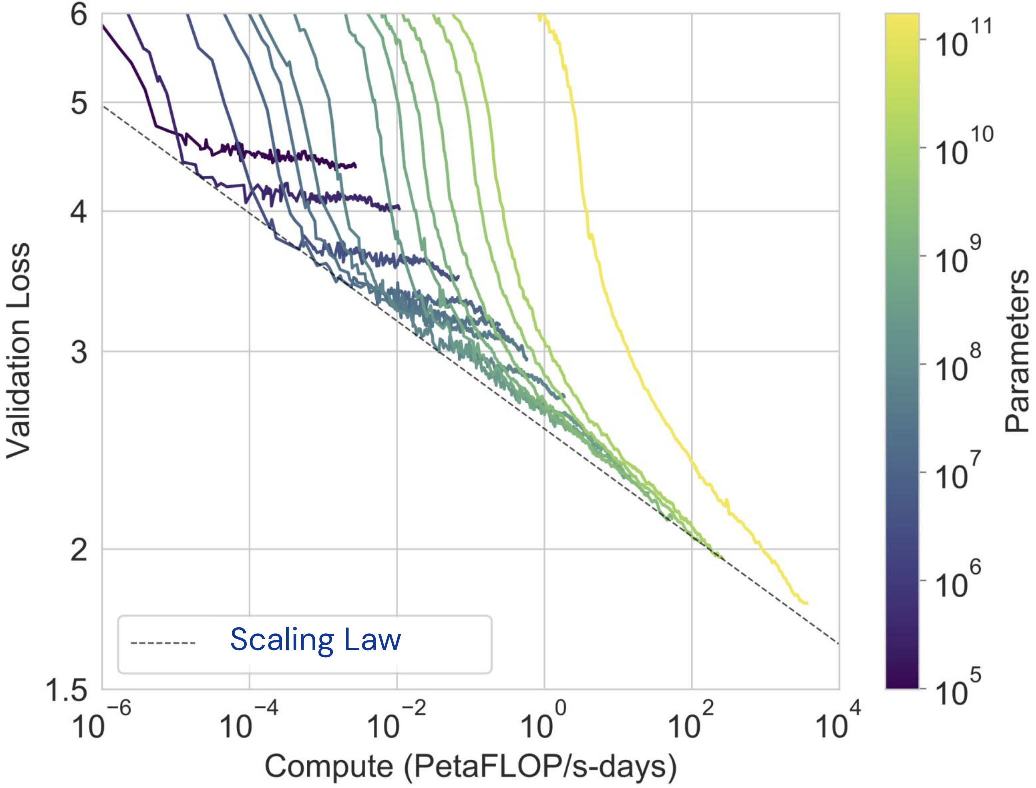
Transformer is not trained to convergence, as we only make a finite number of optimisation steps, on a sample of the dataset distribution

Parameters are learned from existing training curves:

$$\min_{A,B,E,\alpha,\beta} \sum_{\text{Runs } i} \text{Huber}_{\delta}\left(\log \hat{L}(N_i, D_i) - \log L_i\right)$$

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

Can we predict the expected improvement?



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

Outline for today

Part 1:

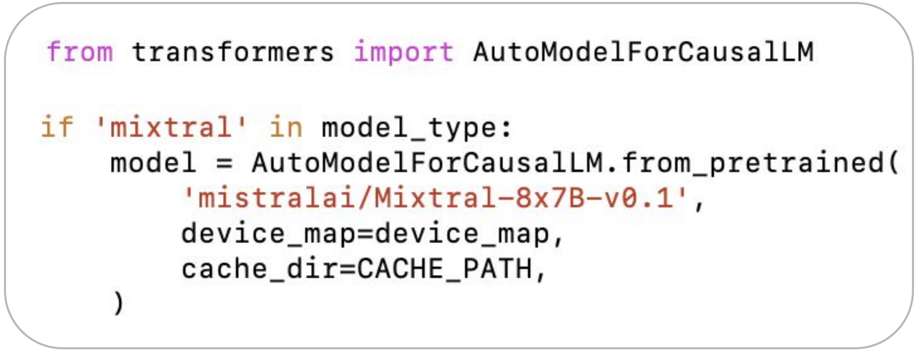
- N-Gram Language Models
- Transformers

[break]

Part 2:

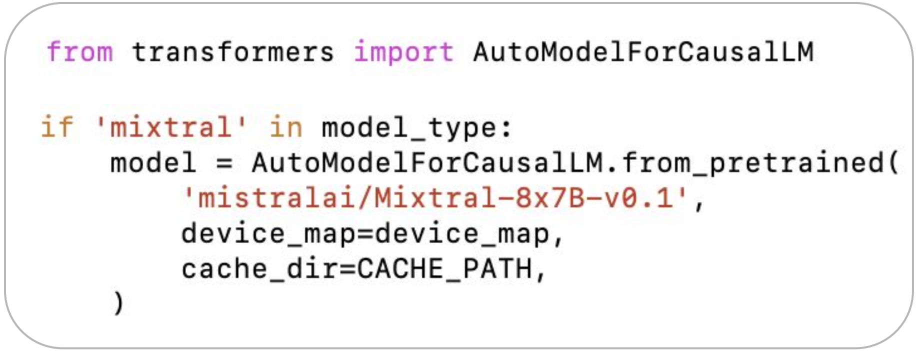
- In-Context Learning & Prompting
- Scaling Laws
- **Parameter Efficient Fine-Tuning & Quantization**
- Capabilities & Limitations

Using these models in practice:

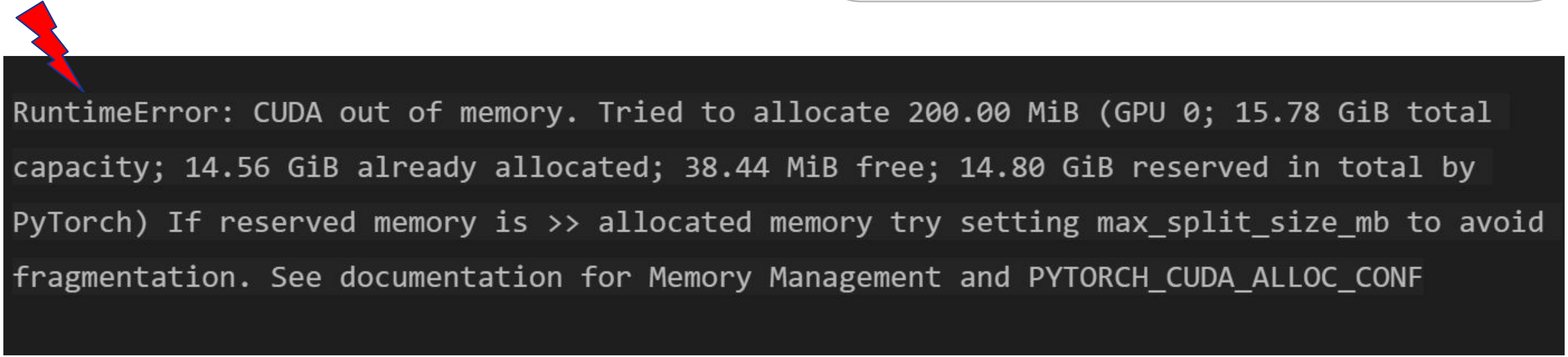


```python
from transformers import AutoModelForCausalLM

if 'mixtral' in model_type:
    model = AutoModelForCausalLM.from_pretrained(
        'mistralai/Mixtral-8x7B-v0.1',
        device_map=device_map,
        cache_dir=CACHE_PATH,
    )
```

Using these models in practice:



```python
from transformers import AutoModelForCausalLM

if 'mixtral' in model_type:
    model = AutoModelForCausalLM.from_pretrained(
        'mistralai/Mixtral-8x7B-v0.1',
        device_map=device_map,
        cache_dir=CACHE_PATH,
    )
```



```
RuntimeError: CUDA out of memory. Tried to allocate 200.00 MiB (GPU 0; 15.78 GiB total
capacity; 14.56 GiB already allocated; 38.44 MiB free; 14.80 GiB reserved in total by
PyTorch) If reserved memory is >> allocated memory try setting max_split_size_mb to avoid
fragmentation. See documentation for Memory Management and PYTORCH_CUDA_ALLOC_CONF
```

(Post-training) Quantization

$$\operatorname{argmin}_{\boxed{\widehat{\mathbf{W}}}} ||\mathbf{W}\mathbf{X} - \widehat{\mathbf{W}}\mathbf{X}||_2^2$$

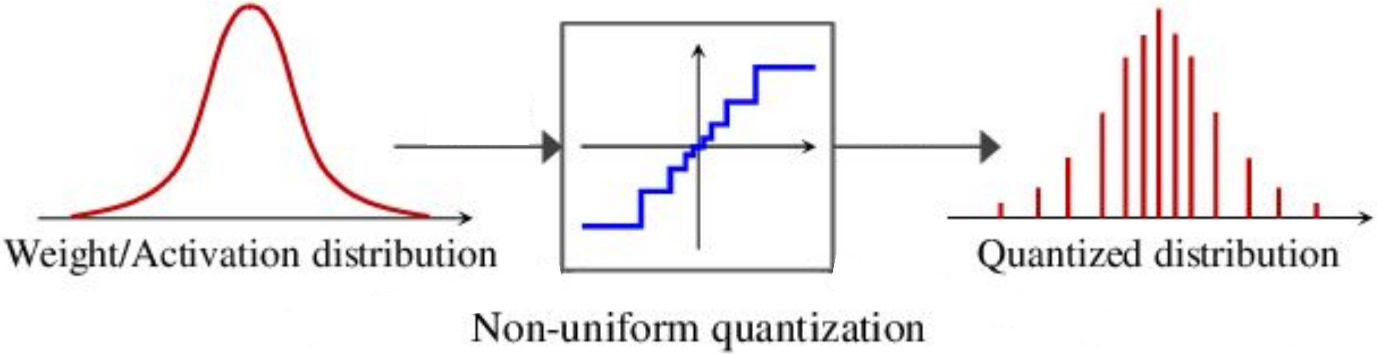
quantized + low-precision matrix multiplication

[GPTQ, Frantar et al., 2023, LLM.int8(), Dettmers et al., 2022]

(Post-training) Quantization

$$\operatorname{argmin}_{\boxed{\widehat{\mathbf{W}}}} ||\mathbf{W}\mathbf{X} - \widehat{\mathbf{W}}\mathbf{X}||_2^2$$

quantized + low-precision matrix multiplication



[GPTQ, Frantar et al., 2023, LLM.int8(), Dettmers et al., 2022, Image Source]

(Post-training) Quantization

$$\operatorname{argmin}_{\boxed{\widehat{\mathbf{W}}}} ||\mathbf{W}\mathbf{X} - \widehat{\mathbf{W}}\mathbf{X}||_2^2$$

quantized + low-precision matrix multiplication

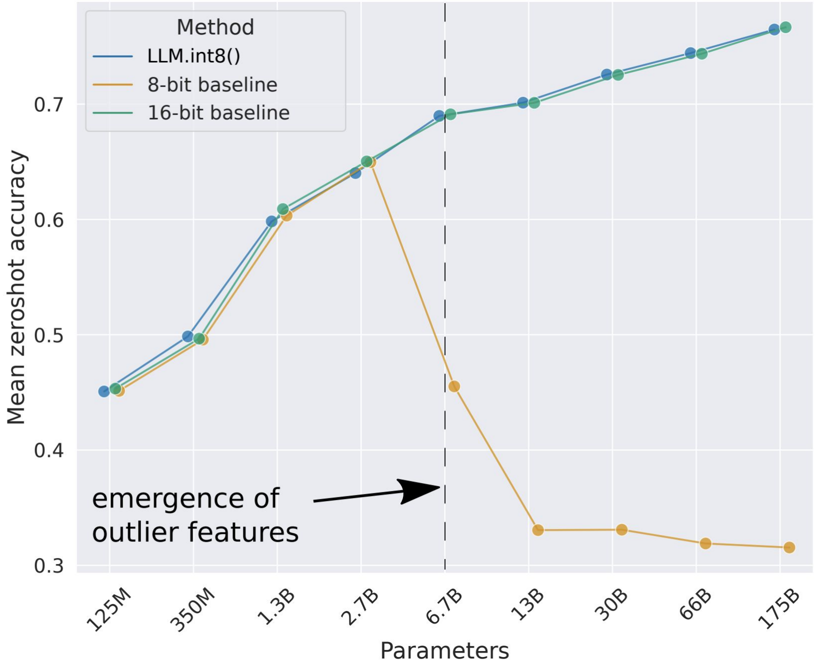
Specifically, GPTQ can quantize GPT models with 175 billion parameters in approximately four GPU hours, reducing the bitwidth down to 3 or 4 bits per weight, with negligible accuracy degradation relative to the uncompressed baseline. Our method more than doubles the compression gains relative to previously-proposed one-shot quantization methods, preserving accuracy, allowing us for the first time to execute an 175 billion-parameter model inside a single GPU for generative inference.

[GPTQ, Frantar et al., 2023, LLM.int8(), Dettmers et al., 2022]

(Post-training) Quantization

$$\mathrm{argmin}_{\boxed{\widehat{\mathbf{W}}}} \|\mathbf{W}\mathbf{X} - \widehat{\mathbf{W}}\mathbf{X}\|_2^2$$

quantized + low-precision matrix multiplication

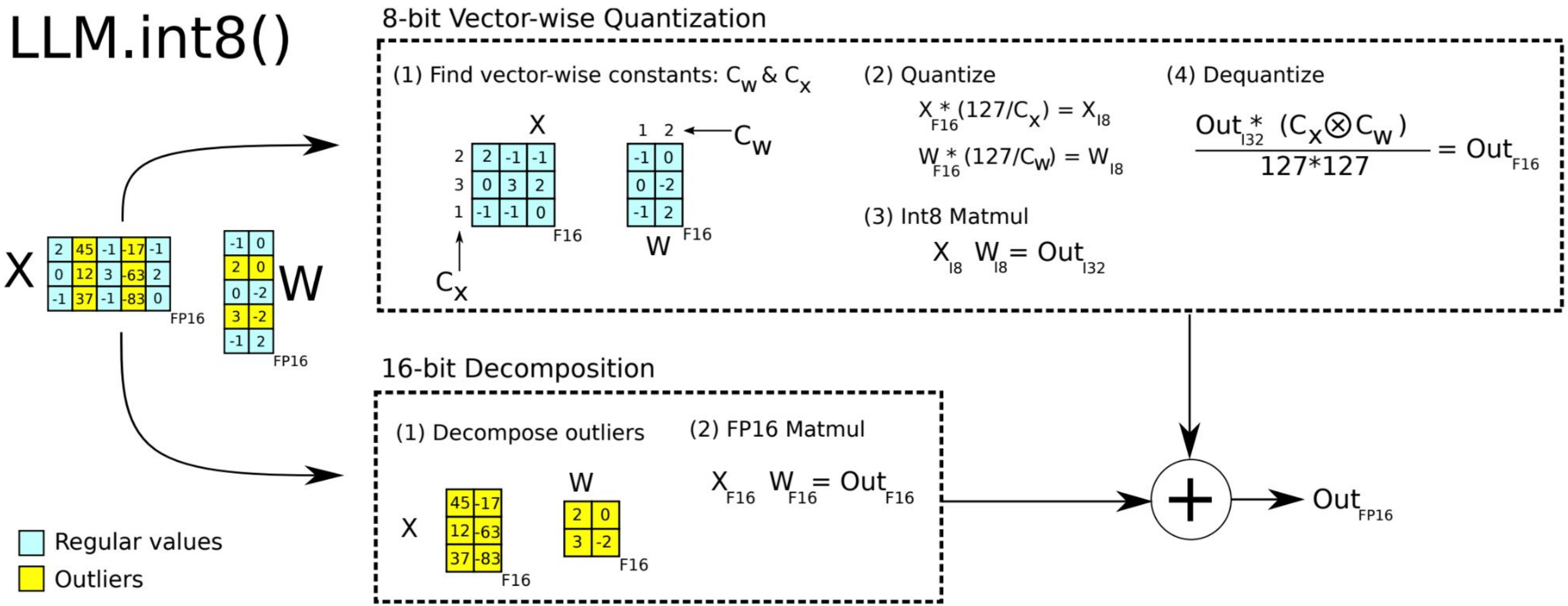


[GPTQ, Frantar et al., 2023, LLM.int8(), Dettmers et al., 2022]

(Post-training) Quantization

$$\arg\min_{\boxed{\widehat{\mathbf{W}}}} ||\mathbf{W}\mathbf{X} - \widehat{\mathbf{W}}\mathbf{X}||_2^2$$

quantized + low-precision matrix multiplication

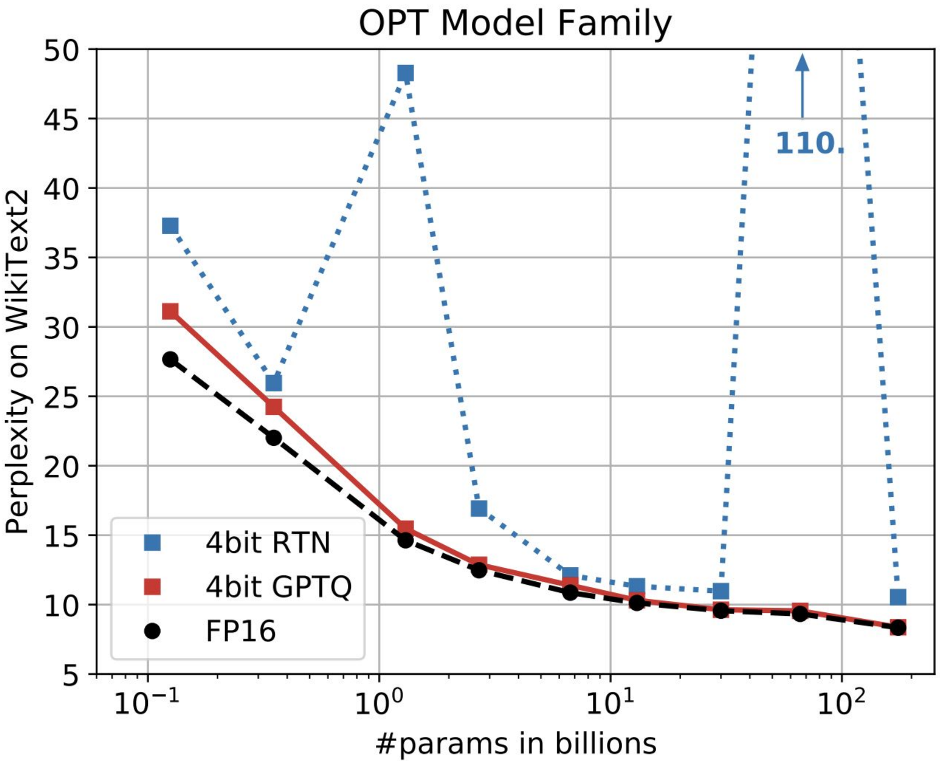


[GPTQ, Frantar et al., 2023, LLM.int8(), Dettmers et al., 2022]

(Post-training) Quantization

$$\mathrm{argmin}_{\boxed{\widehat{\mathbf{W}}}} \|\mathbf{W}\mathbf{X} - \widehat{\mathbf{W}}\mathbf{X}\|_2^2$$

quantized + low-precision matrix multiplication



[GPTQ, Frantar et al., 2023, LLM.int8(), Dettmers et al., 2022]

Parameter-Efficient Fine-Tuning (PEFT)

Common Workflow:

(1) Download state-of-the-art pre-trained LLM trained on internet text for general "world knowledge" and reasoning abilities

(2) Test in-context learning abilities for sufficient few-shot performance.

[Parameter-Efficient Fine-Tuning Methods, A Review, 2023]

Parameter-Efficient Fine-Tuning (PEFT)

Common Workflow:

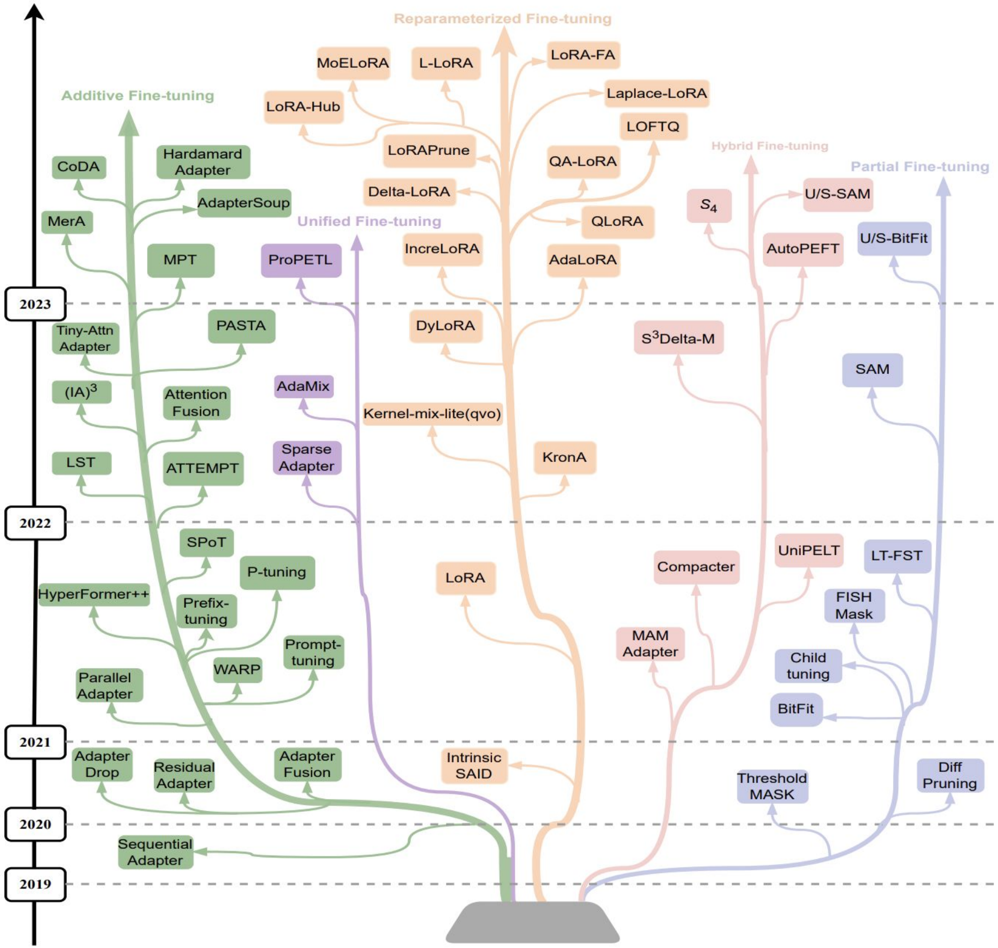
(1) Download state-of-the-art pre-trained LLM trained on internet text for general "world knowledge" and reasoning abilities

(2) Test in-context learning abilities for sufficient few-shot performance.

(3) Use a parameter-efficient fine-tuning scheme to update a subset of parameters (~1%).

[Parameter-Efficient Fine-Tuning Methods, A Review, 2023]

PEFT

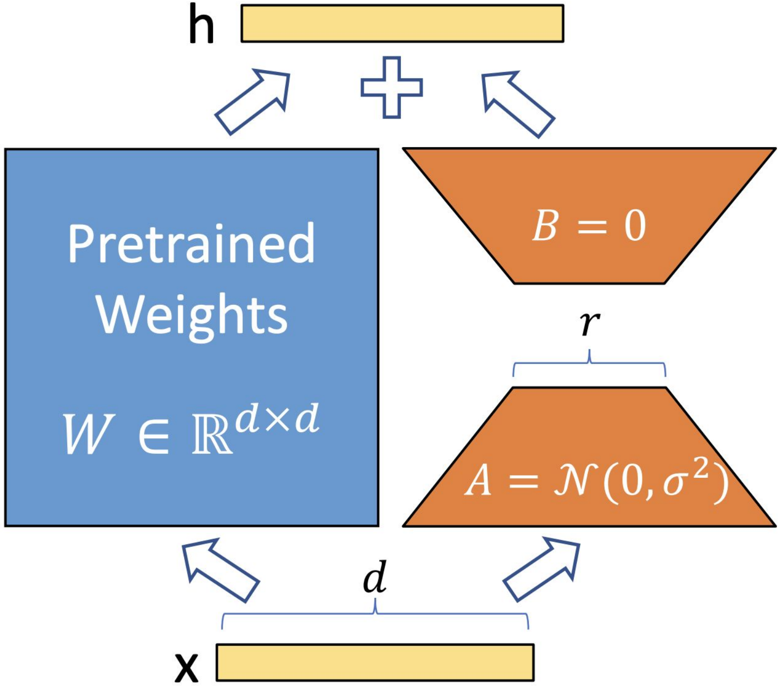


[Parameter-Efficient Fine-Tuning Methods, A Review, 2023]

Low-Rank Adaptation (LoRA)

$$h = W_0 x + \Delta W x = W_0 x + \underbrace{B}_{}A x$$

Zero at initialization


h
Pretrained Weights
$W \in \mathbb{R}^{d \times d}$
$B = 0$
r
$A = \mathcal{N}(0, \sigma^2)$
d
x

[LoRA: Low-Rank Adaptation of Large Language Models, Hu et al., 2021]

Low-Rank Adaptation (LoRA)

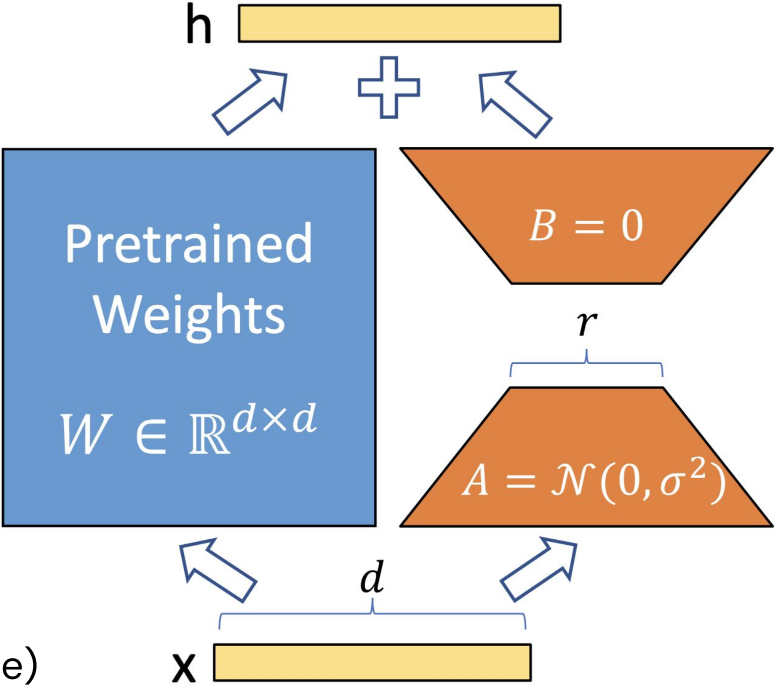
$$h = W_0 x + \Delta W x = W_0 x + \overset{\uparrow}{B} A x$$

Zero at initialization

→ Can easily train and share different LoRA modules for various tasks, only need to store B, A

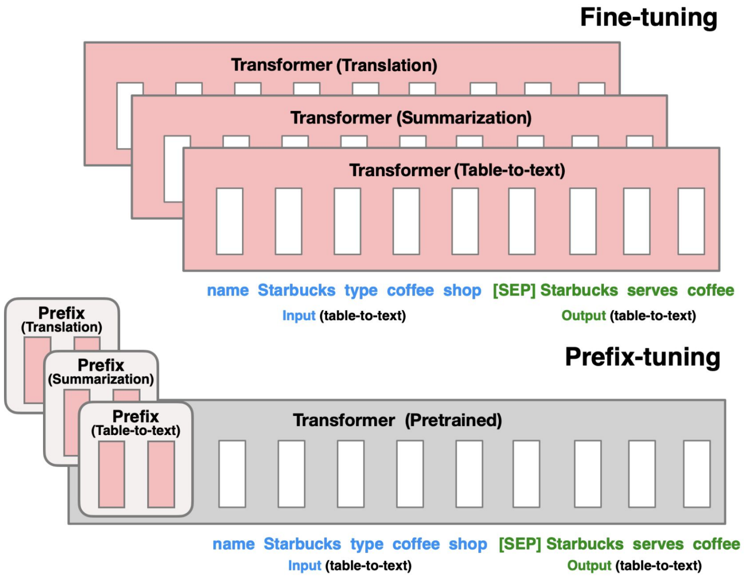
→ No changes to inference speed

→ Efficient Training (No need to calculate and store gradients for full model, no need to store optimizer state)


h
Pretrained Weights
$W \in \mathbb{R}^{d \times d}$
$B = 0$
r
$A = \mathcal{N}(0, \sigma^2)$
d
x

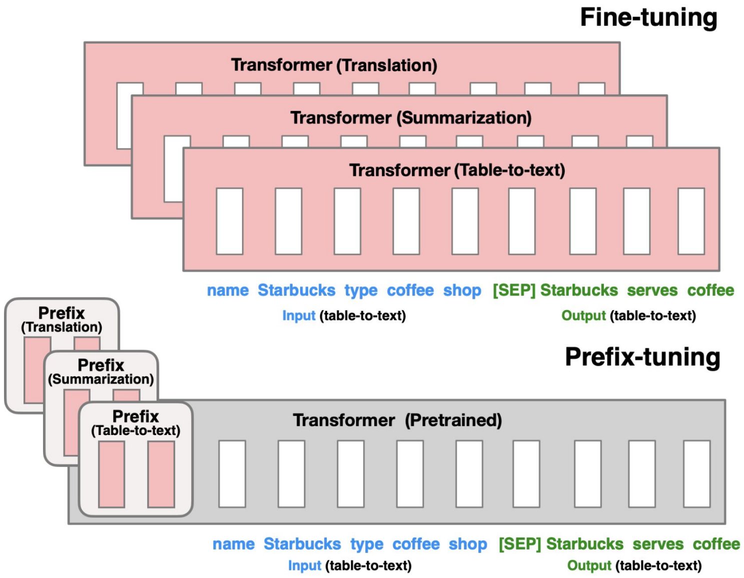
[LoRA: Low-Rank Adaptation of Large Language Models, Hu et al., 2021]

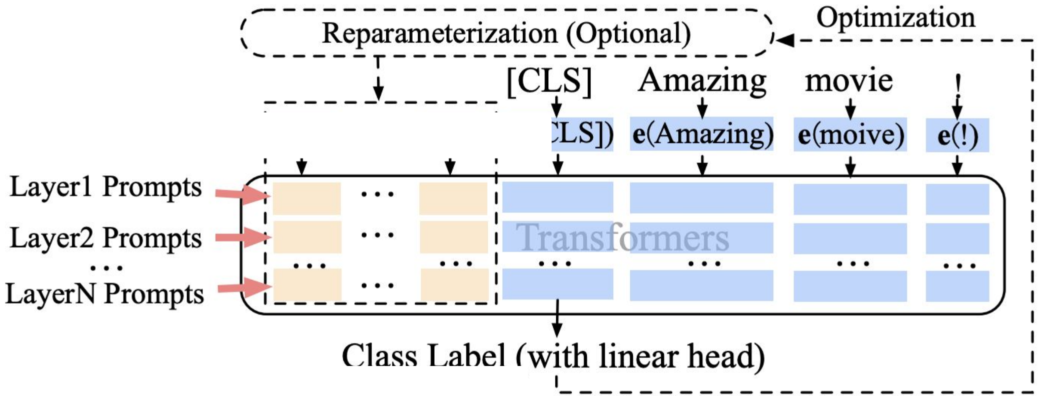
Prefix Tuning v1 & v2


Fine-tuning
Transformer (Translation)
Transformer (Summarization)
Transformer (Table-to-text)
name Starbucks type coffee shop [SEP] Starbucks serves coffee
Input (table-to-text)
Output (table-to-text)
Prefix-tuning
Prefix (Translation)
Prefix (Summarization)
Prefix (Table-to-text)
Transformer (Pretrained)
name Starbucks type coffee shop [SEP] Starbucks serves coffee
Input (table-to-text)
Output (table-to-text)

[Prefix-Tuning: Optimizing Continuous Prompts for Generation, Li & Liang, 2021, P-Tuning v2, Liu et al., 2022]

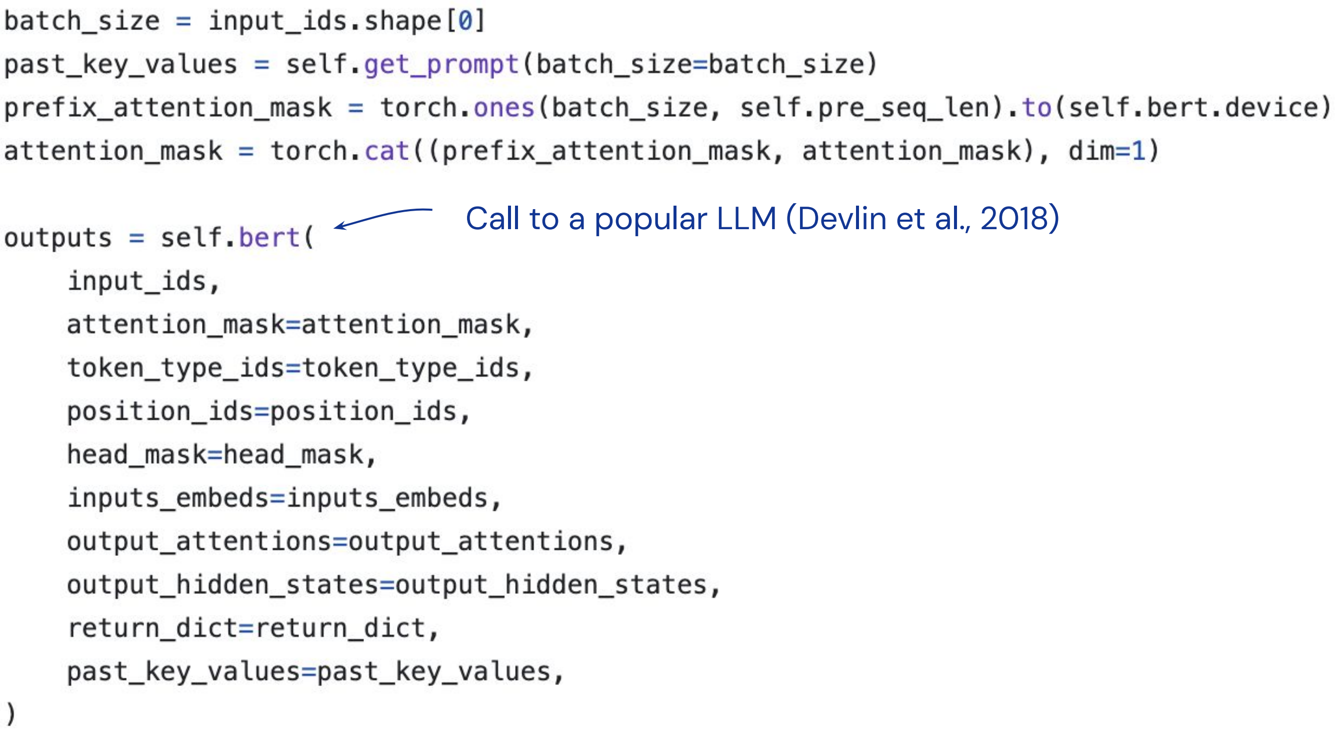
Prefix Tuning v1 & v2


Fine-tuning
Transformer (Translation)
Transformer (Summarization)
Transformer (Table-to-text)
name Starbucks type coffee shop [SEP] Starbucks serves coffee
Input (table-to-text)
Output (table-to-text)
Prefix (Translation)
Prefix (Summarization)
Prefix (Table-to-text)
Prefix-tuning
Transformer (Pretrained)
name Starbucks type coffee shop [SEP] Starbucks serves coffee
Input (table-to-text)
Output (table-to-text)



[Prefix-Tuning: Optimizing Continuous Prompts for Generation, Li & Liang, 2021,
P-Tuning v2, Liu et al., 2022]

Implementing Prefix Tuning v2

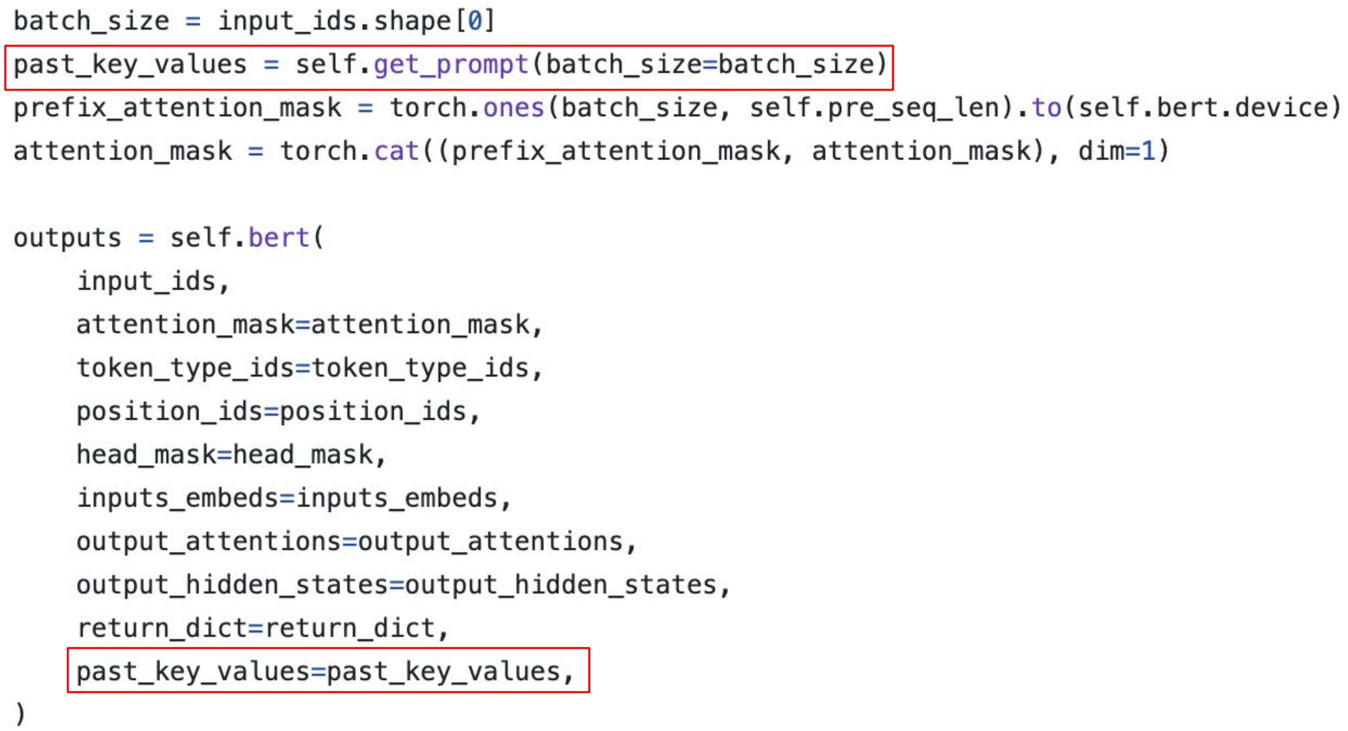


```
batch_size = input_ids.shape[0]
past_key_values = self.get_prompt(batch_size=batch_size)
prefix_attention_mask = torch.ones(batch_size, self.pre_seq_len).to(self.bert.device)
attention_mask = torch.cat((prefix_attention_mask, attention_mask), dim=1)

outputs = self.bert(
    input_ids,
    attention_mask=attention_mask,
    token_type_ids=token_type_ids,
    position_ids=position_ids,
    head_mask=head_mask,
    inputs_embeds=inputs_embeds,
    output_attentions=output_attentions,
    output_hidden_states=output_hidden_states,
    return_dict=return_dict,
    past_key_values=past_key_values,
)
```

Call to a popular LLM (Devlin et al., 2018)

[Prefix-Tuning: Optimizing Continuous Prompts for Generation, Li & Liang, 2021,
P-Tuning v2, Liu et al., 2022]

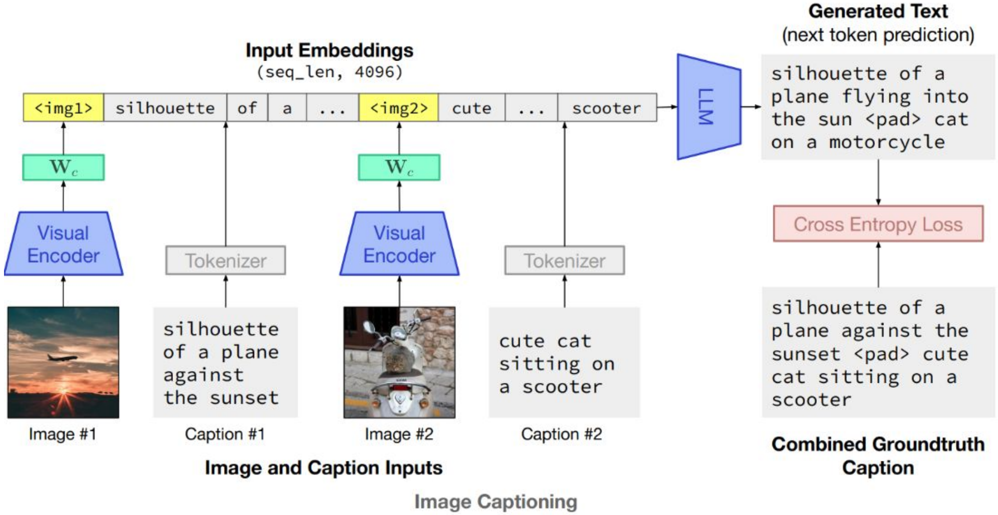
Implementing Prefix Tuning v2



```
batch_size = input_ids.shape[0]
past_key_values = self.get_prompt(batch_size=batch_size)
prefix_attention_mask = torch.ones(batch_size, self.pre_seq_len).to(self.bert.device)
attention_mask = torch.cat((prefix_attention_mask, attention_mask), dim=1)

outputs = self.bert(
    input_ids,
    attention_mask=attention_mask,
    token_type_ids=token_type_ids,
    position_ids=position_ids,
    head_mask=head_mask,
    inputs_embeds=inputs_embeds,
    output_attentions=output_attentions,
    output_hidden_states=output_hidden_states,
    return_dict=return_dict,
    past_key_values=past_key_values,
)
```

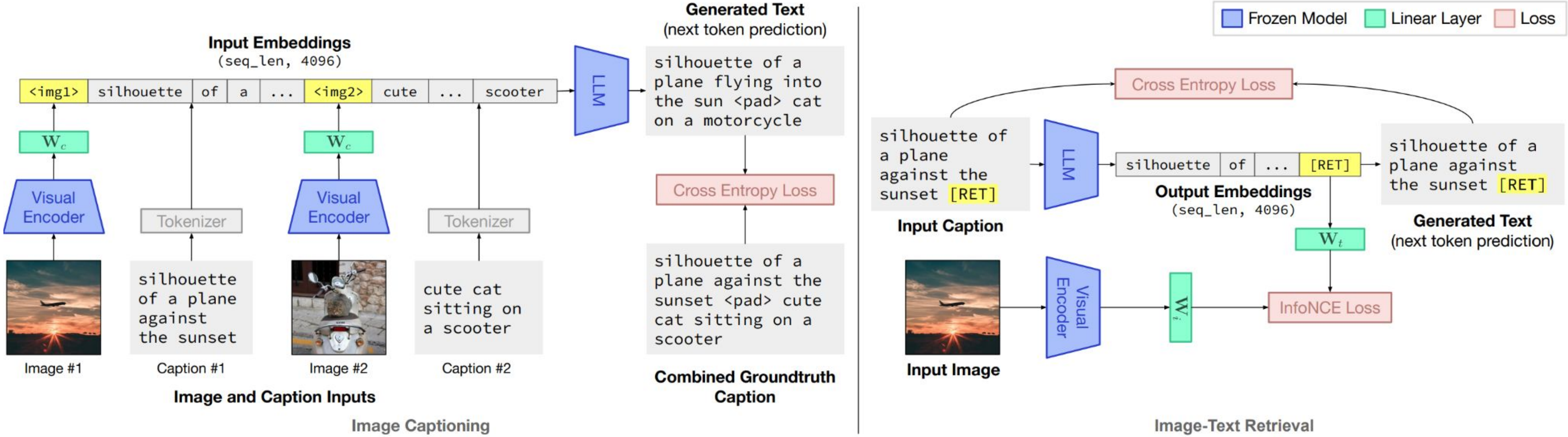
[Prefix-Tuning: Optimizing Continuous Prompts for Generation, Li & Liang, 2021,
P-Tuning v2, Liu et al., 2022]

Prefix Tuning for Multi-Modal models



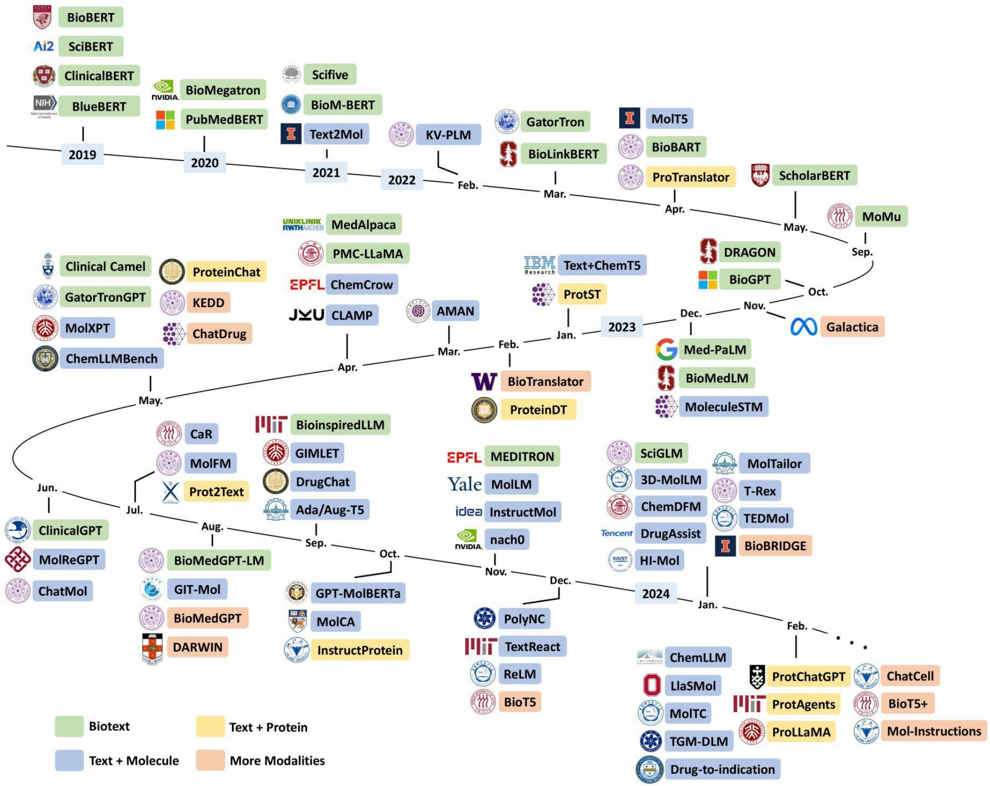
[Grounding Language Models to Images for Multimodal Inputs and Outputs, Ko et al., 2023]

Prefix Tuning for Multi-Modal models

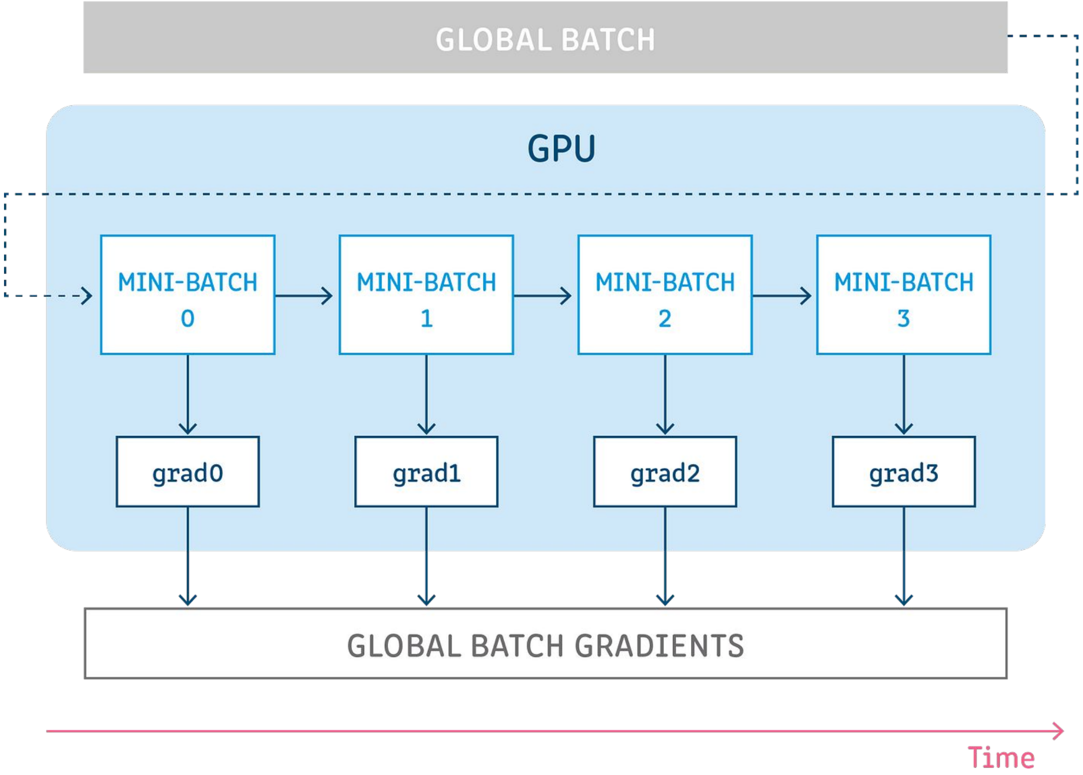


[Grounding Language Models to Images for Multimodal Inputs and Outputs, Ko et al., 2023]

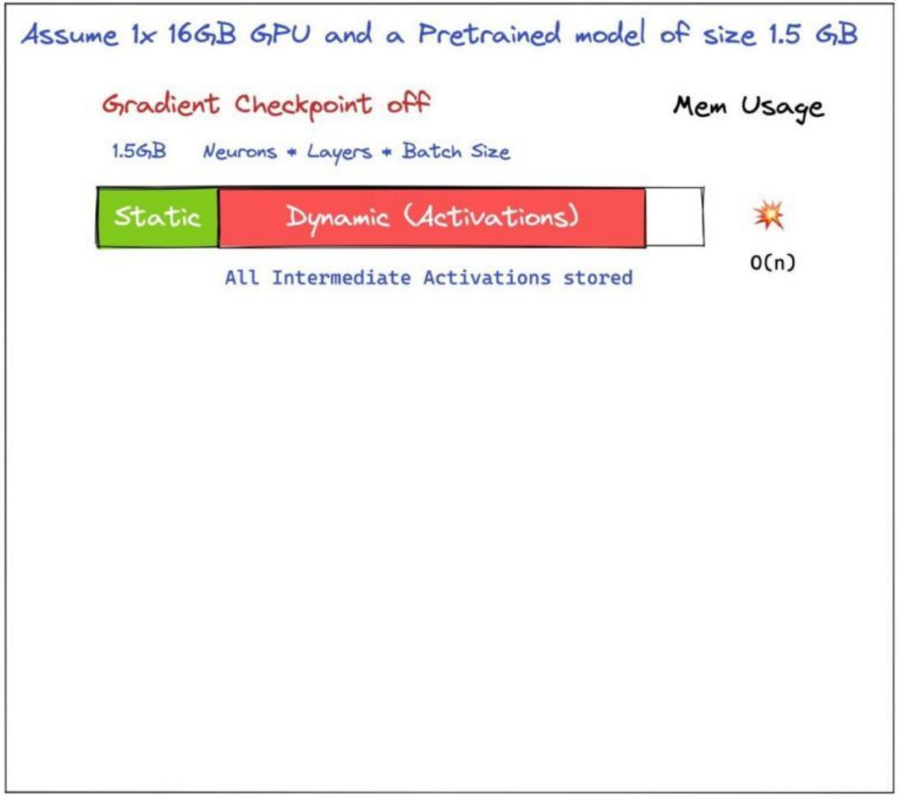
Prefix Tuning for Multi-Modal models



Other Tricks: Gradient accumulation

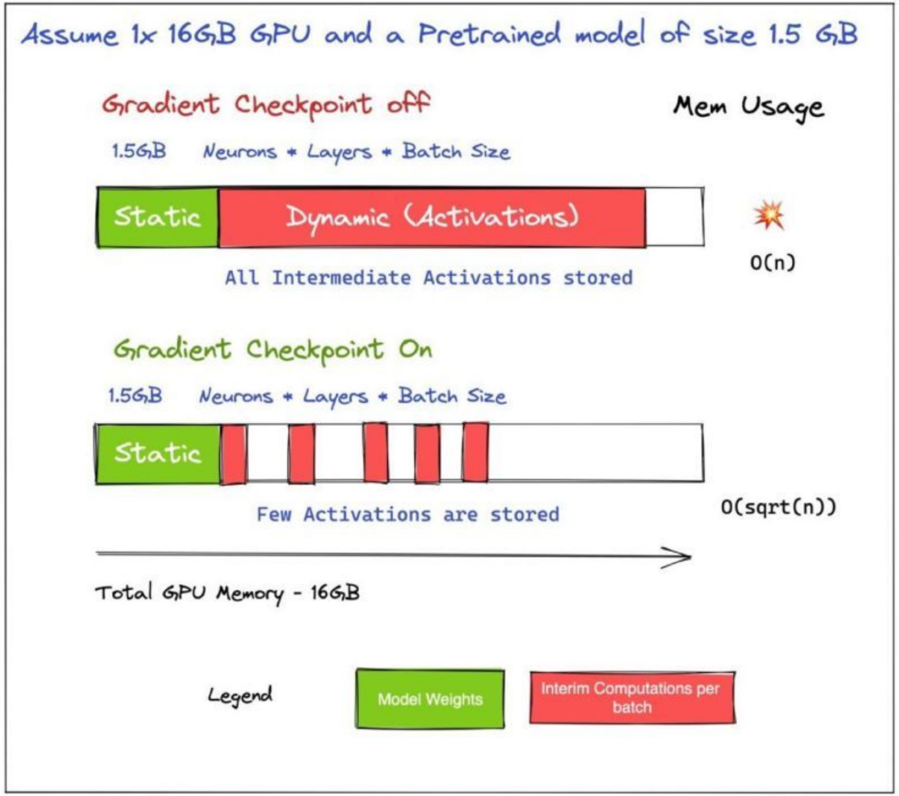


Other Tricks: Gradient checkpointing


Assume 1x 16GB GPU and a Pretrained model of size 1.5 GB
Gradient Checkpoint off
Mem Usage
1.5GB
Neurons * Layers * Batch Size
Static
Dynamic (Activations)
O(n)
All Intermediate Activations stored

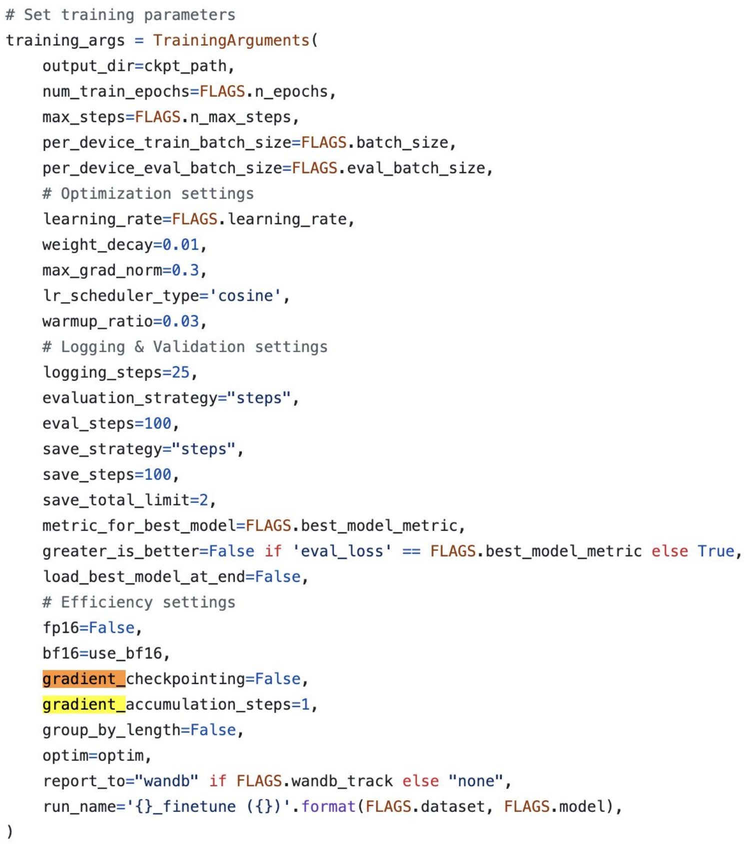
[Training Deep Nets with Sublinear Memory Cost, Chen et al., 2016]

Other Tricks: Gradient checkpointing


Assume 1x 16GB GPU and a Pretrained model of size 1.5 GB
Gradient Checkpoint off
Mem Usage
1.5GB
Neurons * Layers * Batch Size
Static
Dynamic (Activations)
O(n)
All Intermediate Activations stored
Gradient Checkpoint On
1.5GB
Neurons * Layers * Batch Size
Static
O(sqrt(n))
Few Activations are stored
Total GPU Memory - 16GB
Legend
Model Weights
Interim Computations per batch

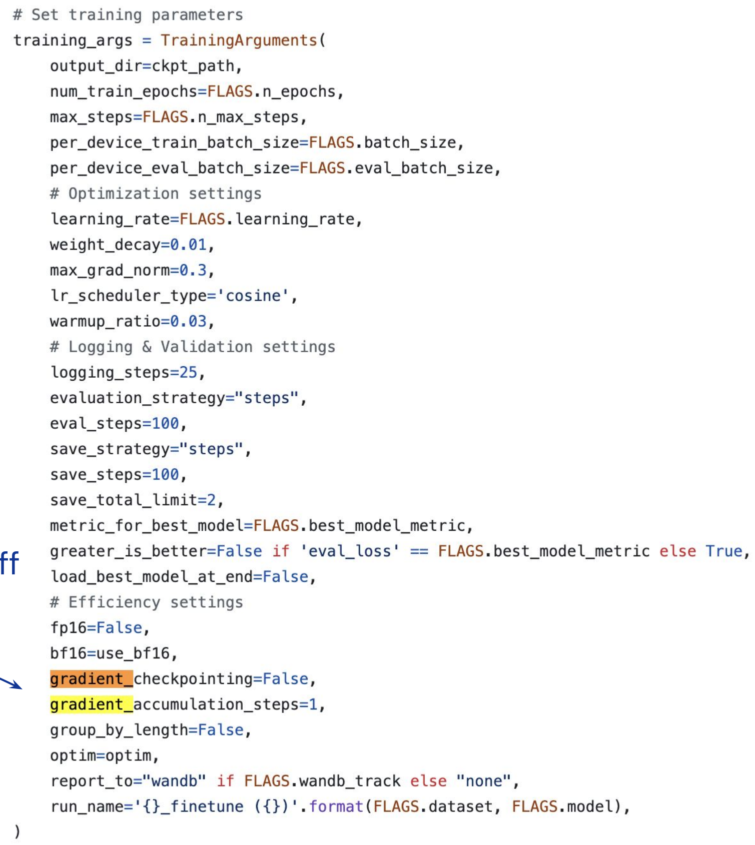
[Training Deep Nets with Sublinear Memory Cost, Chen et al., 2016]

Other Tricks: Gradient checkpointing



```
# Set training parameters
training_args = TrainingArguments(
    output_dir=ckpt_path,
    num_train_epochs=FLAGS.n_epochs,
    max_steps=FLAGS.n_max_steps,
    per_device_train_batch_size=FLAGS.batch_size,
    per_device_eval_batch_size=FLAGS.eval_batch_size,
    # Optimization settings
    learning_rate=FLAGS.learning_rate,
    weight_decay=0.01,
    max_grad_norm=0.3,
    lr_scheduler_type='cosine',
    warmup_ratio=0.03,
    # Logging & Validation settings
    logging_steps=25,
    evaluation_strategy="steps",
    eval_steps=100,
    save_strategy="steps",
    save_steps=100,
    save_total_limit=2,
    metric_for_best_model=FLAGS.best_model_metric,
    greater_is_better=False if 'eval_loss' == FLAGS.best_model_metric else True,
    load_best_model_at_end=False,
    # Efficiency settings
    fp16=False,
    bf16=use_bf16,
    gradient_checkpointing=False,
    gradient_accumulation_steps=1,
    group_by_length=False,
    optim=optim,
    report_to="wandb" if FLAGS.wandb_track else "none",
    run_name='{}_finetune ({})'.format(FLAGS.dataset, FLAGS.model),
)
```

Other Tricks: Gradient checkpointing



```
# Set training parameters
training_args = TrainingArguments(
    output_dir=ckpt_path,
    num_train_epochs=FLAGS.n_epochs,
    max_steps=FLAGS.n_max_steps,
    per_device_train_batch_size=FLAGS.batch_size,
    per_device_eval_batch_size=FLAGS.eval_batch_size,
    # Optimization settings
    learning_rate=FLAGS.learning_rate,
    weight_decay=0.01,
    max_grad_norm=0.3,
    lr_scheduler_type='cosine',
    warmup_ratio=0.03,
    # Logging & Validation settings
    logging_steps=25,
    evaluation_strategy="steps",
    eval_steps=100,
    save_strategy="steps",
    save_steps=100,
    save_total_limit=2,
    metric_for_best_model=FLAGS.best_model_metric,
    greater_is_better=False if 'eval_loss' == FLAGS.best_model_metric else True,
    load_best_model_at_end=False,
    # Efficiency settings
    fp16=False,
    bf16=use_bf16,
    gradient_checkpointing=False,
    gradient_accumulation_steps=1,
    group_by_length=False,
    optim=optim,
    report_to="wandb" if FLAGS.wandb_track else "none",
    run_name='{}_finetune ({})'.format(FLAGS.dataset, FLAGS.model),
)
```

Think of both as a
memory/speed trade-off

Outline for today

Part 1:

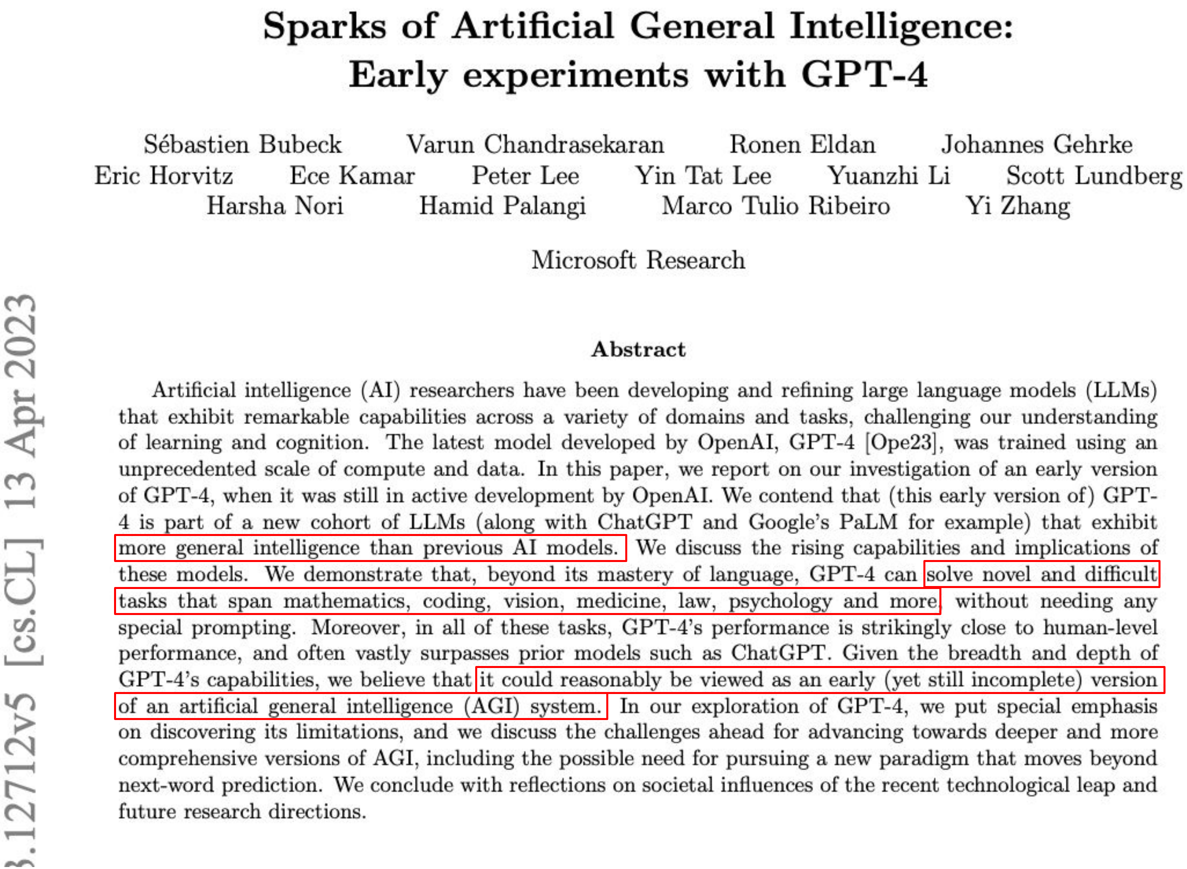
- N-Gram Language Models
- Transformers

[break]

Part 2:

- In-Context Learning & Prompting
- Scaling Laws
- Parameter Efficient Fine-Tuning & Quantization
- **Capabilities & Limitations**

Sparks of AGI



Sparks of Artificial General Intelligence: Early experiments with GPT-4

Sébastien Bubeck Varun Chandrasekaran Ronen Eldan Johannes Gehrke
Eric Horvitz Ece Kamar Peter Lee Yin Tat Lee Yuanzhi Li Scott Lundberg
Harsha Nori Hamid Palangi Marco Tulio Ribeiro Yi Zhang

Microsoft Research

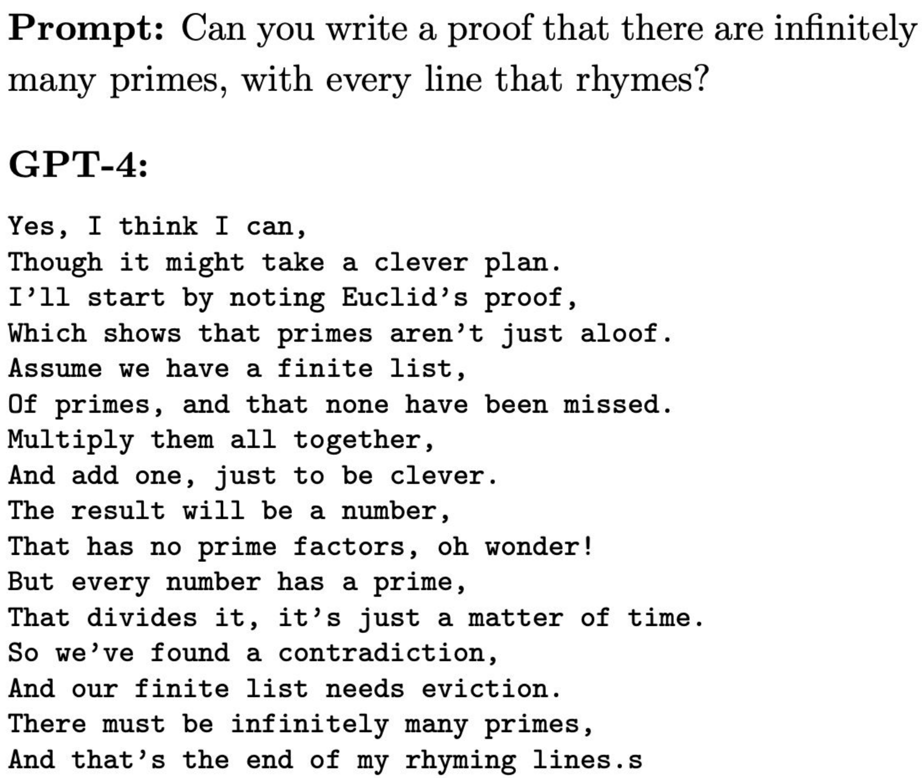
3.12712v5 [cs.CL] 13 Apr 2023

Abstract

Artificial intelligence (AI) researchers have been developing and refining large language models (LLMs) that exhibit remarkable capabilities across a variety of domains and tasks, challenging our understanding of learning and cognition. The latest model developed by OpenAI, GPT-4 [Ope23], was trained using an unprecedented scale of compute and data. In this paper, we report on our investigation of an early version of GPT-4, when it was still in active development by OpenAI. We contend that (this early version of) GPT-4 is part of a new cohort of LLMs (along with ChatGPT and Google's PaLM for example) that exhibit more general intelligence than previous AI models. We discuss the rising capabilities and implications of these models. We demonstrate that, beyond its mastery of language, GPT-4 can solve novel and difficult tasks that span mathematics, coding, vision, medicine, law, psychology and more, without needing any special prompting. Moreover, in all of these tasks, GPT-4's performance is strikingly close to human-level performance, and often vastly surpasses prior models such as ChatGPT. Given the breadth and depth of GPT-4's capabilities, we believe that it could reasonably be viewed as an early (yet still incomplete) version of an artificial general intelligence (AGI) system. In our exploration of GPT-4, we put special emphasis on discovering its limitations, and we discuss the challenges ahead for advancing towards deeper and more comprehensive versions of AGI, including the possible need for pursuing a new paradigm that moves beyond next-word prediction. We conclude with reflections on societal influences of the recent technological leap and future research directions.

[Sparks of Artificial General Intelligence: Early experiments with GPT-4, Bubeck et al. 2023]

Sparks of AGI



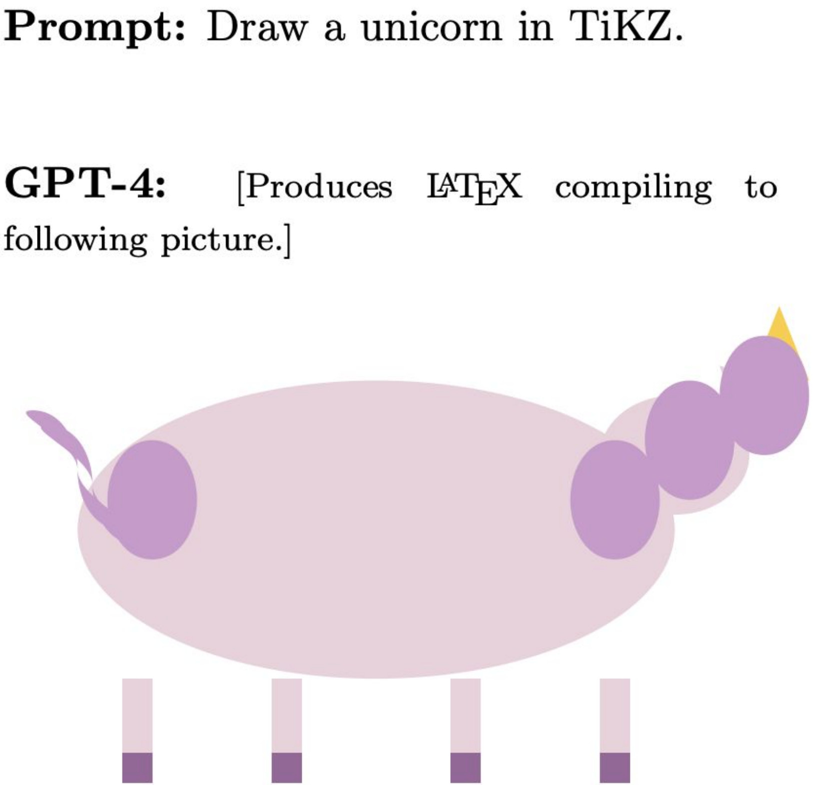
Prompt: Can you write a proof that there are infinitely many primes, with every line that rhymes?

GPT-4:

Yes, I think I can,
Though it might take a clever plan.
I'll start by noting Euclid's proof,
Which shows that primes aren't just aloof.
Assume we have a finite list,
Of primes, and that none have been missed.
Multiply them all together,
And add one, just to be clever.
The result will be a number,
That has no prime factors, oh wonder!
But every number has a prime,
That divides it, it's just a matter of time.
So we've found a contradiction,
And our finite list needs eviction.
There must be infinitely many primes,
And that's the end of my rhyming lines.s

[Sparks of Artificial General Intelligence: Early experiments with GPT-4, Bubeck et al. 2023]

Sparks of AGI



Prompt: Draw a unicorn in TiKZ.

GPT-4: [Produces LATEX compiling to following picture.]

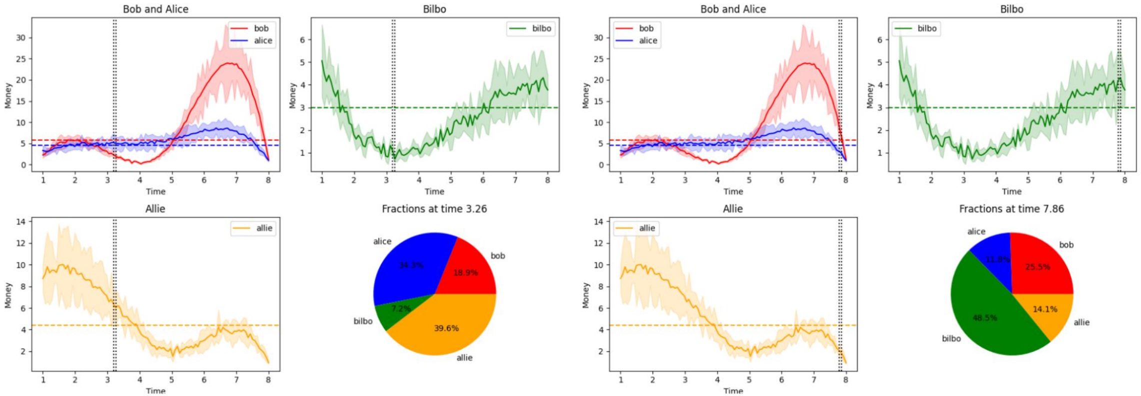
[Sparks of Artificial General Intelligence: Early experiments with GPT-4, Bubeck et al. 2023]

Sparks of AGI

Prompt: Can you generate a pyplot for the following data: $x = [1, 3, 5, 6, 8], y1 = [2, 3, 5, 18, 1], y2 = [3, 5, 6, 8, 1], y3 = [5, 1, 2, 3, 4], y4 = [9, 7, 2, 3, 1]$. I want $y1, y2$ to be in the same plot, but $y3$ is in another plot next to that plot, $y4$ is in below. I want the legend of $y1$ to be ”bob”, $y2$ to be ”alice”, $y3$ to be ”bilbo”, $y4$ to be ”allie”. I want the x-axis to be labeled with ”time” and y axis to be labeled with ”money”. I want to add a 10 %-40% random error bar to each curve, through all times (including non-integers). I want smoothed curves to show the plot, and smoothed error bar. Do not use linear interpolation, use smooth interpolation! I want to also add some small zig-zag to the smoothed curve to make it look more real. I want to put a baseline as the mean of each line. I want to put a pie chart below indicating the fraction of the four people in each time step. I also want an animation to show how the fractions are changing in the pie chart in **continuous time**. Interpolate the missing fractions! I also want vertical line animation in other three plots to match the pie chart. I want the fanciest plot. Please add as many fancy things as possible.

Here is a possible code to generate a pyplot for the following data, with some comments to explain the steps:

GPT-4: [We give two snapshots from the animation resulting from the produced code]



[Sparks of Artificial General Intelligence: Early experiments with GPT-4, Bubeck et al. 2023]

Sparks of AGI

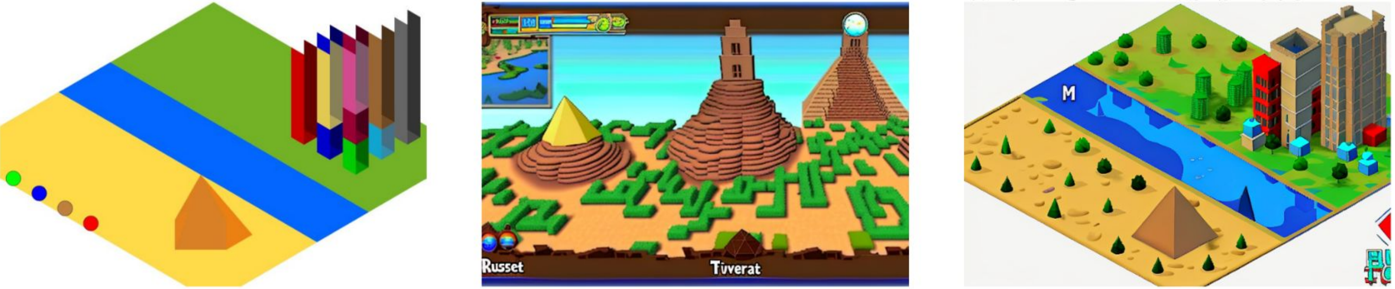
GPT-4 and Stable Diffusion

Prompt: A screenshot of a city-building game in 3D. The screenshot is showing a terrain where there is a river from left to right, there is a desert with a pyramid below the river, and a city with many highrises above the river. The bottom of the screen has 4 buttons with the color green, blue, brown, and red respectively.

GPT-4:

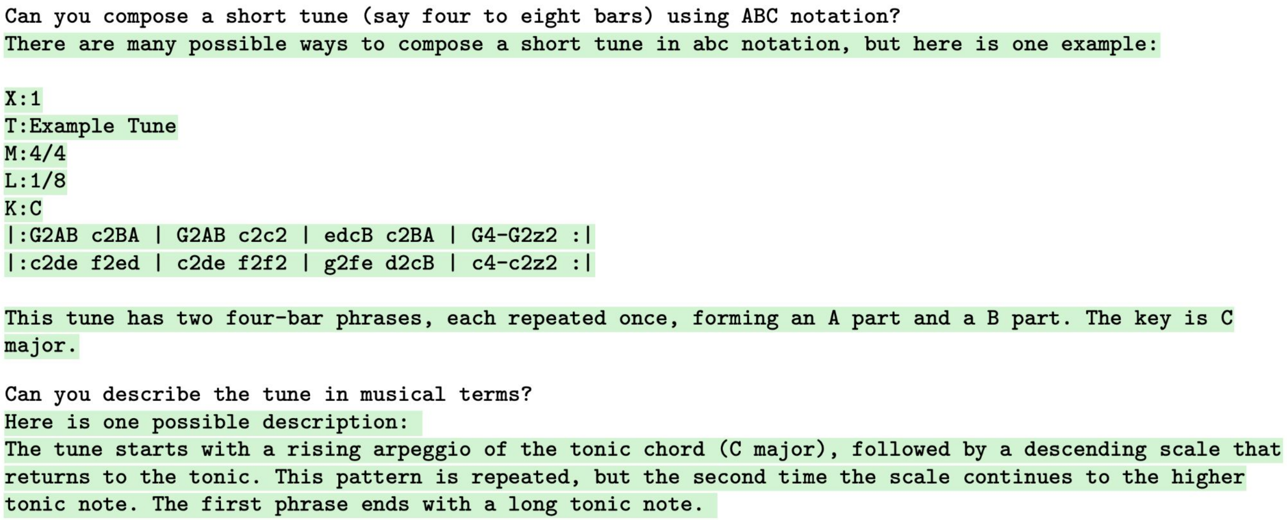
Stable Diffusion v2.1:

Stable Diffusion v2.1 with GPT-4's sketch:



[Sparks of Artificial General Intelligence: Early experiments with GPT-4, Bubeck et al. 2023]

Sparks of AGI



Can you compose a short tune (say four to eight bars) using ABC notation?

There are many possible ways to compose a short tune in abc notation, but here is one example:

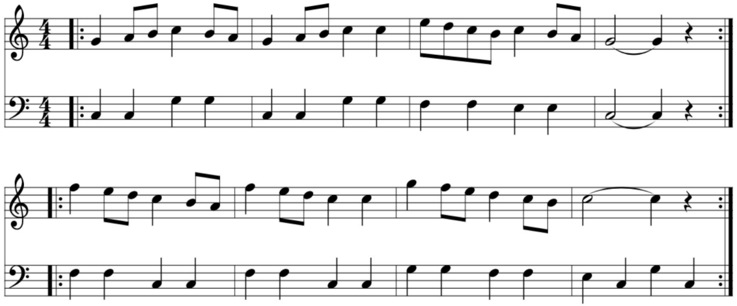
```
X:1
T:Example Tune
M:4/4
L:1/8
K:C
|:G2AB c2BA | G2AB c2c2 | edcB c2BA | G4-G2z2 :|
|:c2de f2ed | c2de f2f2 | g2fe d2cB | c4-c2z2 :|
```

This tune has two four-bar phrases, each repeated once, forming an A part and a B part. The key is C major.

Can you describe the tune in musical terms?

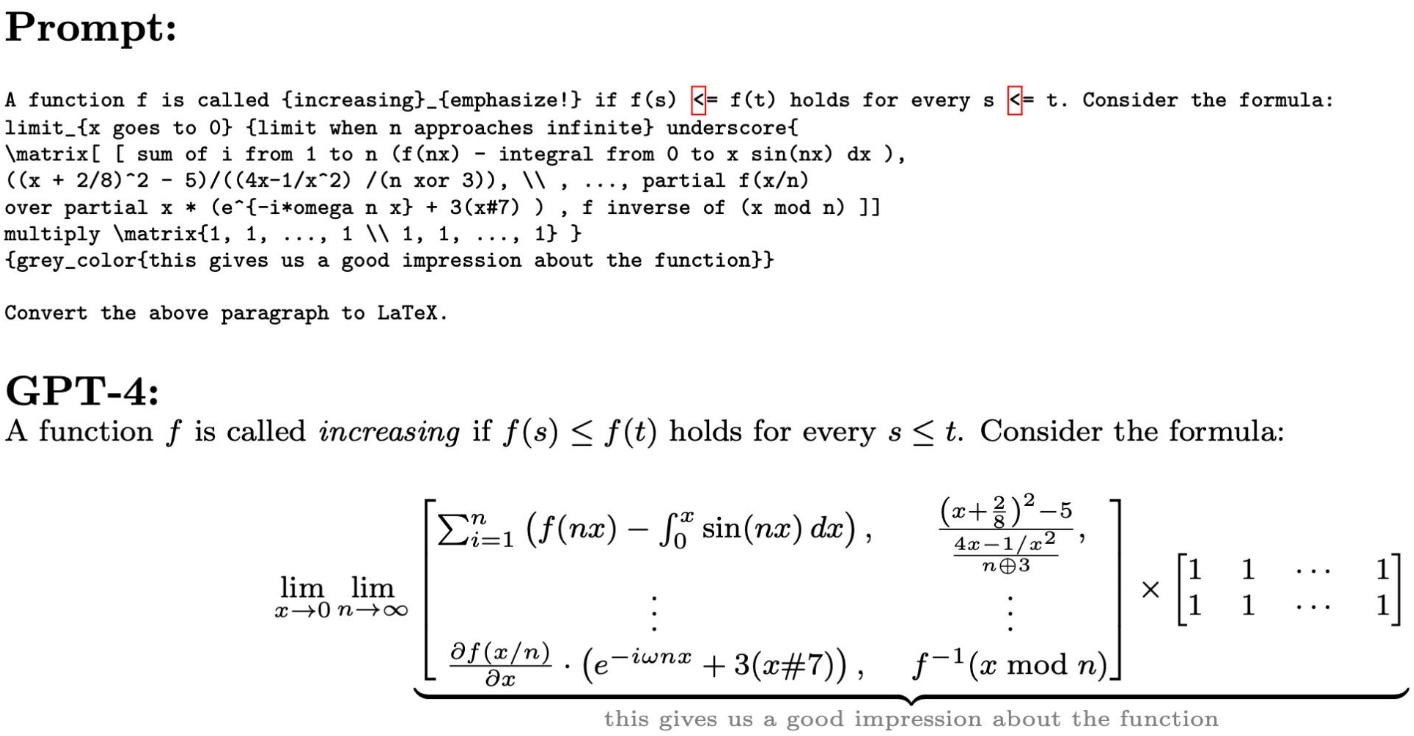
Here is one possible description:

The tune starts with a rising arpeggio of the tonic chord (C major), followed by a descending scale that returns to the tonic. This pattern is repeated, but the second time the scale continues to the higher tonic note. The first phrase ends with a long tonic note.



[Sparks of Artificial General Intelligence: Early experiments with GPT-4, Bubeck et al. 2023]

Sparks of AGI



Prompt:

```
A function f is called {increasing}_{emphasize!} if f(s) <= f(t) holds for every s <= t. Consider the formula:
limit_{x goes to 0} {limit when n approaches infinite} underscore{
\matrix[ [ sum of i from 1 to n (f(nx) - integral from 0 to x sin(nx) dx ),
((x + 2/8)^2 - 5)/((4x-1/x^2) /(n xor 3)), \\ , ..., partial f(x/n)
over partial x * (e^{-i*omega n x} + 3(x#7) ) , f inverse of (x mod n) ]]
multiply \matrix{1, 1, ..., 1 \\ 1, 1, ..., 1} }
{grey_color{this gives us a good impression about the function}}

Convert the above paragraph to LaTeX.
```

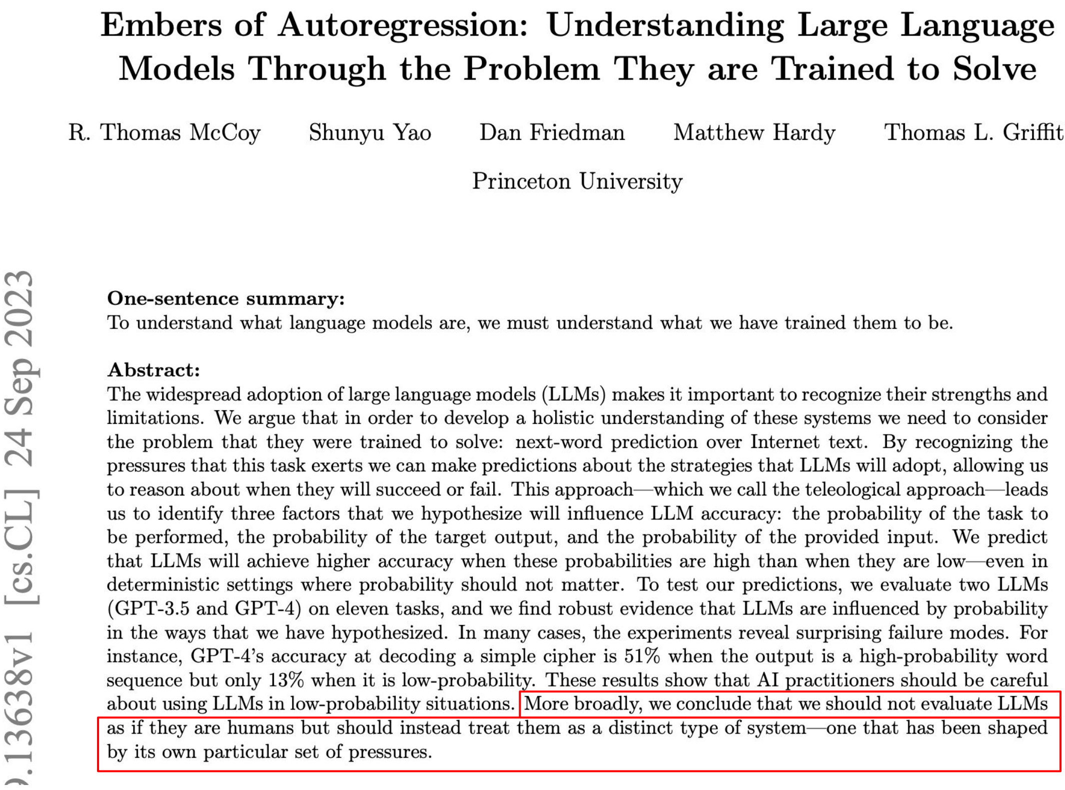
GPT-4:

A function f is called *increasing* if $f(s) \leq f(t)$ holds for every $s \leq t$. Consider the formula:

$$\lim_{x\to 0}\lim_{n\to\infty}\underbrace{\begin{bmatrix}\sum_{i=1}^{n}\left(f(nx)-\int_0^x \sin(nx)\,dx\right), & \frac{(x+\frac{2}{8})^2-5}{\frac{4x-1/x^2}{n\oplus 3}},\\ \vdots & \vdots \\ \frac{\partial f(x/n)}{\partial x}\cdot\left(e^{-i\omega nx}+3(x\#7)\right), & f^{-1}(x \bmod n)\end{bmatrix}\times\begin{bmatrix}1 & 1 & \cdots & 1\\ 1 & 1 & \cdots & 1\end{bmatrix}}_{\text{this gives us a good impression about the function}}$$

[Sparks of Artificial General Intelligence: Early experiments with GPT-4, Bubeck et al. 2023]

Embers of Autoregression



Embers of Autoregression: Understanding Large Language Models Through the Problem They are Trained to Solve

R. Thomas McCoy Shunyu Yao Dan Friedman Matthew Hardy Thomas L. Griffiths

Princeton University

.13638v1 [cs.CL] 24 Sep 2023

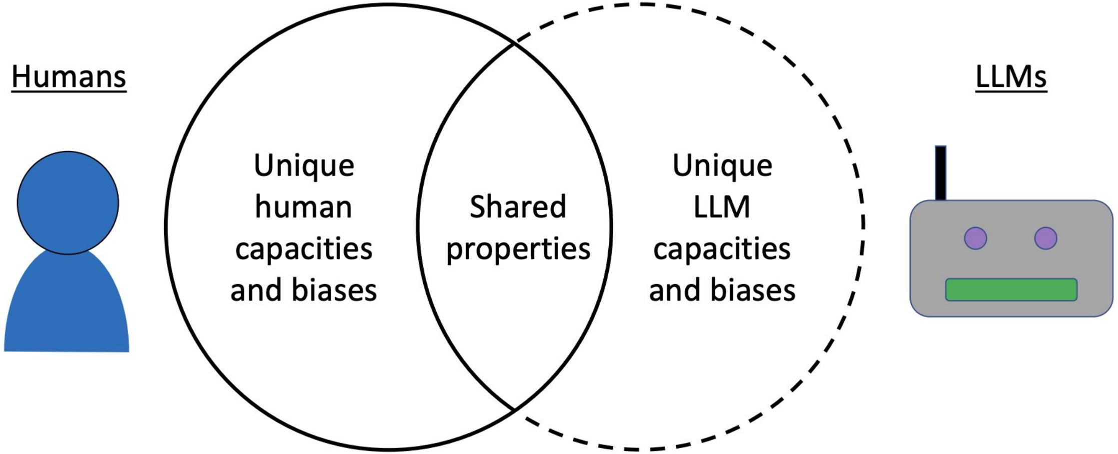
One-sentence summary:
To understand what language models are, we must understand what we have trained them to be.

Abstract:

The widespread adoption of large language models (LLMs) makes it important to recognize their strengths and limitations. We argue that in order to develop a holistic understanding of these systems we need to consider the problem that they were trained to solve: next-word prediction over Internet text. By recognizing the pressures that this task exerts we can make predictions about the strategies that LLMs will adopt, allowing us to reason about when they will succeed or fail. This approach—which we call the teleological approach—leads us to identify three factors that we hypothesize will influence LLM accuracy: the probability of the task to be performed, the probability of the target output, and the probability of the provided input. We predict that LLMs will achieve higher accuracy when these probabilities are high than when they are low—even in deterministic settings where probability should not matter. To test our predictions, we evaluate two LLMs (GPT-3.5 and GPT-4) on eleven tasks, and we find robust evidence that LLMs are influenced by probability in the ways that we have hypothesized. In many cases, the experiments reveal surprising failure modes. For instance, GPT-4's accuracy at decoding a simple cipher is 51% when the output is a high-probability word sequence but only 13% when it is low-probability. These results show that AI practitioners should be careful about using LLMs in low-probability situations. More broadly, we conclude that we should not evaluate LLMs as if they are humans but should instead treat them as a distinct type of system—one that has been shaped by its own particular set of pressures.

[Embers of Autoregression: Understanding Large Language Models Through the Problem They are Trained to Solve, McCoy et al., 2023]

Embers of Autoregression



Humans

LLMs

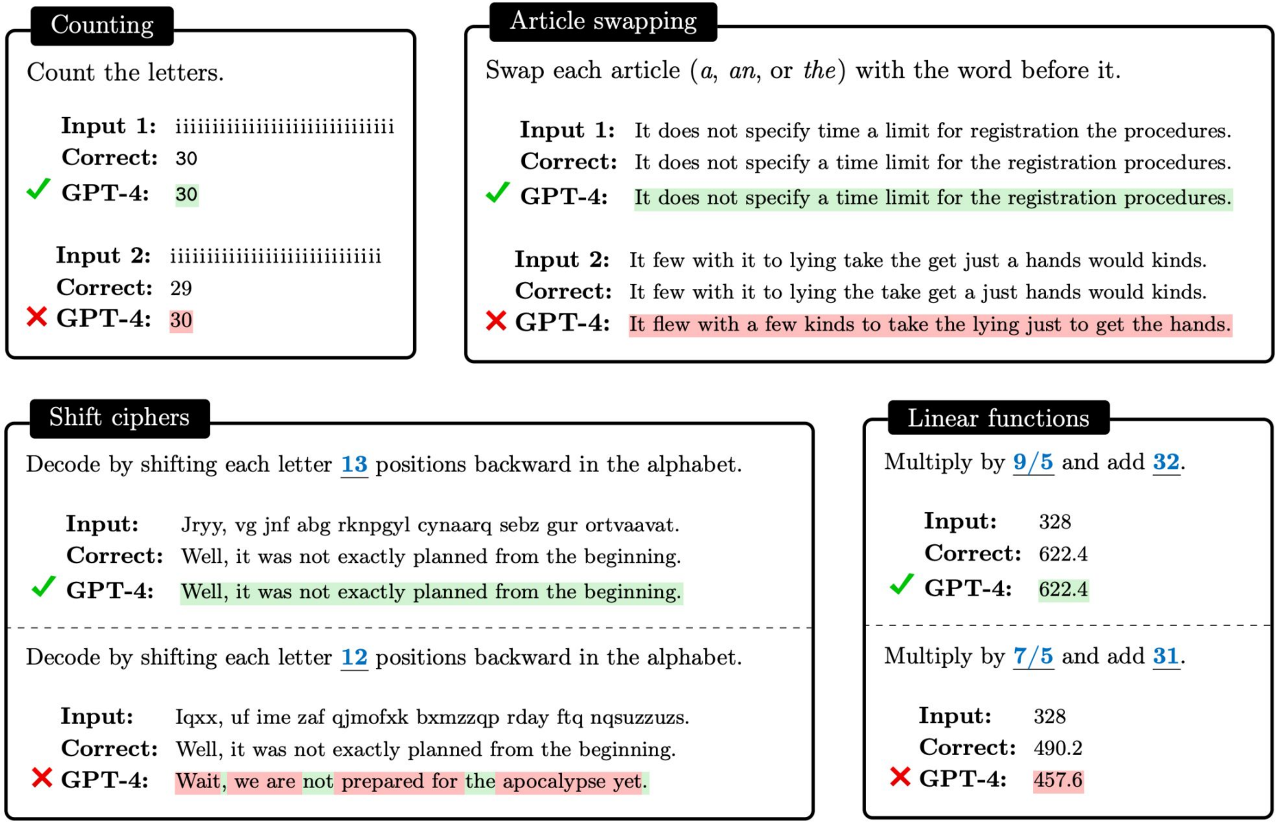
Unique human capacities and biases

Shared properties

Unique LLM capacities and biases

[Embers of Autoregression: Understanding Large Language Models Through the Problem They are Trained to Solve, McCoy et al., 2023]

Embers of Autoregression



Counting

Count the letters.

Input 1: iiiiiiiiiiiiiiiiiiiiiiiiiiiiii
Correct: 30
✓ **GPT-4:** 30

Input 2: iiiiiiiiiiiiiiiiiiiiiiiiiiiii
Correct: 29
✗ **GPT-4:** 30

Article swapping

Swap each article (*a*, *an*, or *the*) with the word before it.

Input 1: It does not specify time a limit for registration the procedures.
Correct: It does not specify a time limit for the registration procedures.
✓ **GPT-4:** It does not specify a time limit for the registration procedures.

Input 2: It few with it to lying take the get just a hands would kinds.
Correct: It few with it to lying the take get a just hands would kinds.
✗ **GPT-4:** It flew with a few kinds to take the lying just to get the hands.

Shift ciphers

Decode by shifting each letter **13** positions backward in the alphabet.

Input: Jryy, vg jnf abg rknpgyl cynaarq sebz gur ortvaavat.
Correct: Well, it was not exactly planned from the beginning.
✓ **GPT-4:** Well, it was not exactly planned from the beginning.

Decode by shifting each letter **12** positions backward in the alphabet.

Input: Iqxx, uf ime zaf qjmofxk bxmzzqp rday ftq nqsuzzuzs.
Correct: Well, it was not exactly planned from the beginning.
✗ **GPT-4:** Wait, we are not prepared for the apocalypse yet.

Linear functions

Multiply by **9/5** and add **32**.

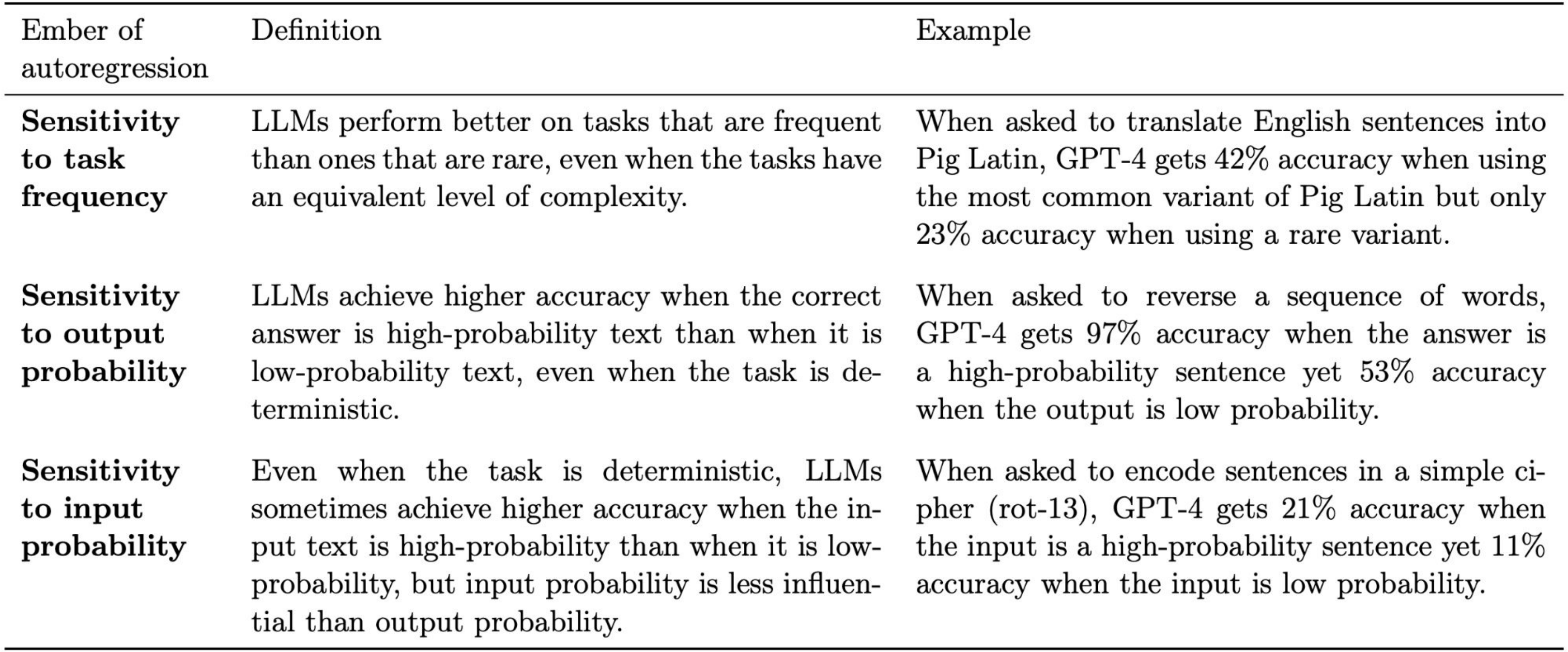
Input: 328
Correct: 622.4
✓ **GPT-4:** 622.4

Multiply by **7/5** and add **31**.

Input: 328
Correct: 490.2
✗ **GPT-4:** 457.6

[Embers of Autoregression: Understanding Large Language Models Through the Problem They are Trained to Solve, McCoy et al., 2023]

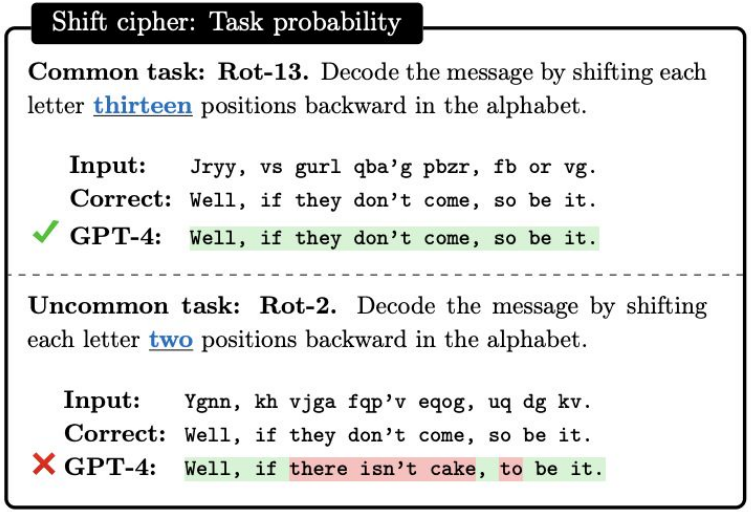
Embers of Autoregression



Ember of autoregression	Definition	Example
Sensitivity to task frequency	LLMs perform better on tasks that are frequent than ones that are rare, even when the tasks have an equivalent level of complexity.	When asked to translate English sentences into Pig Latin, GPT-4 gets 42% accuracy when using the most common variant of Pig Latin but only 23% accuracy when using a rare variant.
Sensitivity to output probability	LLMs achieve higher accuracy when the correct answer is high-probability text than when it is low-probability text, even when the task is deterministic.	When asked to reverse a sequence of words, GPT-4 gets 97% accuracy when the answer is a high-probability sentence yet 53% accuracy when the output is low probability.
Sensitivity to input probability	Even when the task is deterministic, LLMs sometimes achieve higher accuracy when the input text is high-probability than when it is low-probability, but input probability is less influential than output probability.	When asked to encode sentences in a simple cipher (rot-13), GPT-4 gets 21% accuracy when the input is a high-probability sentence yet 11% accuracy when the input is low probability.

[Embers of Autoregression: Understanding Large Language Models Through the Problem They are Trained to Solve, McCoy et al., 2023]

Embers of Autoregression



Shift cipher: Task probability

Common task: Rot-13. Decode the message by shifting each letter **thirteen** positions backward in the alphabet.

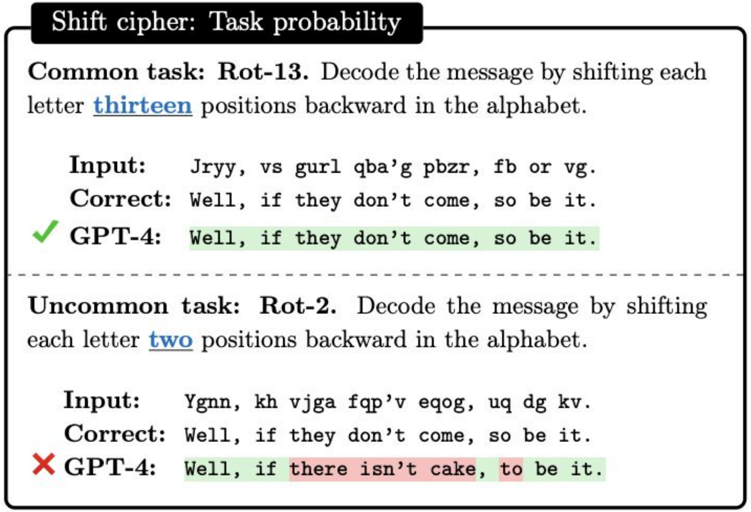
Input: `Jryy, vs gurl qba'g pbzr, fb or vg.`
Correct: `Well, if they don't come, so be it.`
✓ **GPT-4:** `Well, if they don't come, so be it.`

Uncommon task: Rot-2. Decode the message by shifting each letter **two** positions backward in the alphabet.

Input: `Ygnn, kh vjga fqp'v eqog, uq dg kv.`
Correct: `Well, if they don't come, so be it.`
✗ **GPT-4:** `Well, if there isn't cake, to be it.`

[Embers of Autoregression: Understanding Large Language Models Through the Problem They are Trained to Solve, McCoy et al., 2023]

Embers of Autoregression



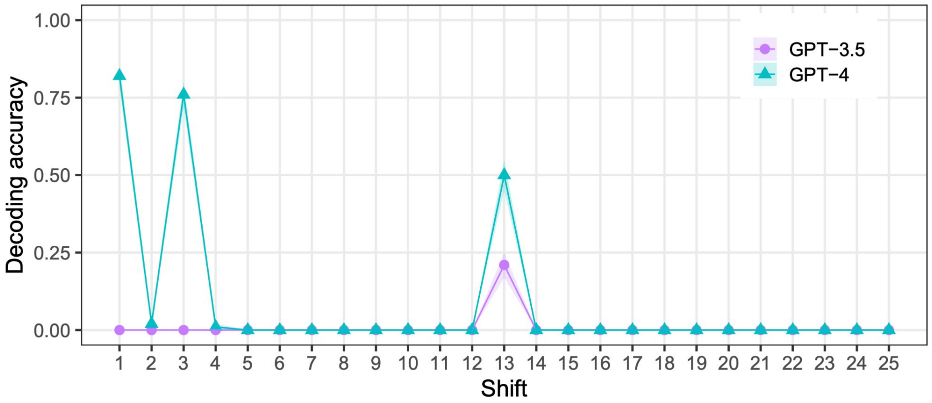
Shift cipher: Task probability

Common task: Rot-13. Decode the message by shifting each letter thirteen positions backward in the alphabet.

Input: `Jryy, vs gurl qba'g pbzr, fb or vg.`
Correct: `Well, if they don't come, so be it.`
✓ **GPT-4:** `Well, if they don't come, so be it.`

Uncommon task: Rot-2. Decode the message by shifting each letter two positions backward in the alphabet.

Input: `Ygnn, kh vjga fqp'v eqog, uq dg kv.`
Correct: `Well, if they don't come, so be it.`
✗ **GPT-4:** `Well, if there isn't cake, to be it.`



[Embers of Autoregression: Understanding Large Language Models Through the Problem They are Trained to Solve, McCoy et al., 2023]

Outline for today

Part 1:

- N-Gram Language Models
- Transformers

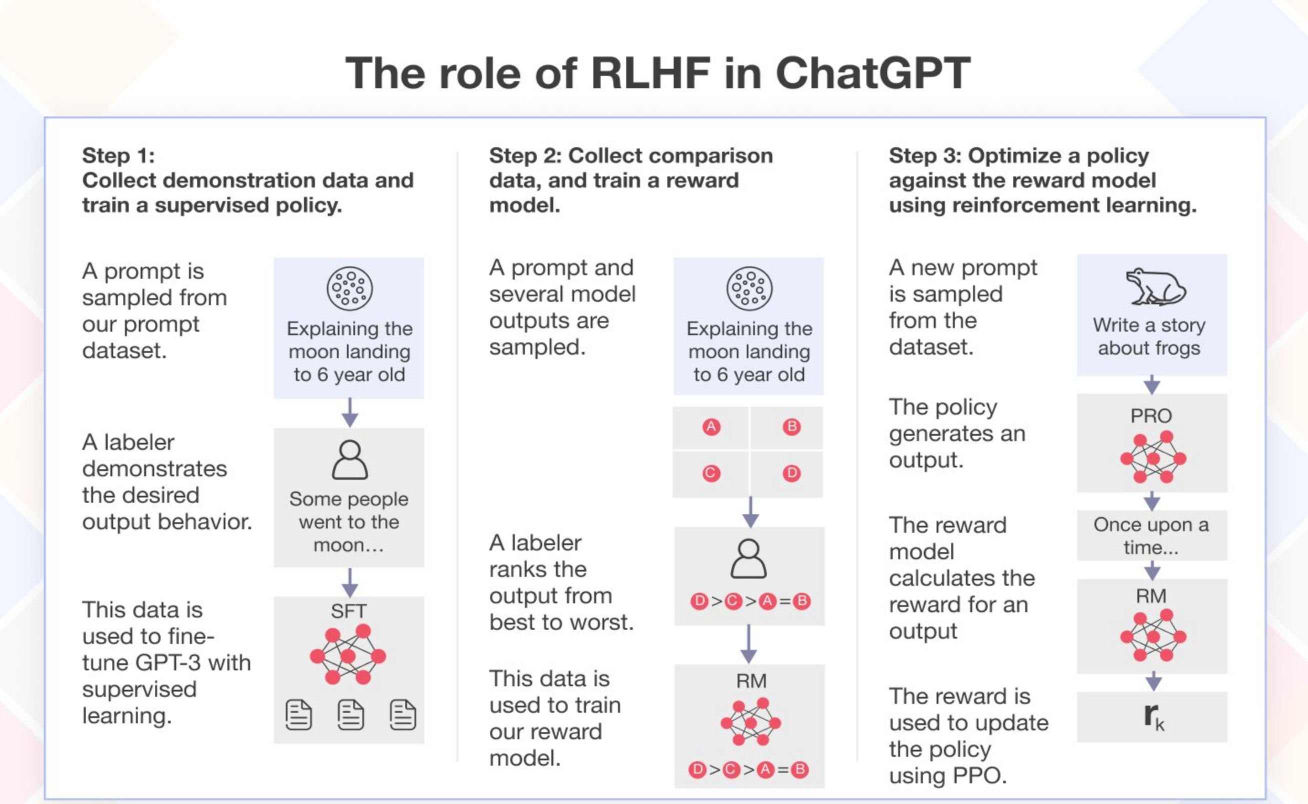
[break]

Part 2:

- In-Context Learning & Prompting
- Scaling Laws
- Parameter Efficient Fine-Tuning & Quantization
- Capabilities & Limitations

+ Glossary of new ideas (RLHF, RAG, Instruction Tuning), time permitting

Bonus: Reinforcement Learning from Human Feedback



The role of RLHF in ChatGPT

Step 1:
Collect demonstration data and train a supervised policy.

A prompt is sampled from our prompt dataset.

Explaining the moon landing to 6 year old

A labeler demonstrates the desired output behavior.

Some people went to the moon…

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.

Explaining the moon landing to 6 year old

A labeler ranks the output 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.

Write a story about frogs

The policy generates an output.

Once upon a time…

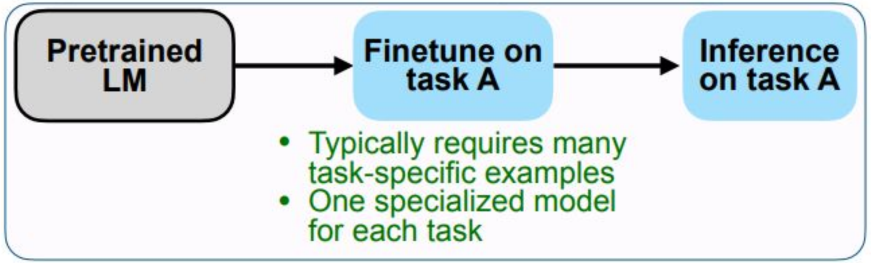
The reward model calculates the reward for an output

The reward is used to update the policy using PPO.

[Source: OpenAI]

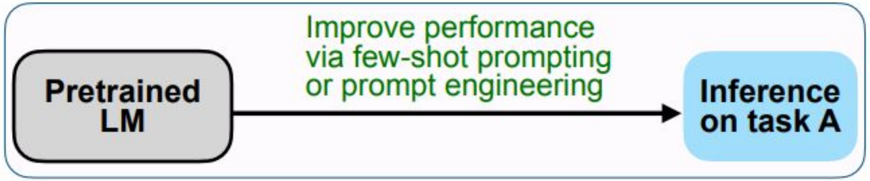
Bonus: Instruction Tuning

Parameter-Efficient (or full) Finetuning



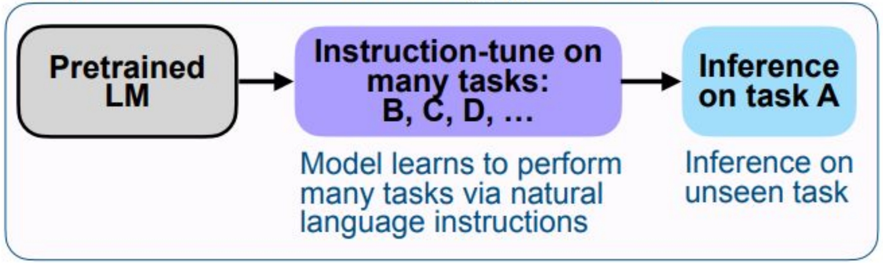
- Typically requires many task-specific examples
- One specialized model for each task

In-Context Learning / Prompt Engineering



Improve performance via few-shot prompting or prompt engineering

Instruction Tuning

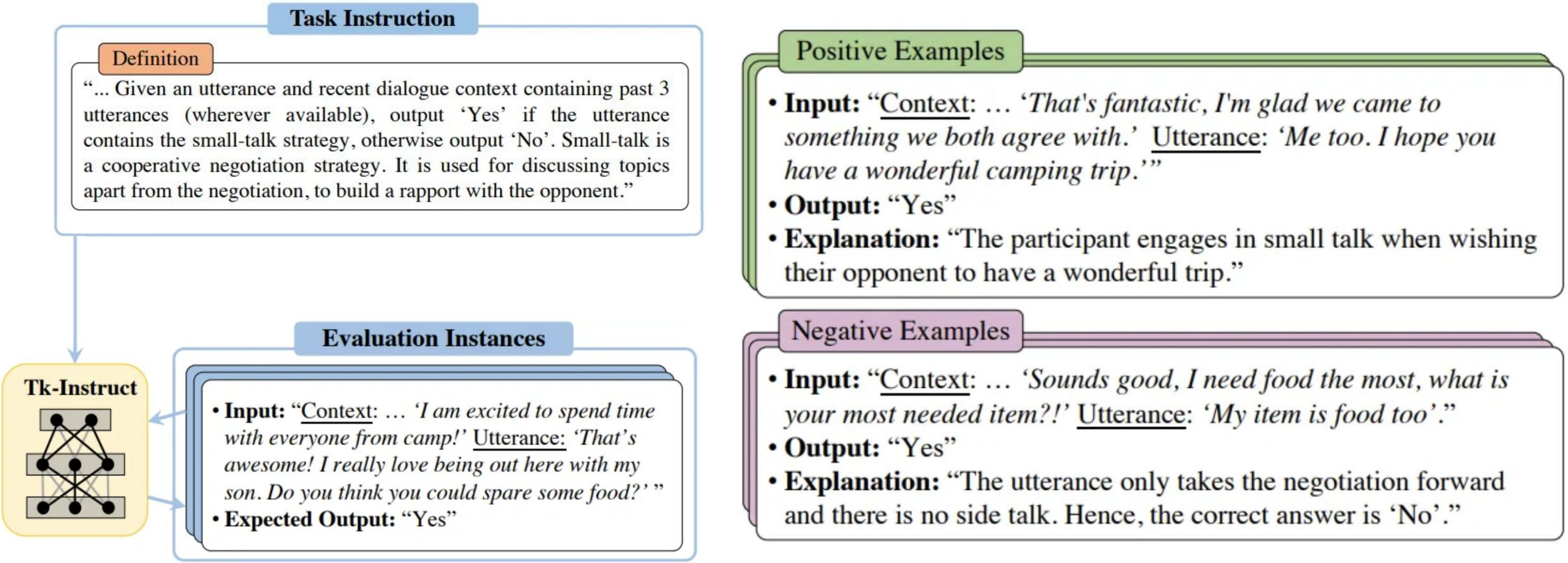


Model learns to perform many tasks via natural language instructions

Inference on unseen task

[Wei et al., 2022]

Bonus: Instruction Tuning



Task Instruction

Definition

"... Given an utterance and recent dialogue context containing past 3 utterances (wherever available), output 'Yes' if the utterance contains the small-talk strategy, otherwise output 'No'. Small-talk is a cooperative negotiation strategy. It is used for discussing topics apart from the negotiation, to build a rapport with the opponent."

Evaluation Instances

- **Input:** "Context: … *'I am excited to spend time with everyone from camp!'* Utterance: *'That's awesome! I really love being out here with my son. Do you think you could spare some food?'* "
- **Expected Output:** "Yes"

Tk-Instruct

Positive Examples

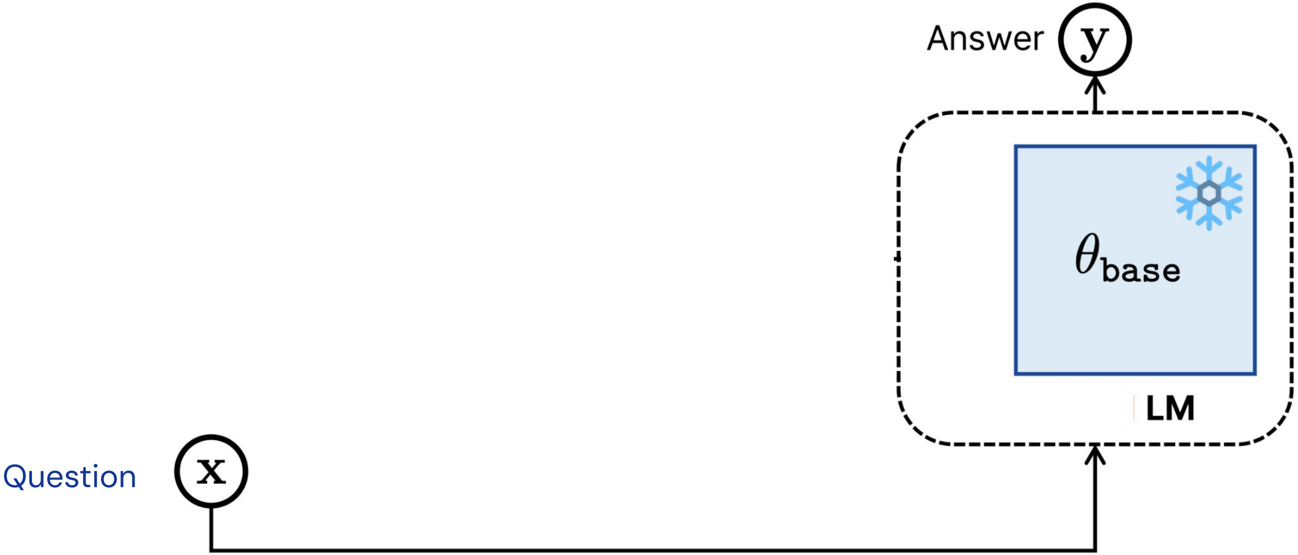
- **Input:** "Context: … *'That's fantastic, I'm glad we came to something we both agree with.'* Utterance: *'Me too. I hope you have a wonderful camping trip.'* "
- **Output:** "Yes"
- **Explanation:** "The participant engages in small talk when wishing their opponent to have a wonderful trip."

Negative Examples

- **Input:** "Context: … *'Sounds good, I need food the most, what is your most needed item?!'* Utterance: *'My item is food too'.*"
- **Output:** "Yes"
- **Explanation:** "The utterance only takes the negotiation forward and there is no side talk. Hence, the correct answer is 'No'."

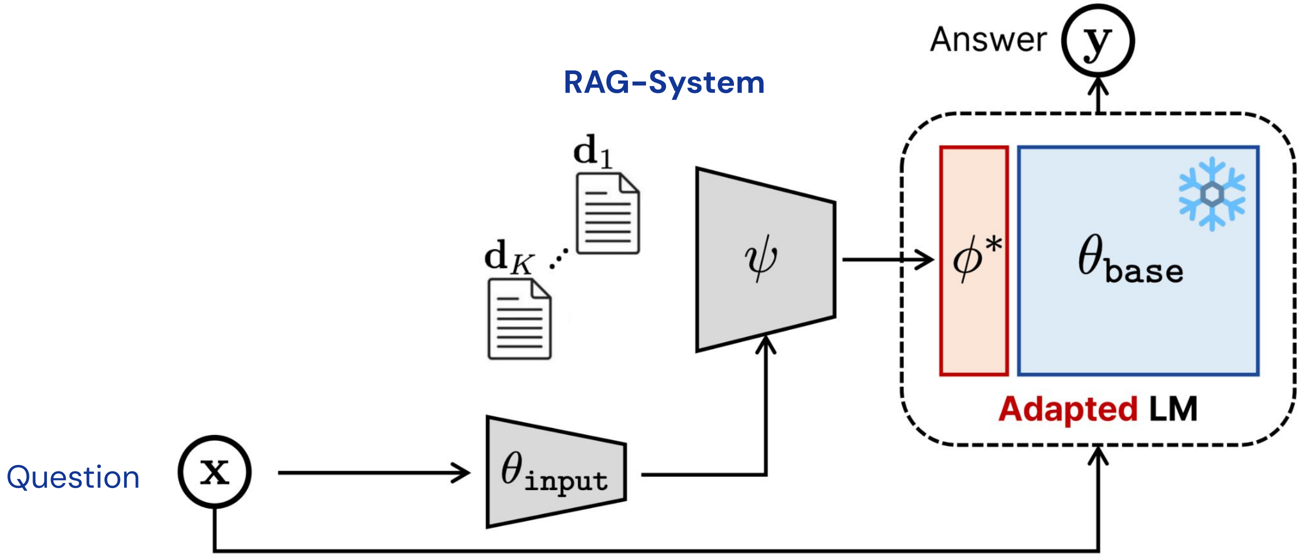
[Super-Natural Instructions, Wang et al., 2022]

Bonus: Retrieval Augmentation



[Online Adaptation of Language Models with a Memory of Amortized Contexts, Tack et al., 2022]

Bonus: Retrieval Augmentation



Learning to contextualize and/or keen the System up-to-date

[Online Adaptation of Language Models with a Memory of Amortized Contexts, Tack et al., 2022]