CENG501 – Deep Learning

Week 7

Spring 2026

Sinan Kalkan

Dept. of Computer Engineering, METU

Attention

Previously on CENG501

BLEU: Bilingual Evaluation Understudy

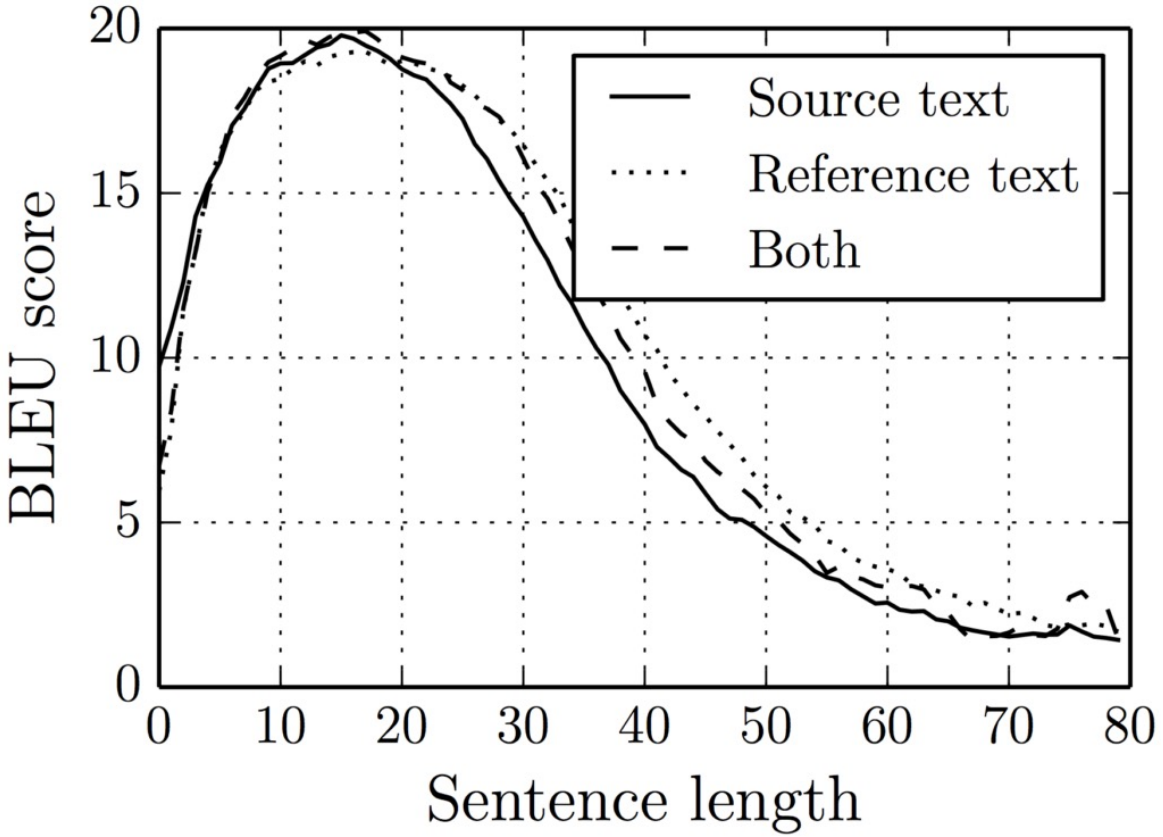
https://cloud.google.com/translate/automl/docs/evaluate#bleu

Published as a conference paper at ICLR 2015

NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE

Dzmitry Bahdanau
Jacobs University Bremen, Germany

KyungHyun Cho **Yoshua Bengio***
Université de Montréal



Attention

Previously on CENG391

Published as a conference paper at ICLR 2015

NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE

Dzmitry Bahdanau
Jacobs University Bremen, Germany

KyungHyun Cho **Yoshua Bengio***
Université de Montréal

In a new model architecture, we define each conditional probability in Eq. (2) as:

$$p(y_i|y_1,\dots,y_{i-1},\mathbf{x}) = g(y_{i-1},s_i,c_i), \tag{4}$$

where s_i is an RNN hidden state for time i, computed by

$$s_i = f(s_{i-1},y_{i-1},c_i).$$

It should be noted that unlike the existing encoder–decoder approach (see Eq. (2)), here the probability is conditioned on a distinct context vector c_i for each target word y_i.

The context vector c_i depends on a sequence of *annotations* $(h_1,\cdots,h_{T_x})$ to which an encoder maps the input sentence. Each annotation h_i contains information about the whole input sequence with a strong focus on the parts surrounding the i-th word of the input sequence. We explain in detail how the annotations are computed in the next section.

The context vector c_i is, then, computed as a weighted sum of these annotations h_i:

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j. \tag{5}$$

The weight α_{ij} of each annotation h_j is computed by

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x}\exp(e_{ik})}, \tag{6}$$

where

$$e_{ij} = a(s_{i-1},h_j)$$

is an *alignment model* which scores how well the inputs around position j and the output at position i match. The score is based on the RNN hidden state s_{i-1} (just before emitting y_i, Eq. (4)) and the j-th annotation h_j of the input sentence.

We parametrize the alignment model a as a feedforward neural network which is jointly trained with all the other components of the proposed system. Note that unlike in traditional machine translation,

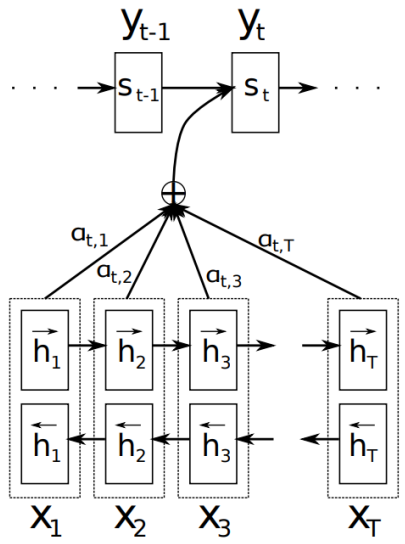


Figure 1: The graphical illustration of the proposed model trying to generate the t-th target word y_t given a source sentence $(x_1,x_2,\dots,x_T)$.

Previously on CENG501

Attention Types

Name	Alignment score function	Citation
Content-base attention	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \text{cosine}[\boldsymbol{s}_t, \boldsymbol{h}_i]$	Graves2014
Additive(*)	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \mathbf{v}_a^\top \tanh(\mathbf{W}_a[\boldsymbol{s}_t; \boldsymbol{h}_i])$	Bahdanau2015
Location-Base	$\alpha_{t,i} = \text{softmax}(\mathbf{W}_a \boldsymbol{s}_t)$ Note: This simplifies the softmax alignment to only depend on the target position.	Luong2015
General	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \boldsymbol{s}_t^\top \mathbf{W}_a \boldsymbol{h}_i$ where $\mathbf{W}_a$ is a trainable weight matrix in the attention layer.	Luong2015
Dot-Product	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \boldsymbol{s}_t^\top \boldsymbol{h}_i$	Luong2015
Scaled Dot-Product(^)	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \frac{\boldsymbol{s}_t^\top \boldsymbol{h}_i}{\sqrt{n}}$ Note: very similar to the dot-product attention except for a scaling factor; where n is the dimension of the source hidden state.	Vaswani2017

(*) Referred to as "concat" in Luong, et al., 2015 and as "additive attention" in Vaswani, et al., 2017.
(^) It adds a scaling factor $1/\sqrt{n}$, motivated by the concern when the input is large, the softmax function may have an extremely small gradient, hard for efficient learning.

Table: https://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html

Previously on CENG501

Attention: Transformer

- Vanilla self attention:

$$e_i' = \sum_j \frac{\exp\left(e_j^T e_i\right)}{\sum_m \exp(e_m^T e_i)} e_j$$

- Scaled-dot product attention:

$$e_i' = \sum_j \frac{\exp\left(k\left(e_j^T\right) q(e_i)\right)}{\sum_m \exp(k(e_m^T) q(e_i))} v(e_j)$$

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

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

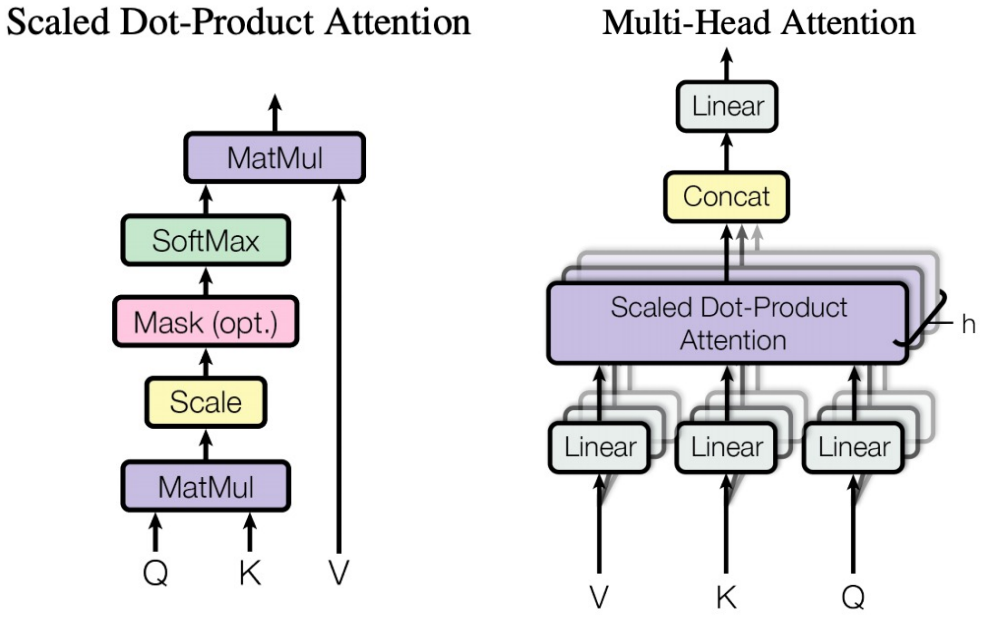
Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com

Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu

Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Polosukhin* ‡ illia.polosukhin@gmail.com



Previously on CENG501

Multi-head Attention

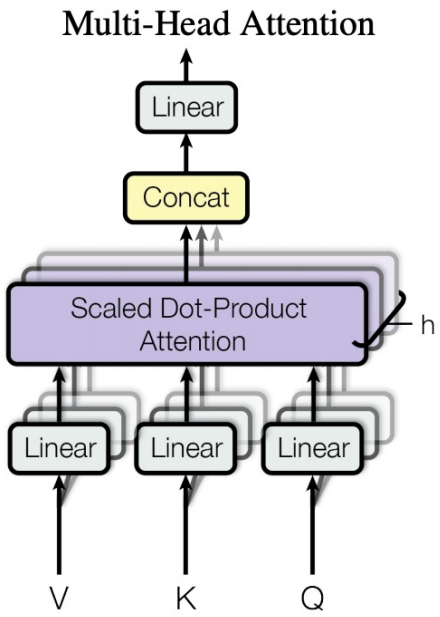


Fig: Attention is all you need.

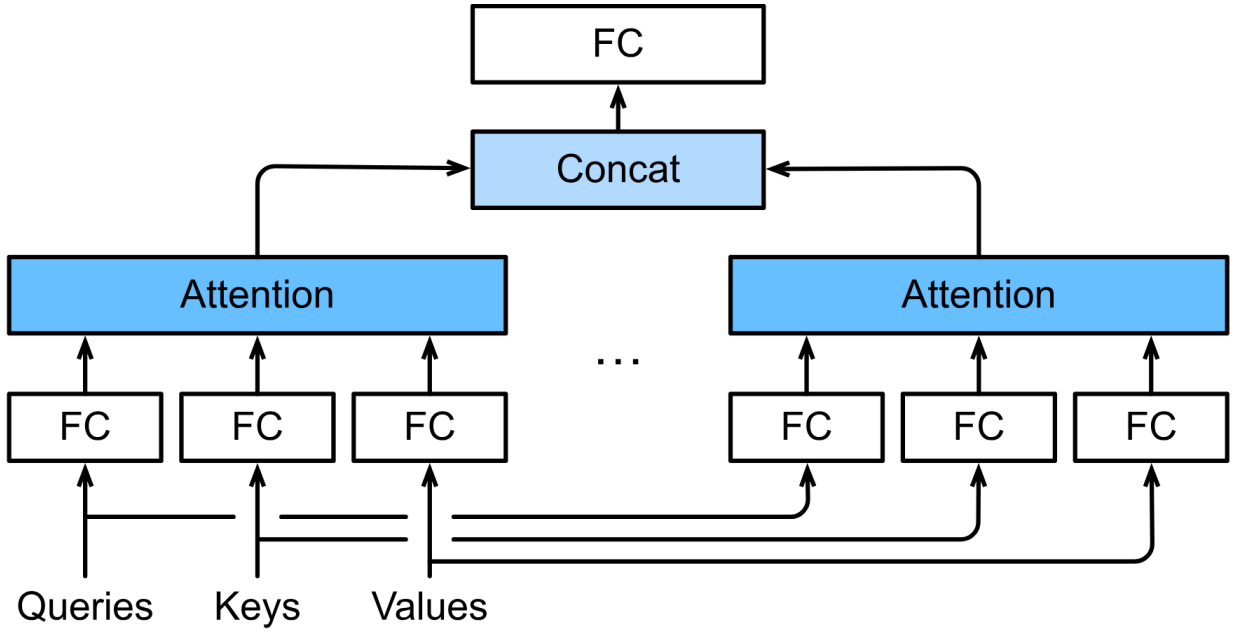


Fig: https://d2l.ai/chapter_attention-mechanisms-and-transformers/multihead-attention.html

Previously on CENG501

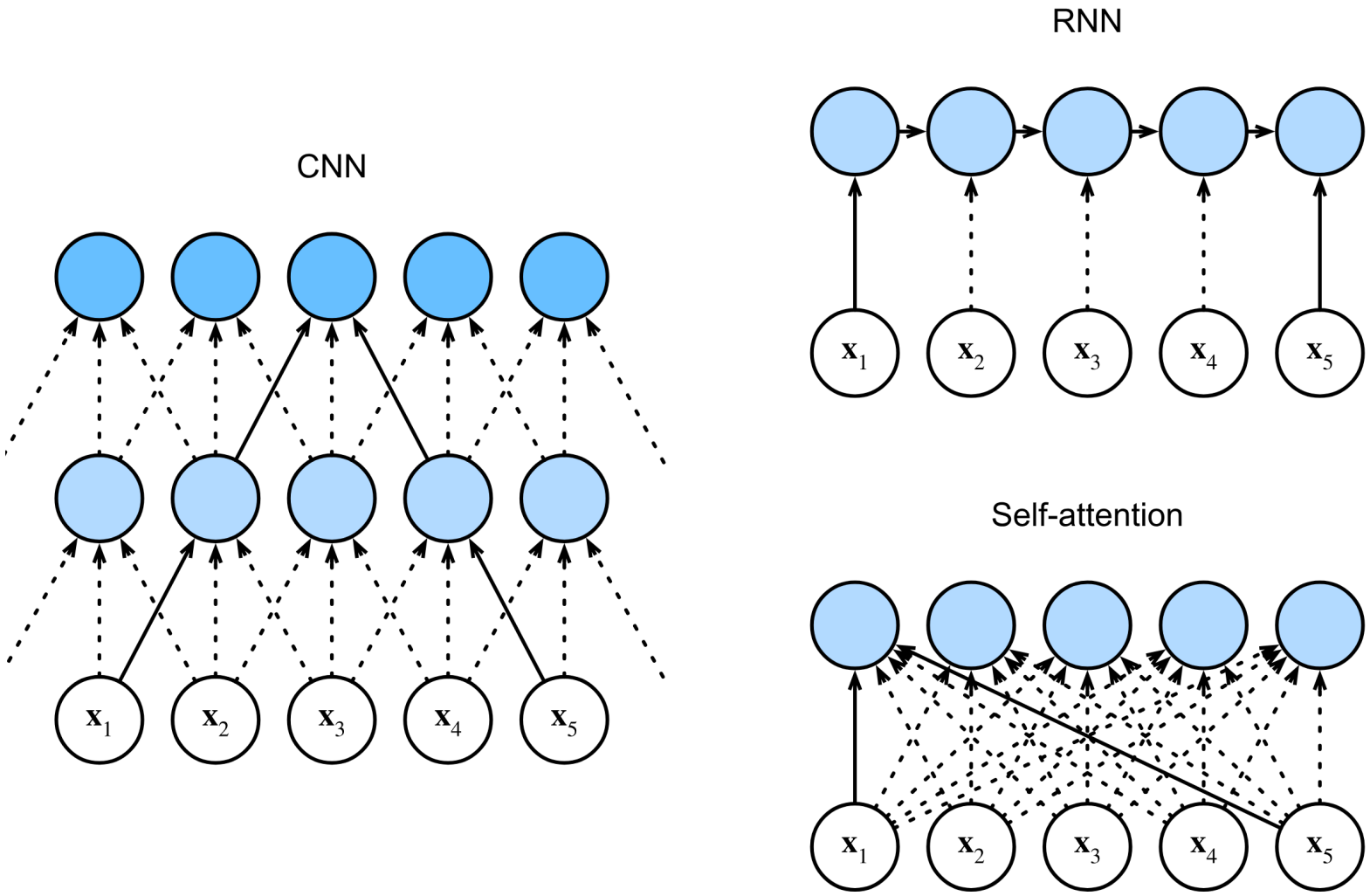
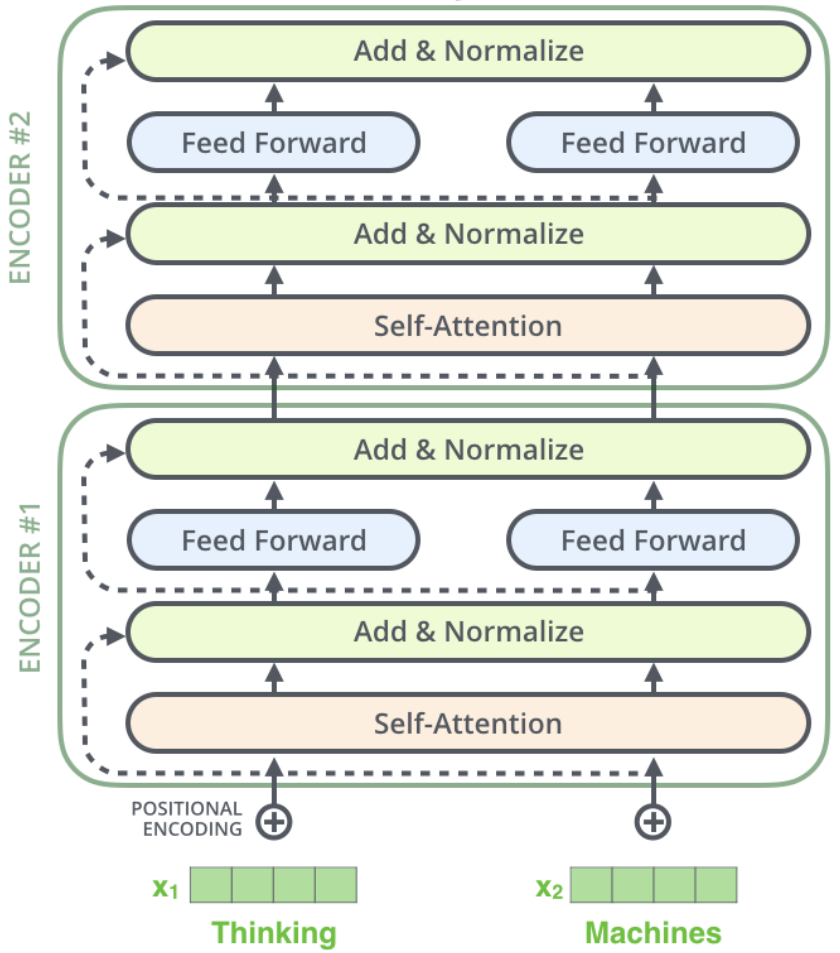


Fig: https://d2l.ai/chapter_attention-mechanisms-and-transformers/self-attention-and-positional-encoding.html

Previously on CENG501

Encoder



https://jalammar.github.io/illustrated-transformer/

Previously on CENG501

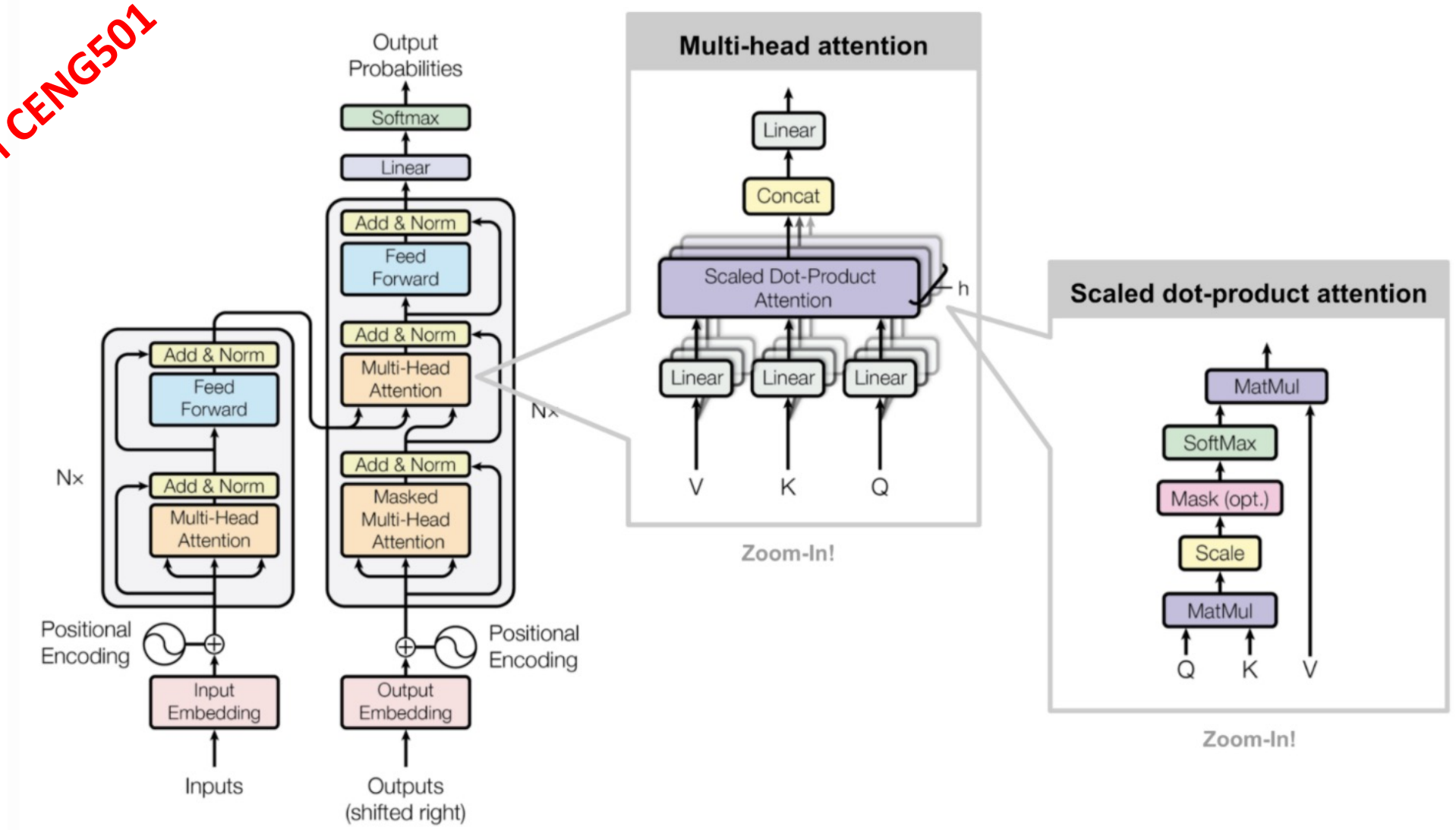
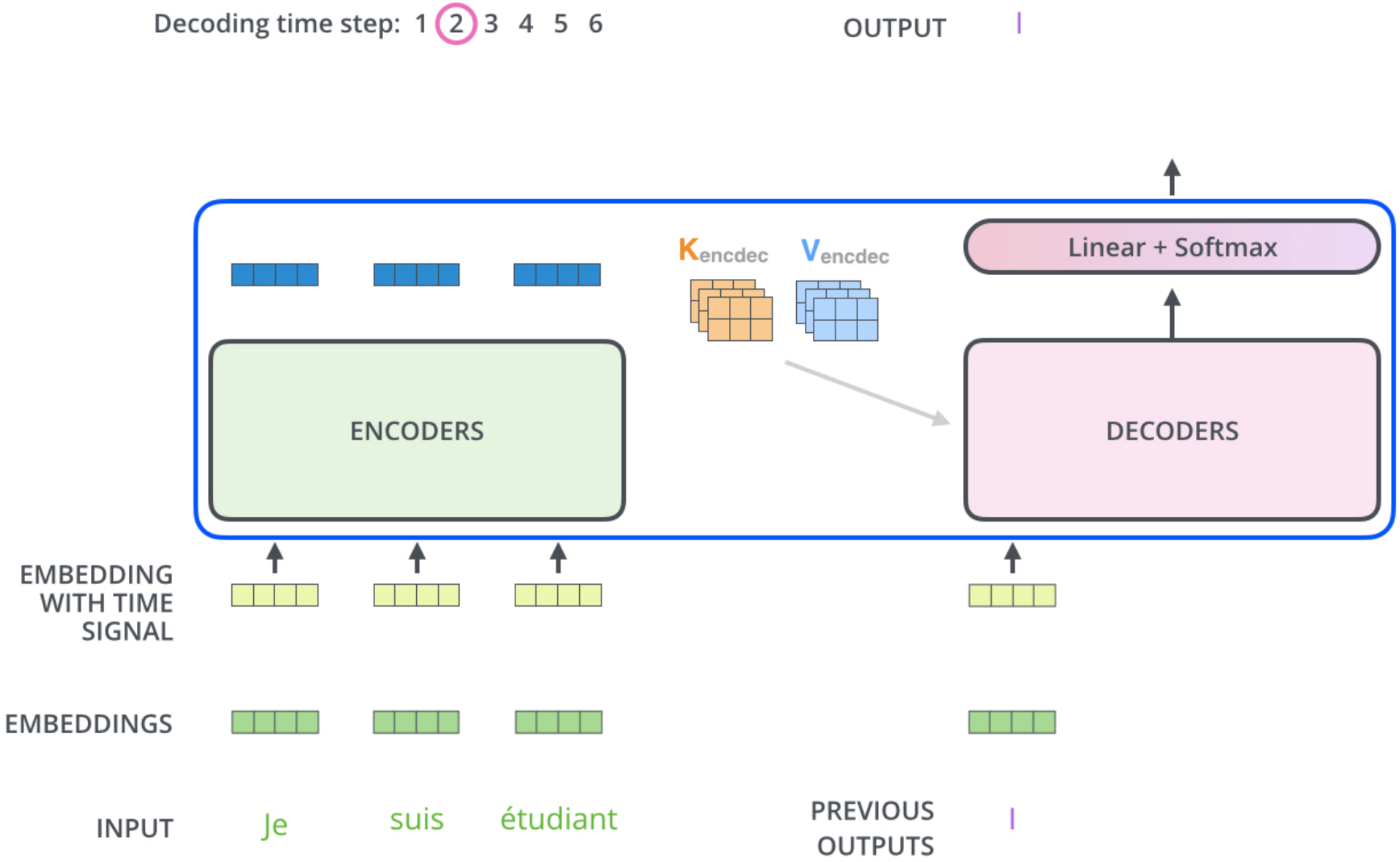


Fig. 17. The full model architecture of the transformer. (Image source: Fig 1 & 2 in Vaswani, et al., 2017.)

https://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html

Decoder

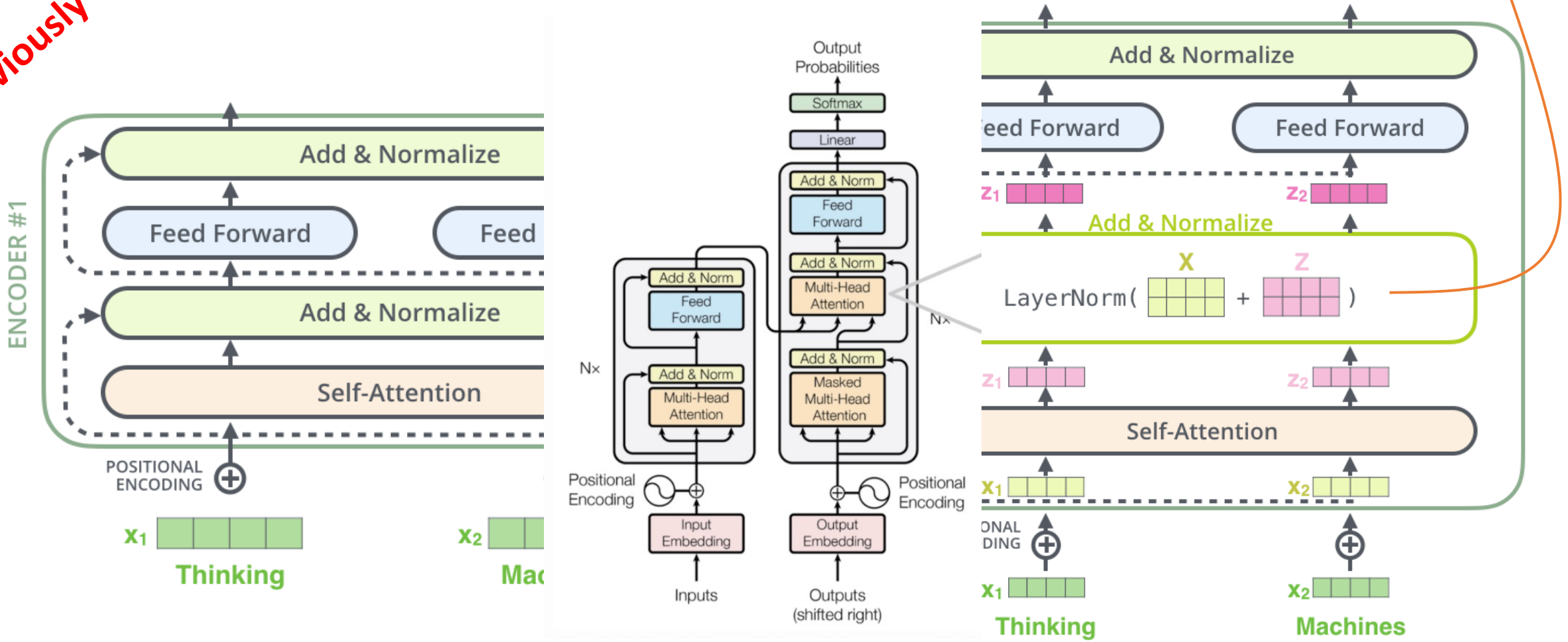
Previously on CENG501



Skip Connections & Normalization

Previously on CENG501

LayerNorm applied to each embedding independently



https://jalammar.github.io/illustrated-transformer/

Previously on CENG501

Byte-pair Encoding (BPE)

Example from: https://huggingface.co/learn/llm-course/en/chapter6/5

- Represent frequent byte-pairs as tokens
- E.g., given the corpus:

  Corpus (“word”, frequency): ("hug", 10), ("pug", 5), ("pun", 12), ("bun", 4), ("hugs", 5)

  our vocabulary would be:

  ("h" "u" "g", 10), ("p" "u" "g", 5), ("p" "u" "n", 12), ("b" "u" "n", 4), ("h" "u" "g" "s", 5)
- “ug” and “un” can be recognized to be very frequent. So, combine them:

  Vocabulary: ["b", "g", "h", "n", "p", "s", "u", "ug", "un", "hug"]

  Corpus: ("hug", 10), ("p" "ug", 5), ("p" "un", 12), ("b" "un", 4), ("hug" "s", 5)

Previously on CENG501

Comparison among Embeddings

Method	Out-of-Vocabulary Generalization	Vocabulary Size	Sub-word Semantics	Language Specificity	Sequence Length
Character-level	Excellent	Very small	Weak	Language agnostic	Very long
Word-level	Poor	Very large	None	Language specific	Short (one token per word)
Byte-pair	Very good	Medium/flexible (user-defined)	Very good	Medium	Longer than word-level
WordPiece	Good (slightly worse than BPE)	Medium/flexible (user-defined)	Very good	Medium	Longer than word-level

Previously on CSENG501

Positional Encoding: Motivation

important to distinguish:

“man bites dog”

vs

“dog bites man”

Self – attention is unaware about positions:

$$e_i' = \sum_j \frac{\exp\left(k\left(e_j^T\right)q(e_i)\right)}{\sum_m \exp(k(e_m^T)q(e_i))} v(e_j)$$

	Index 2	Index 1	Index 0
Embedding 0	0	0	0
Embedding 1	0	0	1
Embedding 2	0	1	0
Embedding 3	0	1	1
Embedding 4	1	0	0
Embedding 5	1	0	1
Embedding 6	1	1	0
Embedding 7	1	1	1

Previously on CENG501

Positional Encoding: Alternatives

Important to distinguish: “man bites dog” vs “dog bites man”

- Hand-crafted position embeddings (using the sin function)
- Learnable position embeddings
- Relative position embeddings
- Rotary positional embedding (ROPE)

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$

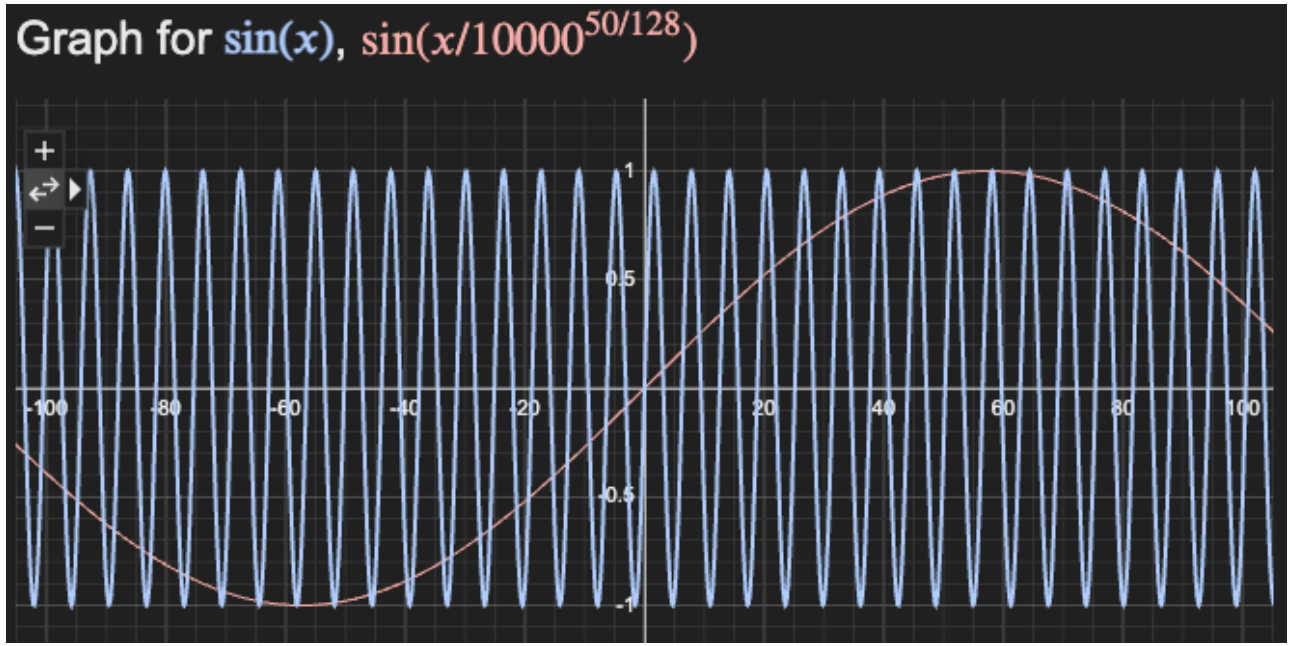
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

Position Encoding: More Details

$$PE_{(pos,2i)} = \sin(pos \cdot \omega_i)$$

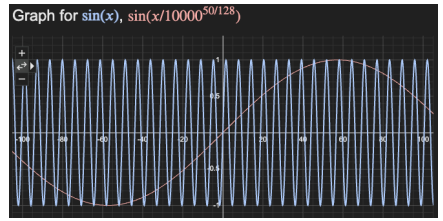
$$PE_{(pos,2i+1)} = \cos(pos \cdot \omega_i)$$

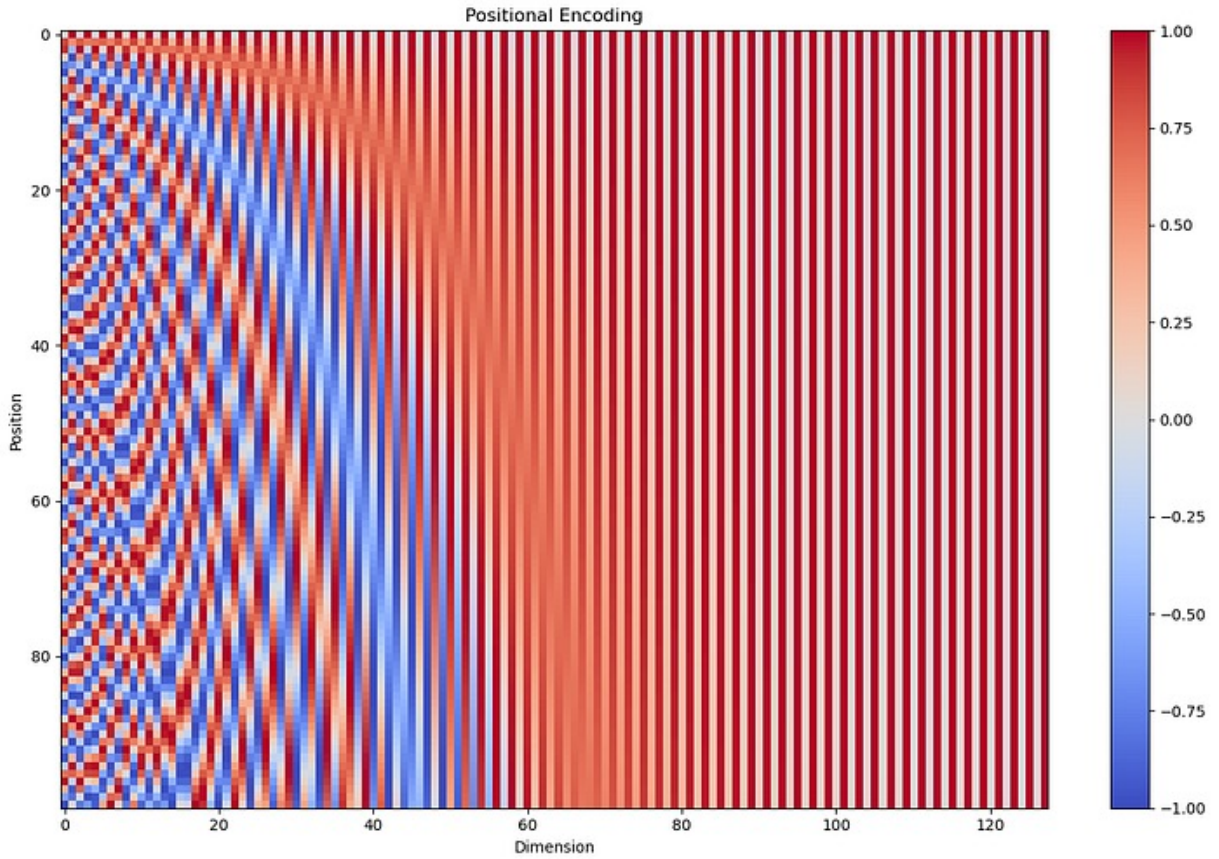
$$\omega_i = \frac{1}{10000^{2i/d_{model}}}$$ is the frequency.



```
import numpy as np
import matplotlib.pyplot as plt

def sinusoidal_positional_encoding(max_position, d_model):
    position = np.arange(max_position)[:, np.newaxis]
    # The original formula pos / 10000^(2i/d_model) is equivalent to pos * (1 / 10000^(2i/d_model)).
    # I use the below version for numerical stability
    div_term = np.exp(np.arange(0, d_model, 2) * -(np.log(10000.0) / d_model))

    pe = np.zeros((max_position, d_model))
    pe[:, 0::2] = np.sin(position * div_term)
    pe[:, 1::2] = np.cos(position * div_term)

    return pe

max_position = 100 # Maximum sequence length
d_model = 128 # Embedding dimension

pe = sinusoidal_positional_encoding(max_position, d_model)
```

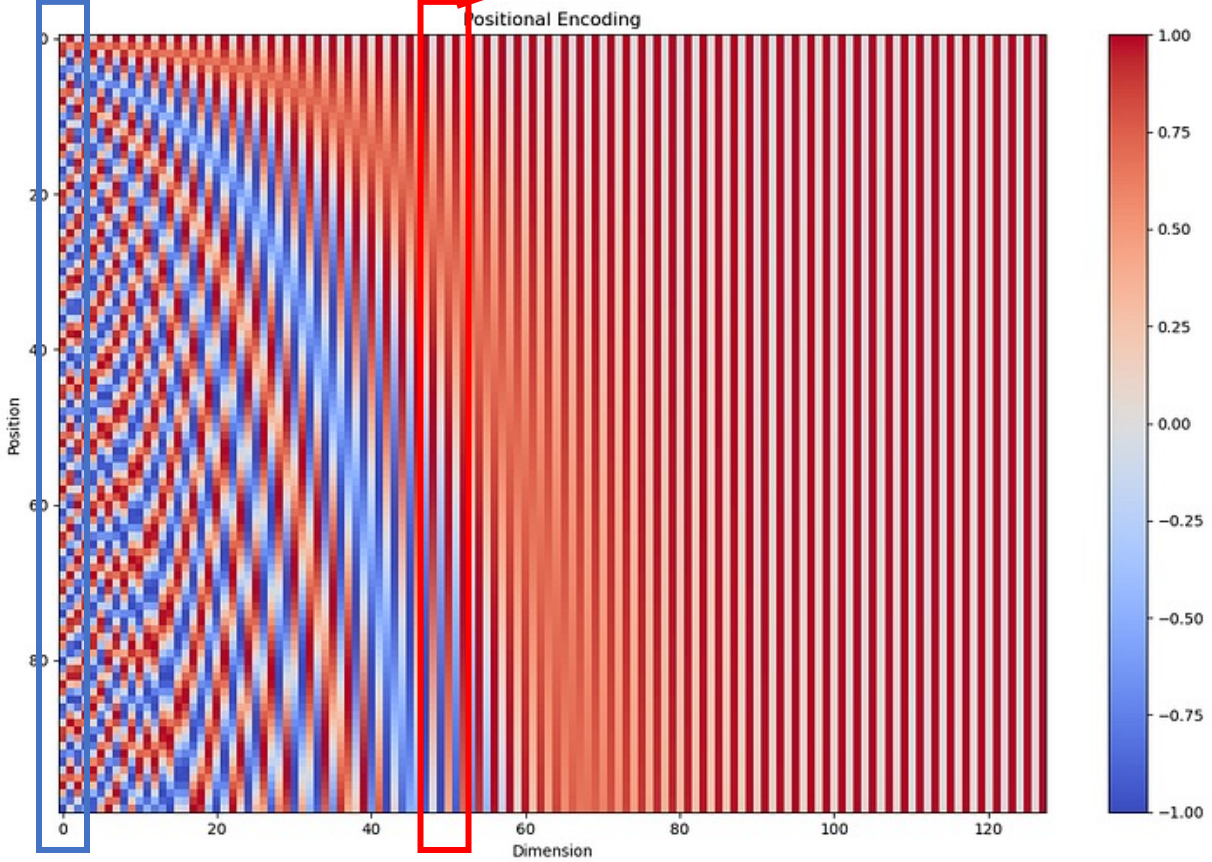
$$PE_{(pos,i)} = \begin{cases} \sin\left(\frac{pos}{10000^{i/d_{model}}}\right) & \text{if } i \text{ is even} \\ \cos\left(\frac{pos}{10000^{(i-1)/d_{model}}}\right) & \text{if } i \text{ is odd} \end{cases}$$

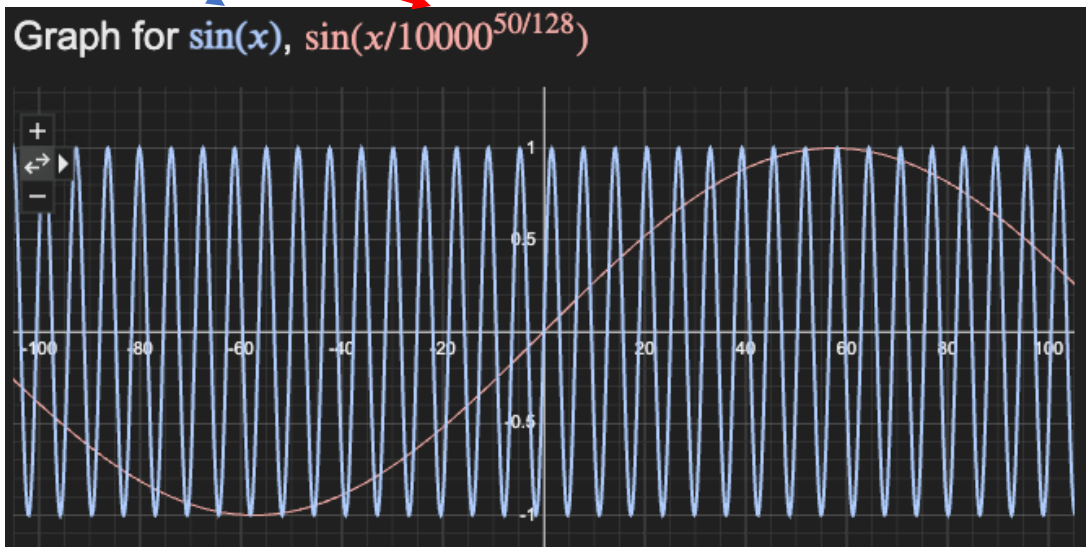




Material from: https://medium.com/thedeephub/positional-encoding-explained-a-deep-dive-into-transformer-pe-65cfe8cfe10b

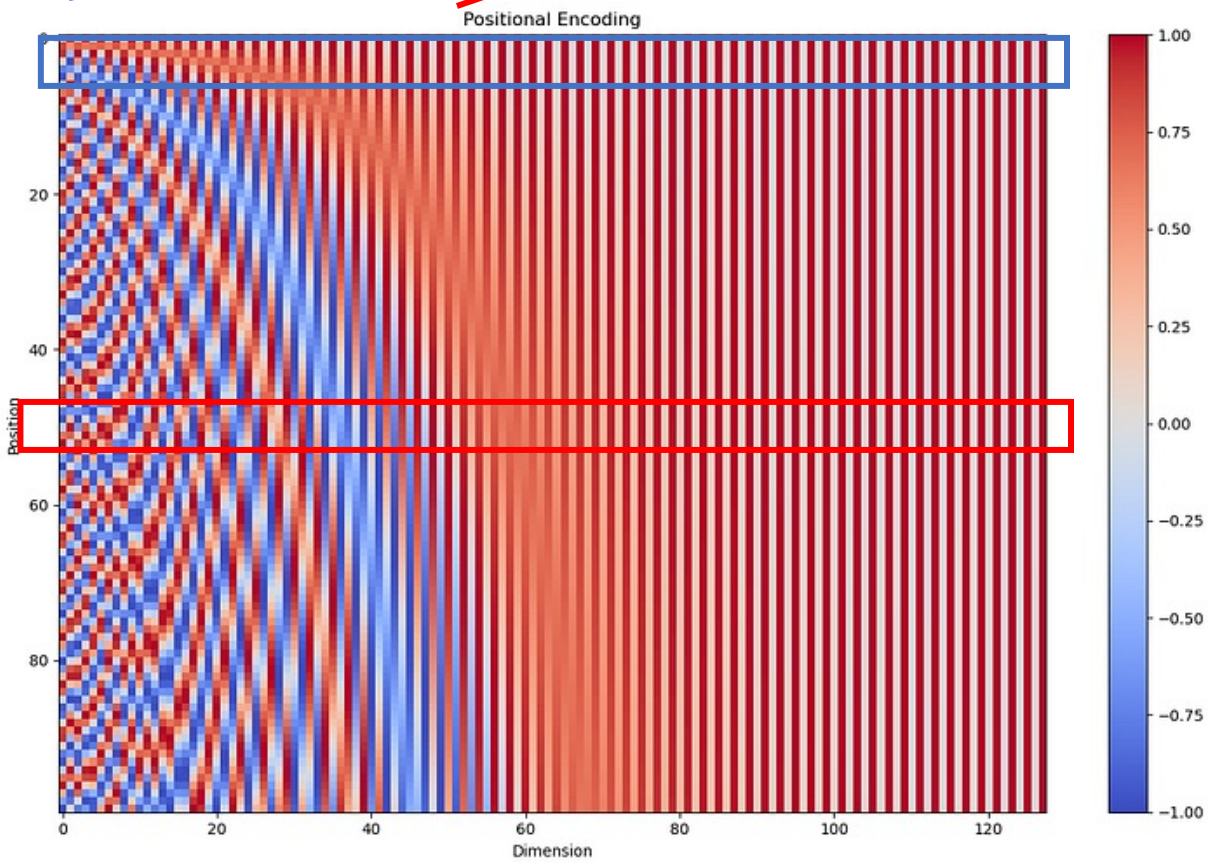
$$PE_{(pos,i)} = \begin{cases} \sin\left(\frac{pos}{10000^{i/d_{model}}}\right) & \text{if } i \text{ is even} \\ \cos\left(\frac{pos}{10000^{(i-1)/d_{model}}}\right) & \text{if } i \text{ is odd} \end{cases}$$

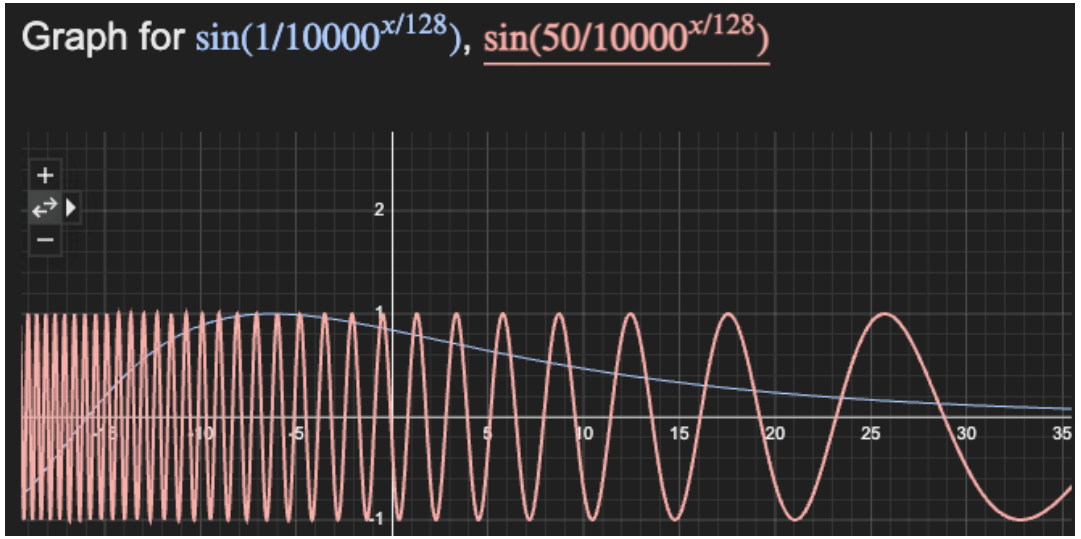




Material from: https://medium.com/thedeephub/positional-encoding-explained-a-deep-dive-into-transformer-pe-65cfe8cfe10b

$$PE_{(pos,i)} = \begin{cases} \sin\left(\frac{pos}{10000^{i/d_{model}}}\right) & \text{if } i \text{ is even} \\ \cos\left(\frac{pos}{10000^{(i-1)/d_{model}}}\right) & \text{if } i \text{ is odd} \end{cases}$$





What happens if pos is zero?

Material from: https://medium.com/thedeephub/positional-encoding-explained-a-deep-dive-into-transformer-pe-65cfe8cfe10b

Position Encoding: More Details

$$PE_{(pos,2i)} = \sin(pos \cdot \omega_i)$$

$$PE_{(pos,2i+1)} = \cos(pos \cdot \omega_i)$$

$\omega_i = \frac{1}{10000^{2i/d_{model}}}$ is the frequency.

This approach provides relative positional information as well:

- Assume a token at $pos + k$.
- Standard trigonometry tells us:

$$\sin(A + B) = \sin A \cos B + \cos A \sin B$$

$$\cos(A + B) = \cos A \cos B - \sin A \sin B$$

- Using these for $pos + k$:

$$PE_{(pos+k,2i)} = \sin(pos \cdot \omega_i)\cos(k \cdot \omega_i) + \cos(pos \cdot \omega_i)\sin(k \cdot \omega_i)$$

$$PE_{(pos+k,2i+1)} = \cos(pos \cdot \omega_i)\cos(k \cdot \omega_i) - \sin(pos \cdot \omega_i)\sin(k \cdot \omega_i)$$

- Which can be written as a matrix multiplication:

$$\begin{bmatrix} PE_{(pos+k,2i)} \\ PE_{(pos+k,2i+1)} \end{bmatrix} = \begin{bmatrix} \cos(k \cdot \omega_i) & \sin(k \cdot \omega_i) \\ -\sin(k \cdot \omega_i) & \cos(k \cdot \omega_i) \end{bmatrix} \begin{bmatrix} PE_{(pos,2i)} \\ PE_{(pos,2i+1)} \end{bmatrix}$$

- This tells us: The PE of $pos + k$ can be written in terms of PE of pos.
- Self-attention between embeddings at these positions reduces to a simple term that depends on k:

$$PE_{pos} \cdot PE_{pos+k} = \sum_{i=0}^{d/2-1} \cos(k \cdot \omega_i)$$

Absolute vs. Relative Positional Encoding

- If the sinusoidal positional encoding can encode relative positions, why need relative positional encoding methods?

- In theory, the sinusoidal PE can encode relative positions. However, it is harder to deduce relative positions:

$$Score_{i,j} = \Big((e_i + p_i)W_Q\Big) \cdot \Big((e_j + p_j)W_K\Big)$$

  - This translates to:
    - $(e_i W_Q) \cdot (e_j W_K)$: Semantic similarity
    - $(p_i W_Q) \cdot (p_j W_K)$: Position similarity. Encodes relative distance $i - j$.
    - $(e_i W_Q) \cdot (p_j W_K)$: Depends on absolute position.
    - $(p_i W_Q) \cdot (e_j W_K)$: Depends on absolute position.

Previously on CENG501

Positional Encoding: Alternatives

- Hand-crafted position embeddings (using the sin function)
- Learnable position embeddings
- Relative position embeddings
- Rotary positional embedding (ROPE)

Define position embedding matrix as a learnable tensor (each e_{ij} is a learnable parameter):

	Index 2	Index 1	Index 0
Embedding 0	e_{02}	e_{01}	e_{00}
Embedding 1	e_{12}	e_{11}	e_{10}
Embedding 2	e_{22}	e_{21}	e_{20}
Embedding 3	e_{32}	e_{31}	e_{30}
Embedding 4	e_{42}	e_{41}	e_{40}
Embedding 5	e_{52}	e_{51}	e_{50}
Embedding 6	e_{62}	e_{61}	e_{50}
Embedding 7	e_{72}	e_{71}	e_{70}

Previously on CENG501

Positional Encoding: Alternatives

- Hand-crafted position embeddings (using the sin function)
- Learnable position embeddings
- Relative position embeddings
- Rotary positional embedding (ROPE)

Self-Attention with Standard Position Embeddings

$$z_i = \sum_{j=1}^{n} \alpha_{ij}(x_j W^V) \qquad \alpha_{ij} = \frac{\exp e_{ij}}{\sum_{k=1}^{n} \exp e_{ik}} \qquad e_{ij} = \frac{(x_i W^Q)(x_j W^K)^T}{\sqrt{d_z}}$$

Self-Attention with Relative Position Embeddings

$$z_i = \sum_{j=1}^{n} \alpha_{ij}(x_j W^V + a_{ij}^V) \qquad \alpha_{ij} = \frac{\exp e_{ij}}{\sum_{k=1}^{n} \exp e_{ik}} \qquad e_{ij} = \frac{x_i W^Q (x_j W^K + a_{ij}^K)^T}{\sqrt{d_z}}$$

a_{ij} encodes relative positions between embeddings x_i and x_j.
w: learnable parameter.

$$a_{ij}^K = w^K_{\text{clip}(j-i,k)}$$
$$a_{ij}^V = w^V_{\text{clip}(j-i,k)}$$
$$\text{clip}(x,k) = \max(-k, \min(k,x))$$

Shaw et al., Self-Attention with Relative Position Representations, 2018.

Previously on CENG501

Positional Encoding: Alternatives

- Hand-crafted position embeddings (using the sin function)
- Learnable position embeddings
- Relative position embeddings
- Rotary positional embedding (ROPE)

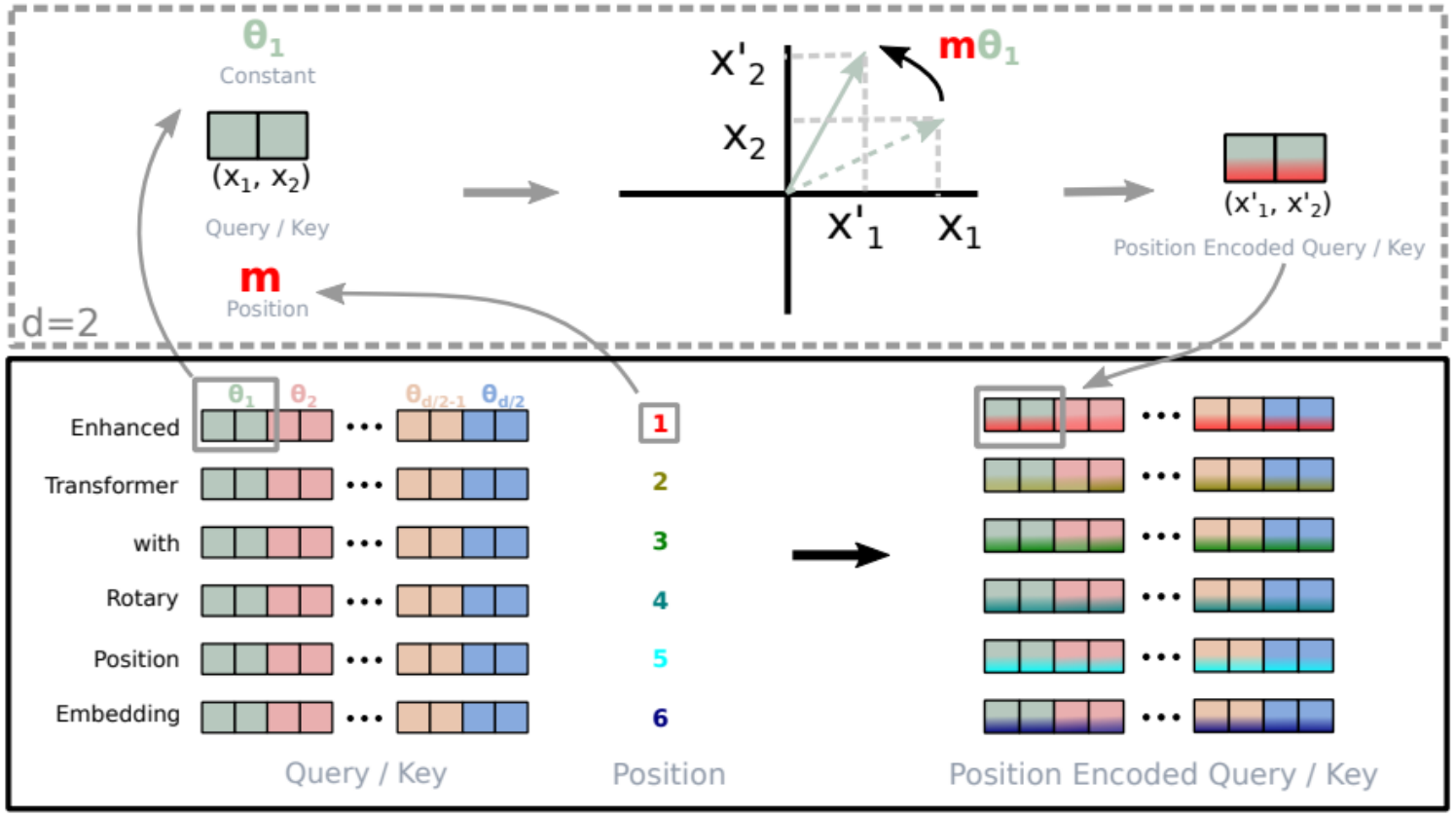


Fig: Su et al., RoFormer: Enhanced Transformer with Rotary Position Embedding, 2021.

$$\theta_i = 10000^{-2(i-1)/d}, i \in [1, 2, ..., d/2]$$

Previously on CENG501

A Significant Issue with Self-Attention: Complexity

$$e_i' = \sum_j \frac{\exp\left(k\left(e_j^T\right)q(e_i)\right)}{\sum_m \exp(k(e_m^T)q(e_i))} v(e_j)$$

- If there are n tokens/embeddings,
  - Updating a single token requires $O(n)$ operations.
  - Overall: $O(n^2)$
- What is the complexity of an RNN layer with n time steps?

Previously on CENG501

Linear Attention

Self-attention:

$$
\begin{aligned}
Q &= xW_Q, \\
K &= xW_K, \\
V &= xW_V, \\
A_l(x) = V' &= \text{softmax}\left(\frac{QK^T}{\sqrt{D}}\right)V.
\end{aligned} \tag{2}
$$

Rewrite Eq 2 for one row of the matrix:

$$
V'_i = \frac{\sum_{j=1}^{N} \text{sim}\left(Q_i, K_j\right) V_j}{\sum_{j=1}^{N} \text{sim}\left(Q_i, K_j\right)}. \tag{3}
$$

Equation 3 is equivalent to equation 2 if we substitute the similarity function with $\text{sim}\left(q, k\right) = \exp\left(\frac{q^T k}{\sqrt{D}}\right)$.

ICML 2020

Transformers are RNNs:
Fast Autoregressive Transformers with Linear Attention

Angelos Katharopoulos [1 2] Apoorv Vyas [1 2] Nikolaos Pappas [3] François Fleuret [2 4 *]

Constraint for sim(): It should be non-negative.
Then, we can choose any other kernel/function:

Given such a kernel with a feature representation $\phi\left(x\right)$ we can rewrite equation 2 as follows,

$\phi\left(x\right) = \text{elu}(x) + 1$

$$
V'_i = \frac{\sum_{j=1}^{N} \phi\left(Q_i\right)^T \phi\left(K_j\right) V_j}{\sum_{j=1}^{N} \phi\left(Q_i\right)^T \phi\left(K_j\right)}, \tag{4}
$$

and then further simplify it by making use of the associative property of matrix multiplication to

$$
V'_i = \frac{\phi\left(Q_i\right)^T \sum_{j=1}^{N} \phi\left(K_j\right) V_j^T}{\phi\left(Q_i\right)^T \sum_{j=1}^{N} \phi\left(K_j\right)}. \tag{5}
$$

Today

- State-space Models
- Mamba
- Pretraining Deep Networks for NLP Tasks
- Large-Language Models

Administrative Notes

- Project next steps:
  - Milestones:
    1. Milestone (April 10, midnight):
       - Read & understand the paper
       - Download the datasets
       - Prepare the Readme file excluding the results & conclusion
    2. Milestone (May 4, midnight)
       - The results of the first experiment
    3. Milestone (June 1, midnight)
       - Final report (Readme file)
       - Repo with all code & trained models

Mamba: Linear-Time Sequence Modeling with Selective State Spaces

Albert Gu*[1] and Tri Dao*[2]

[1]Machine Learning Department, Carnegie Mellon University

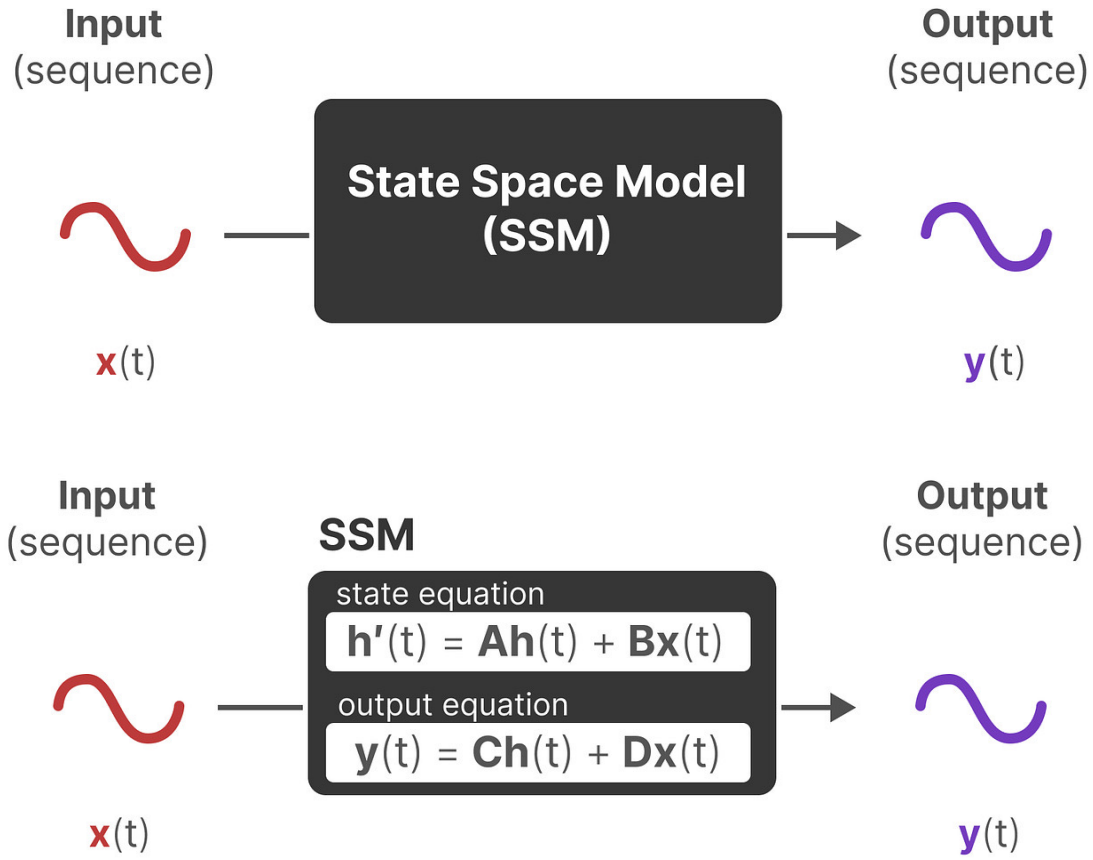
[2]Department of Computer Science, Princeton University
agu@cs.cmu.edu, tri@tridao.me

Rejected at ICLR2024

Structured State Space Sequence (S4) Model

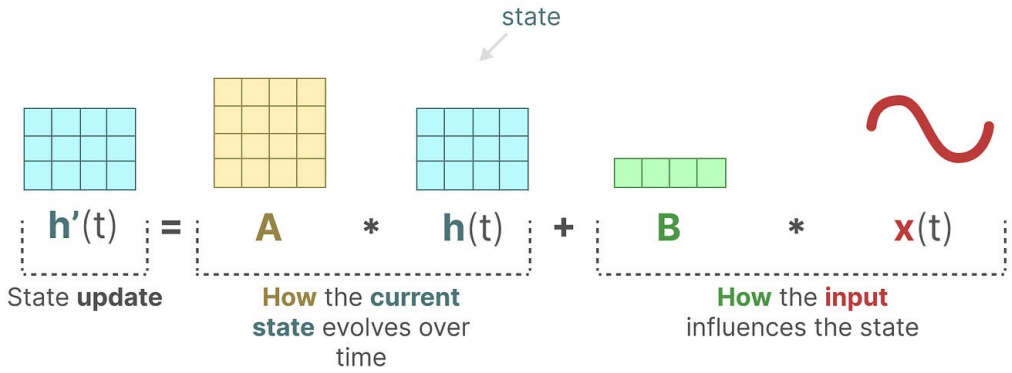
State Space Models (SSMs)

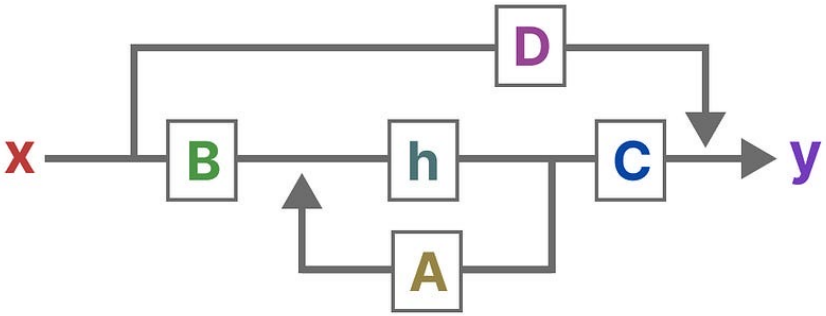
- Notation:
  - x(t): input (e.g., observation)
  - h(t): latent state representation
  - y(t): predicted output
- State update equation:
  - $h'(t) = A\,h(t) + B\,x(t)$
- Output equation:
  - $y(t) = C\,h(t) + D\,x(t)$
- A, B, C, D: learnable params

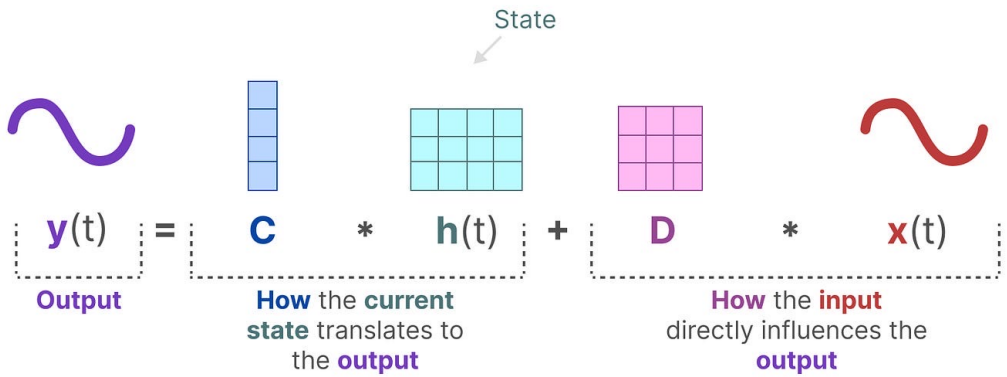


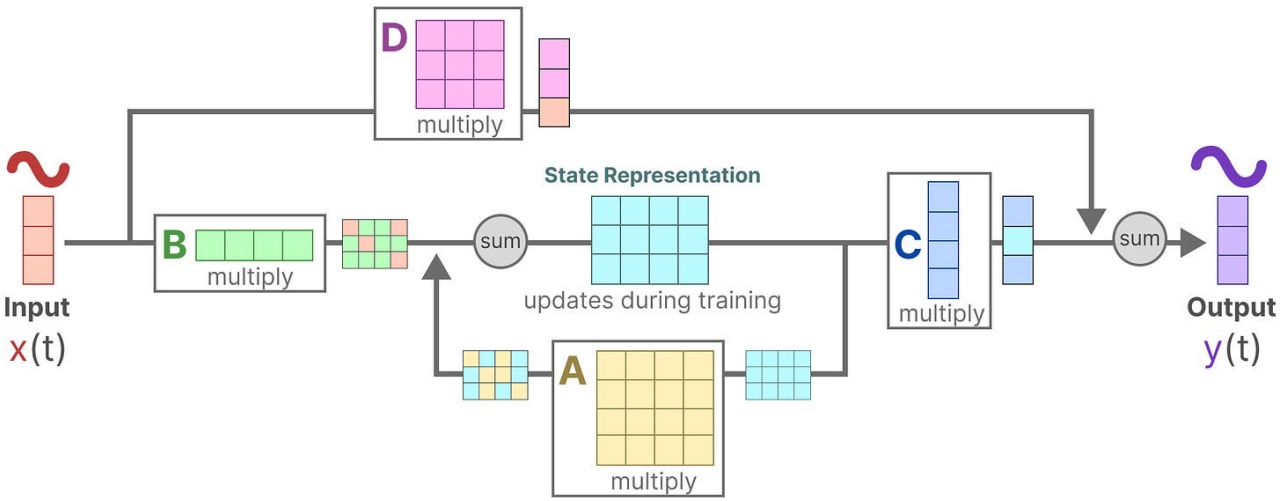
https://newsletter.maartengrootendorst.com/p/a-visual-guide-to-mamba-and-state

State Space Models (SSMs)



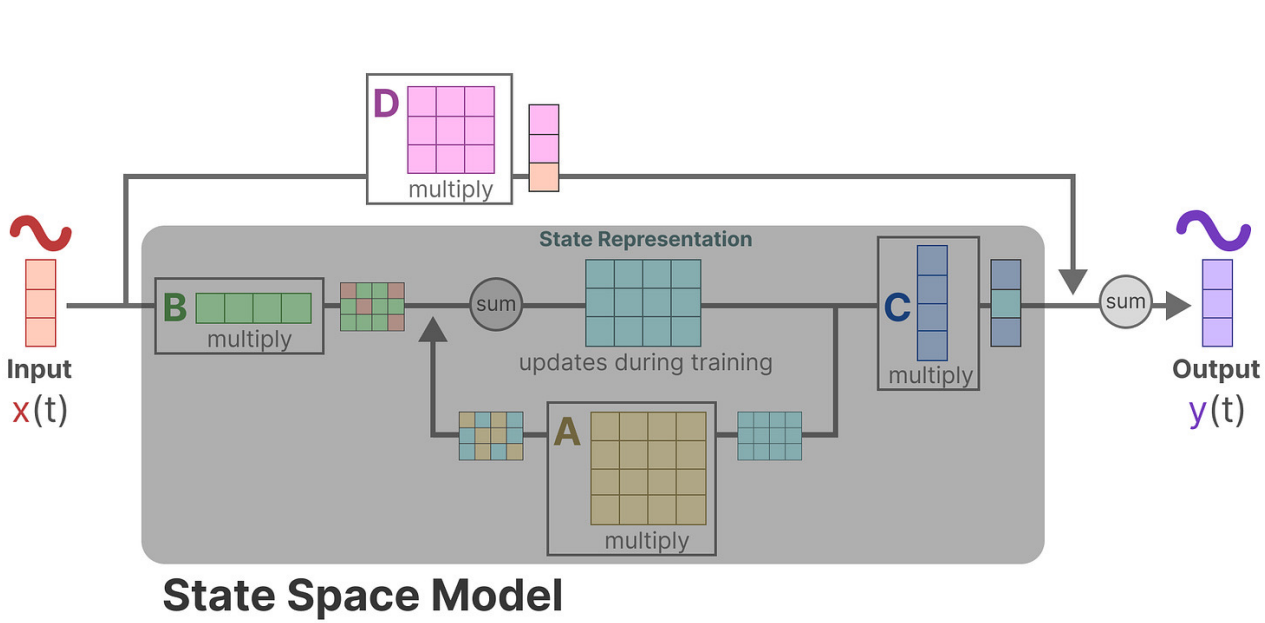


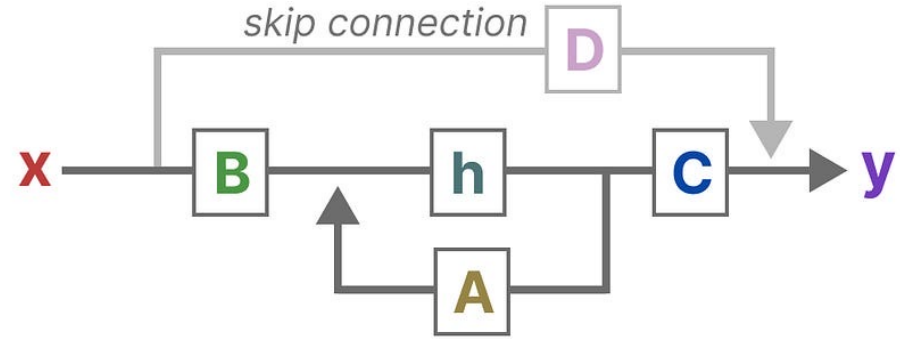


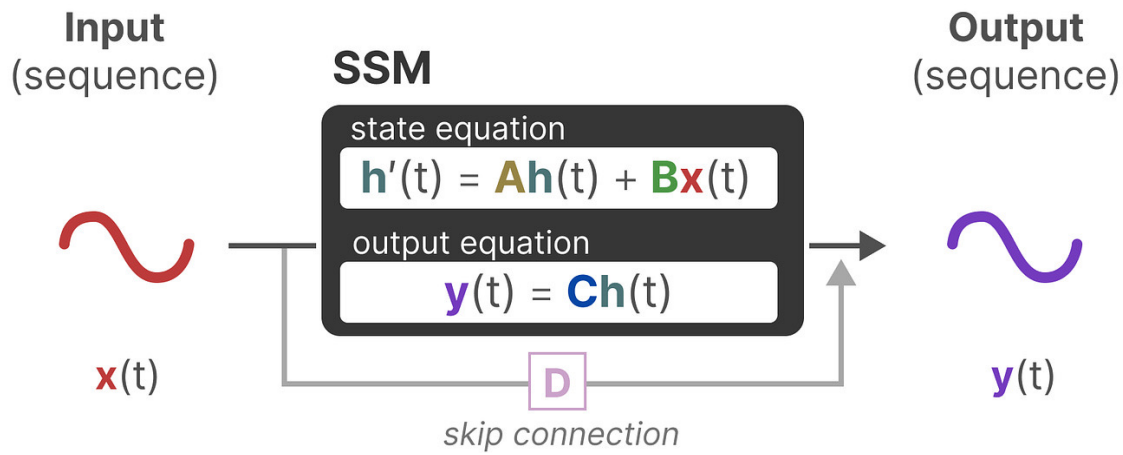


https://newsletter.maartengrootendorst.com/p/a-visual-guide-to-mamba-and-state

State Space Models (SSMs)



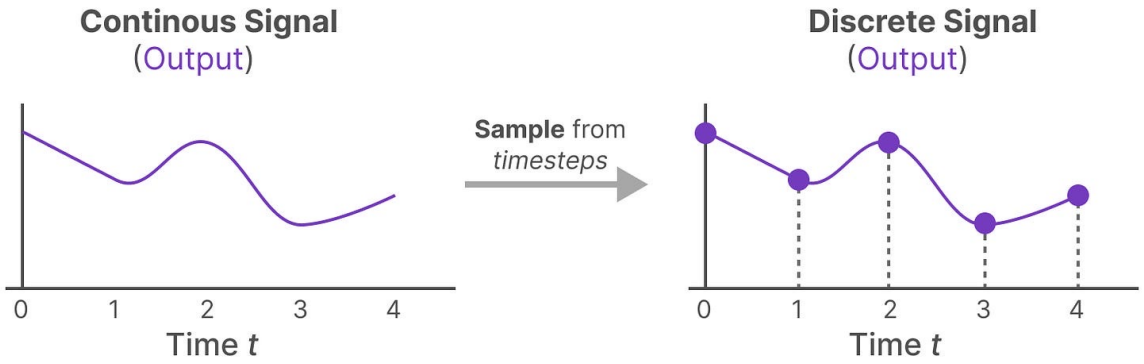


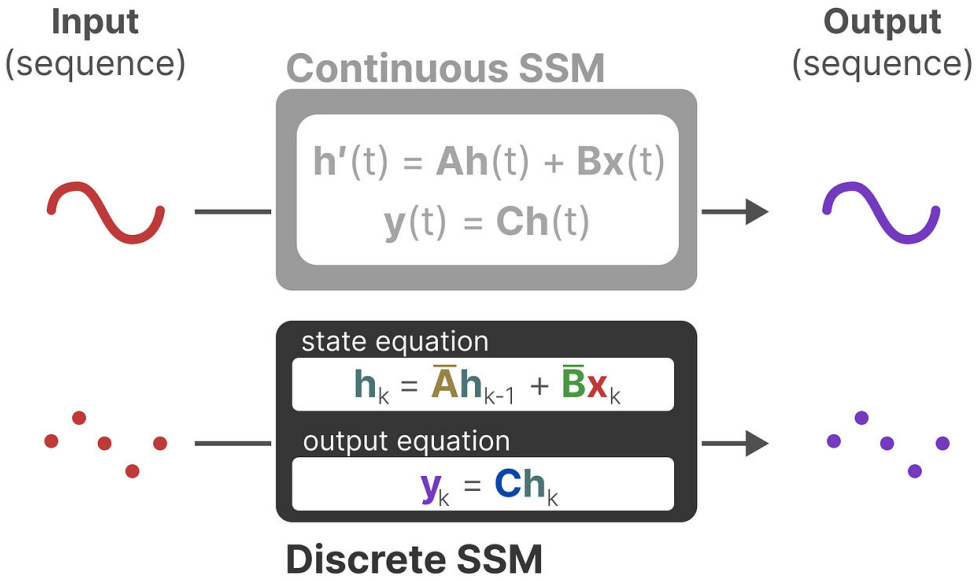


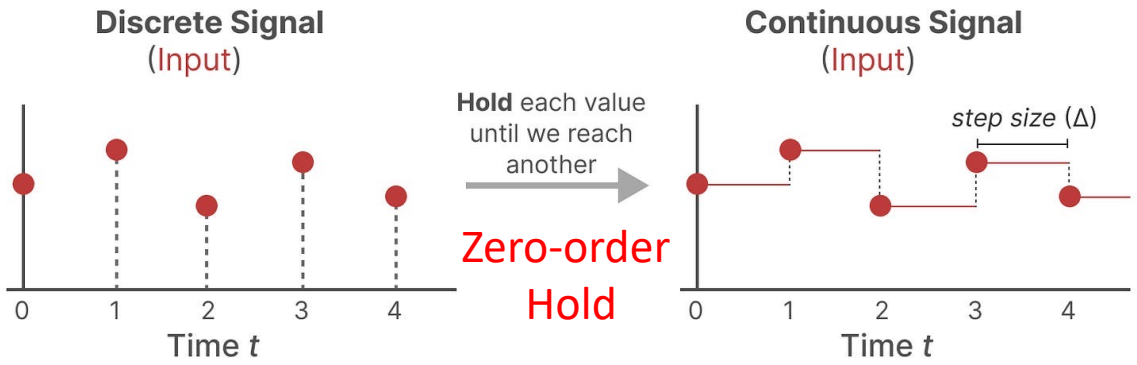
https://newsletter.maartengrootendorst.com/p/a-visual-guide-to-mamba-and-state

State Space Models (SSMs)

- Convert discrete signal to a continuous signal
- Obtain a continuous output
- Convert the continuous output to a discrete signal







Δ: Hold interval -- Learnable

Discretized matrix **A**

$$\bar{A} = \exp(\Delta A)$$

Discretized matrix **B**

$$\bar{B} = (\Delta A)^{-1}(\exp(\Delta A) - I) \cdot \Delta B$$

https://newsletter.maartengrootendorst.com/p/a-visual-guide-to-mamba-and-state

To discretize the continuous case, let's use the trapezoid method where the principle is to assimilate the region under the representative curve of a function f defined on a segment $[t_n, t_{n+1}]$ to a trapezoid and calculate its area $T : T = (t_{n+1} - t_n)\frac{f(t_n)+f(t_{n+1})}{2}$.

We then have: $x_{n+1} - x_n = \frac{1}{2}\Delta(f(t_n) + f(t_{n+1}))$ with $\Delta = t_{n+1} - t_n$.
If $x'_n = \mathbf{A}x_n + \mathbf{B}u_n$ (first line of the SSM equation), corresponds to f, so:

$$
\begin{aligned}
x_{n+1} &= x_n + \frac{\Delta}{2}(\mathbf{A}x_n + \mathbf{B}u_n + \mathbf{A}x_{n+1} + \mathbf{B}u_{n+1}) \\
\iff x_{n+1} - \frac{\Delta}{2}\mathbf{A}x_{n+1} &= x_n + \frac{\Delta}{2}\mathbf{A}x_n + \frac{\Delta}{2}\mathbf{B}(u_{n+1} + u_n) \\
(*) \iff (\mathbf{I} - \frac{\Delta}{2}\mathbf{A})x_{n+1} &= (\mathbf{I} + \frac{\Delta}{2}\mathbf{A})x_n + \Delta\mathbf{B}u_{n+1} \\
\iff x_{n+1} &= (\mathbf{I} - \frac{\Delta}{2}\mathbf{A})^{-1}(\mathbf{I} + \frac{\Delta}{2}\mathbf{A})x_n + (\mathbf{I} - \frac{\Delta}{2}\mathbf{A})^{-1}\Delta\mathbf{B}u_{n+1}
\end{aligned}
$$

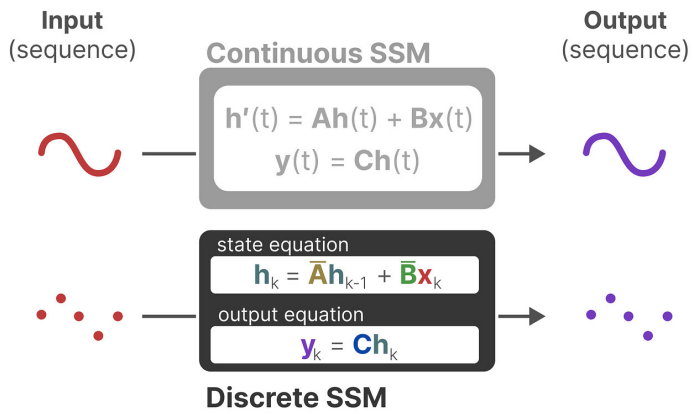
(*) $u_{n+1} \overset{\Delta}{\simeq} u_n$ (the control vector is assumed to be constant over a small Δ).

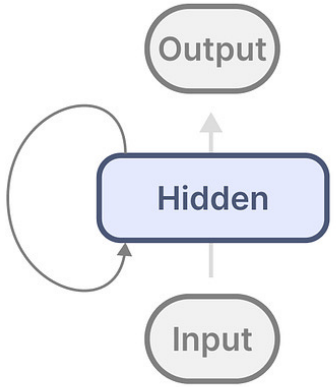
We've just obtained our discretized SSM!
To make this completely explicit, let's pose :

$$
\begin{aligned}
\bar{\mathbf{A}} &= (\mathbf{I} - \frac{\Delta}{2}\mathbf{A})^{-1}(\mathbf{I} + \frac{\Delta}{2}\mathbf{A}) \\
\bar{\mathbf{B}} &= (\mathbf{I} - \frac{\Delta}{2}\mathbf{A})^{-1}\Delta\mathbf{B} \\
\bar{\mathbf{C}} &= \mathbf{C}
\end{aligned}
$$

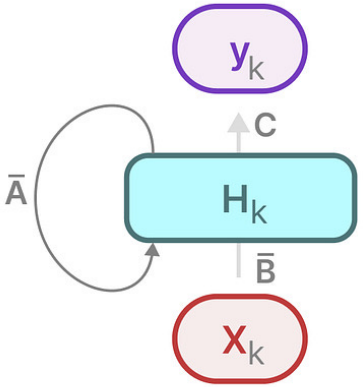
https://huggingface.co/blog/lbourdois/get-on-the-ssm-train

State Space Models (SSMs)

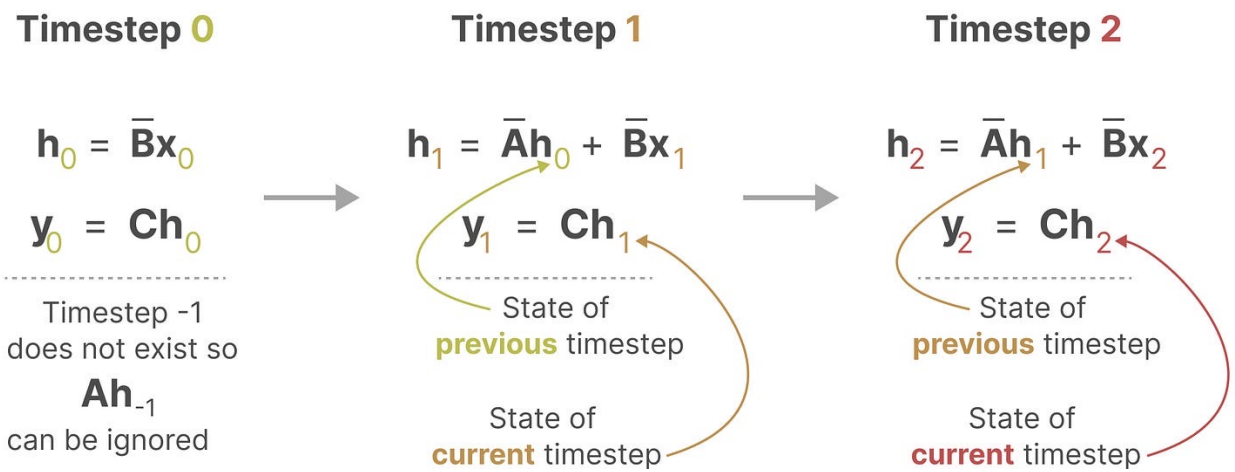


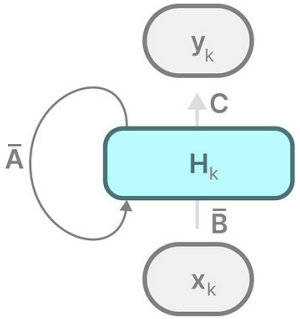


RNN

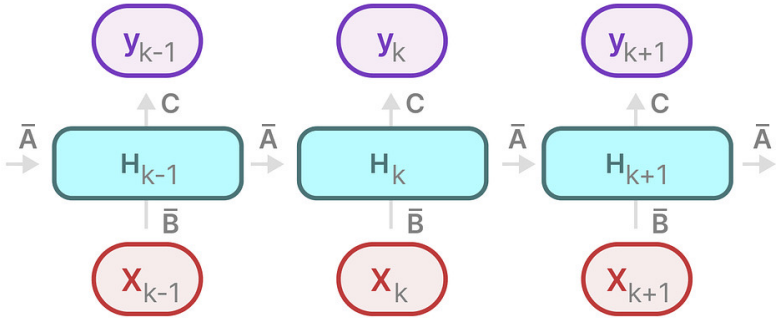


SSM
(Recurrent)





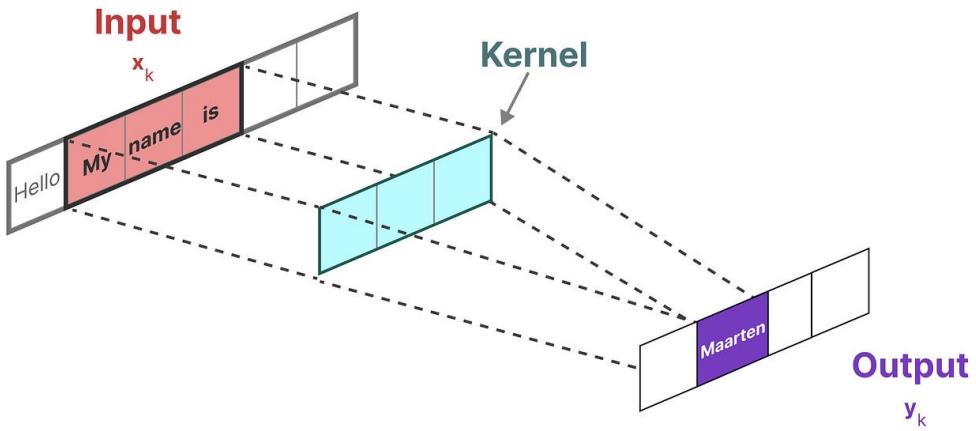
SSM
(Recurrent)

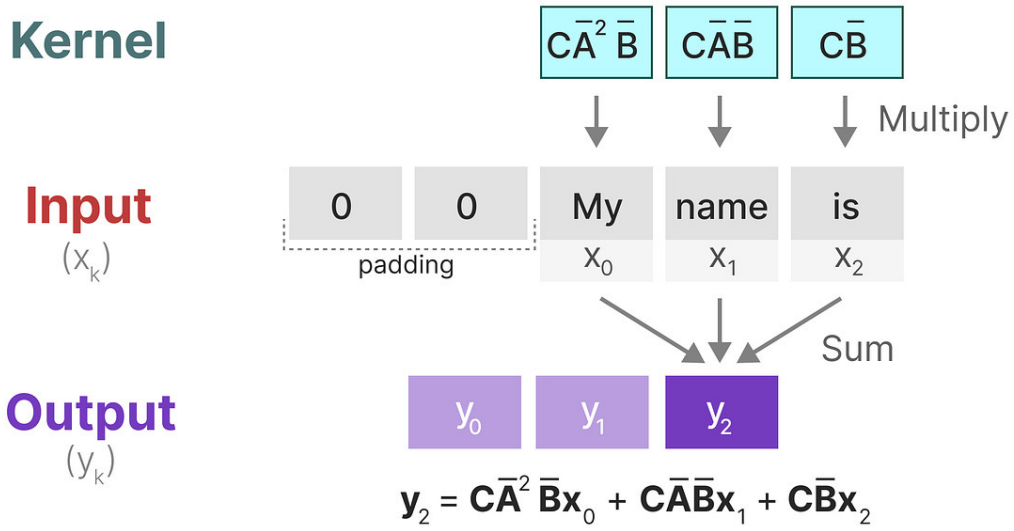


SSM
(Recurrent + Unfolded)

https://newsletter.maartengrootendorst.com/p/a-visual-guide-to-mamba-and-state

State Space Models (SSMs)





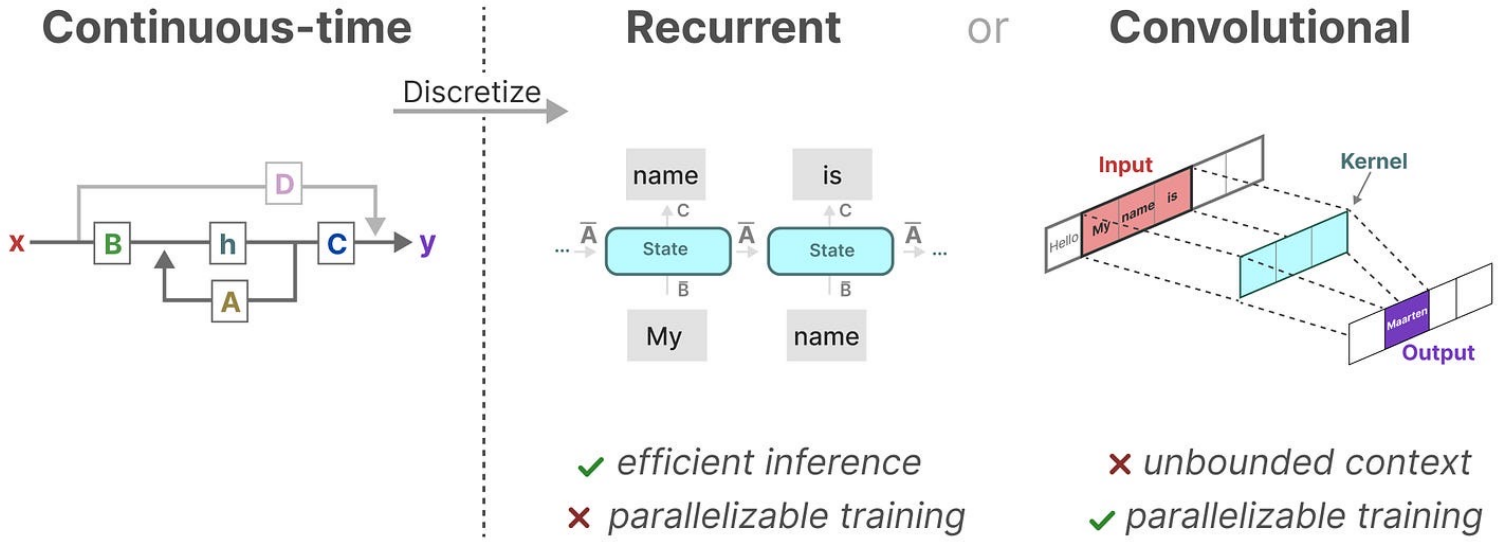
$$\text{kernel} \longrightarrow \bar{K} = (C\bar{B}, C\bar{A}\bar{B}, \ldots, C\bar{A}^{k}\bar{B}, \ldots)$$

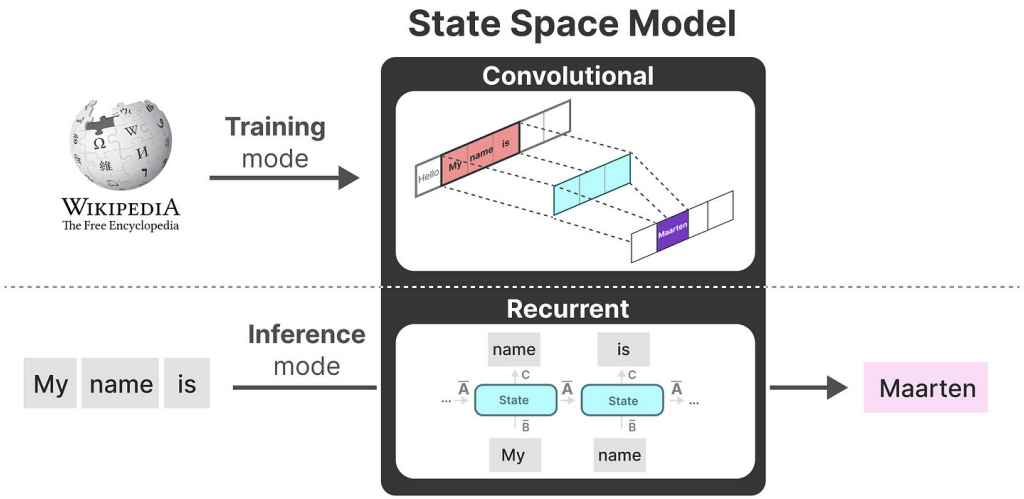
$$y = x * \bar{K}$$

output input kernel

https://newsletter.maartengrootendorst.com/p/a-visual-guide-to-mamba-and-state

State Space Models (SSMs)





These representations share an important property, namely that of ***Linear Time Invariance*** (LTI). LTI states that the SSMs parameters, A, B, and C, are fixed for all timesteps. This means that matrices A, B, and C are the same for every token the SSM generates.

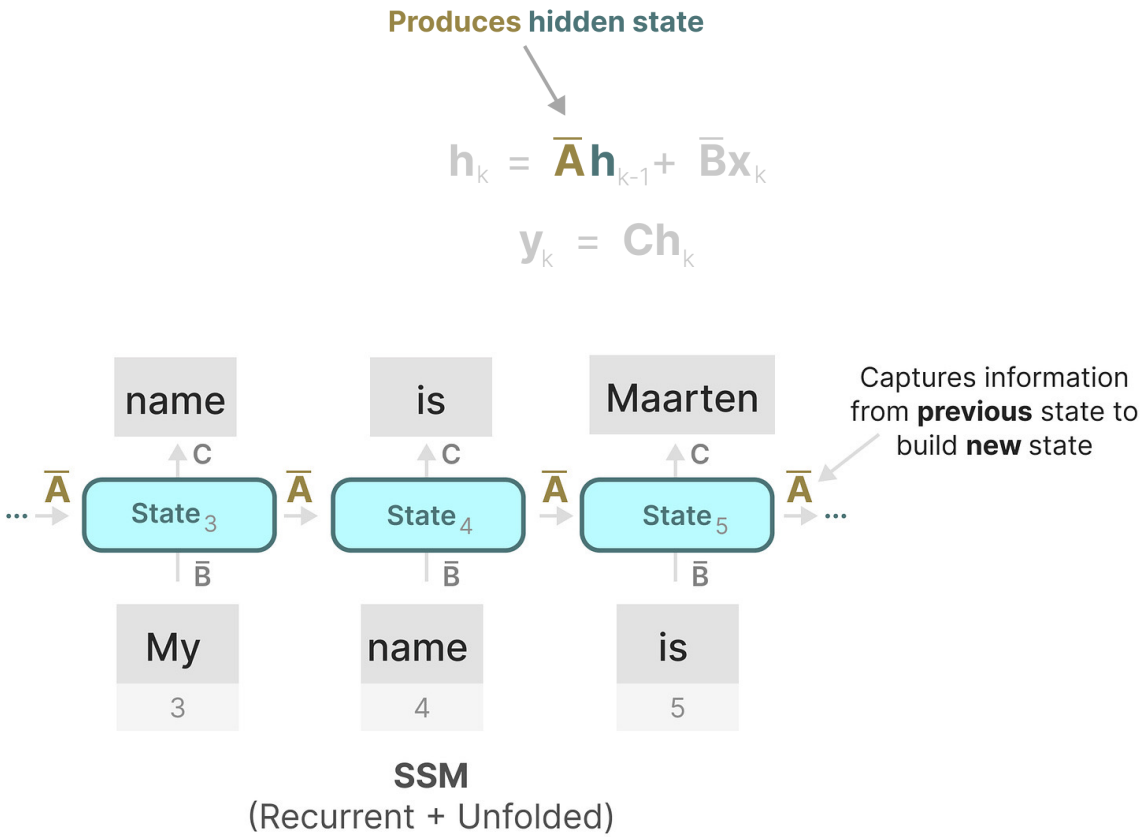
In other words, regardless of what sequence you give the SSM, the values of A, B, and C remain the same. We have a static representation that is not content-aware.

https://newsletter.maartengrootendorst.com/p/a-visual-guide-to-mamba-and-state

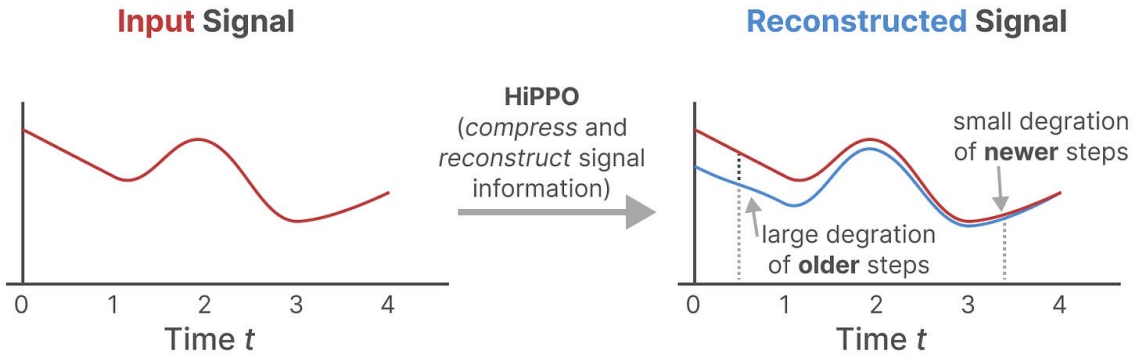
State Space Models (SSMs)

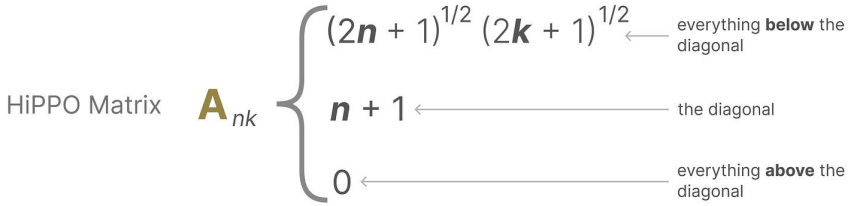
Gu, Albert, et al. "Hippo: Recurrent memory with optimal polynomial projections." Advances in neural information processing systems 33 (2020): 1474-1487.

"So how can we create matrix A in a way that retains a large memory (context size)?"



High-order **P**olynomial **P**rojection **O**perators (HIPPO)





HiPPO Matrix

1	0	0	0
1	2	0	0
1	3	3	0
1	3	5	4

k (rows) × *n* (columns)

https://newsletter.maartengrootendorst.com/p/a-visual-guide-to-mamba-and-state

State Space Models (SSMs)

https://srush.github.io/annotated-s4/

“Prior work found that the basic SSM actually performs very poorly in practice. Intuitively, one explanation is that they suffer from gradients scaling exponentially in the sequence length (i.e., the vanishing/exploding gradients problem)."

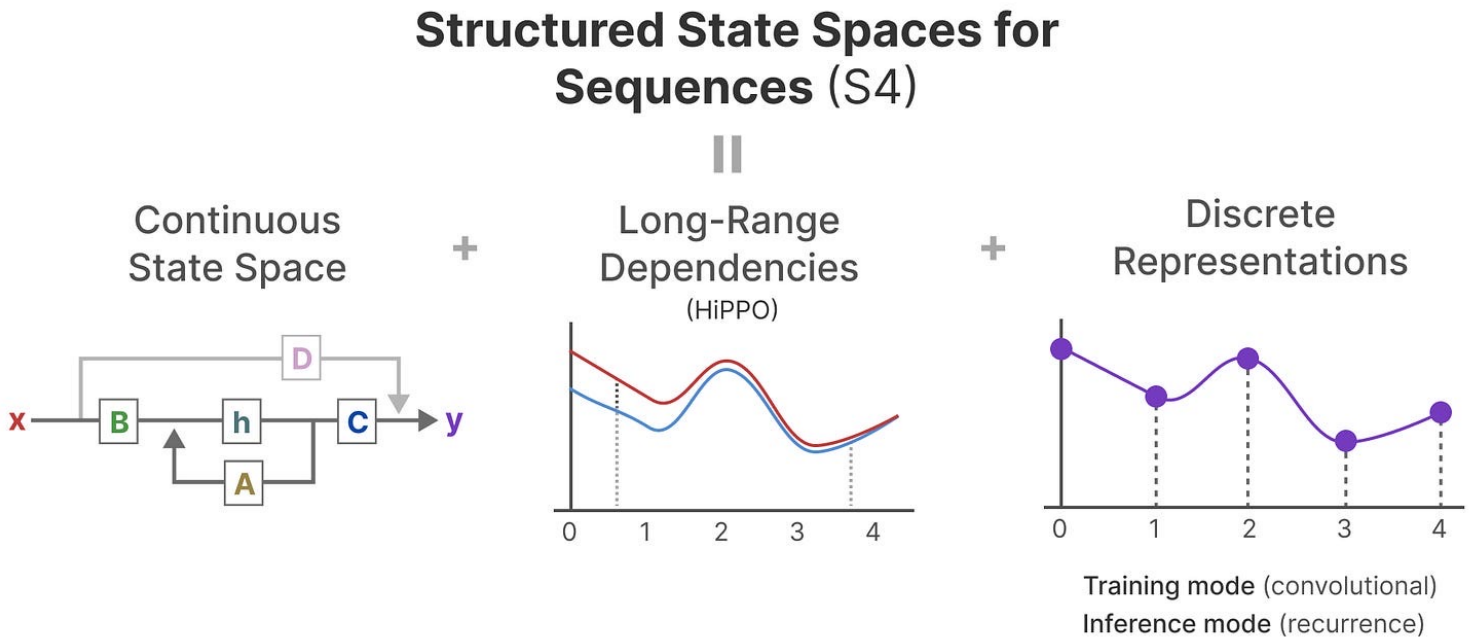
“For our purposes we mainly need to know that: 1) we only need to calculate it once, and 2) it has a nice, simple structure (which we will exploit in part 2). Without going into the ODE math, the main takeaway is that this matrix aims to compress the past history into a state that has enough information to approximately reconstruct the history.”

“Previous work found that simply modifying an SSM from a random matrix A to HiPPO improved its performance on the sequential MNIST classification benchmark from 60% to 98%.”

“Diving a bit deeper, the intuitive explanation of this matrix is that it produces a hidden state that memorizes its history. It does this by keeping track of the coefficients of a Legendre polynomial. These coefficients let it approximate all of the previous history.”

Voelker, Aaron R.; Kajić, Ivana; Eliasmith, Chris (2019). Legendre Memory Units: Continuous-Time Representation in Recurrent Neural Networks (PDF). Advances in Neural Information Processing Systems.

State Space Models (SSMs)



For more on this: https://srush.github.io/annotated-s4/

https://newsletter.maartengrootendorst.com/p/a-visual-guide-to-mamba-and-state

Reading material

- Introduction to SSM
  - https://huggingface.co/blog/lbourdois/get-on-the-ssm-train
- A History of SSM Models:
  - https://huggingface.co/blog/lbourdois/ssm-2022
- A Visual Guide to Mamba and SSMs:
  - https://newsletter.maartengrootendorst.com/p/a-visual-guide-to-mamba-and-state

MAMBA

Mamba: Linear-Time Sequence Modeling with Selective State Spaces

Albert Gu*[1] and Tri Dao*[2]

[1]Machine Learning Department, Carnegie Mellon University

[2]Department of Computer Science, Princeton University
agu@cs.cmu.edu, tri@tridao.me

Rejected at ICLR2024

Abstract

Foundation models, now powering most of the exciting applications in deep learning, are almost universally based on the Transformer architecture and its core attention module. Many subquadratic-time architectures such as linear attention, gated convolution and recurrent models, and structured state space models (SSMs) have been developed to address Transformers' computational inefficiency on long sequences, but they have not performed as well as attention on important modalities such as language. We identify that a key weakness of such models is their inability to perform content-based reasoning, and make several improvements. First, simply letting the SSM parameters be functions of the input addresses their weakness with discrete modalities, allowing the model to *selectively* propagate or forget information along the sequence length dimension depending on the current token. Second, even though this change prevents the use of efficient convolutions, we design a hardware-aware parallel algorithm in recurrent mode. We integrate these selective SSMs into a simplified end-to-end neural network architecture without attention or even MLP blocks (**Mamba**). Mamba enjoys fast inference (5× higher throughput than Transformers) and linear scaling in sequence length, and its performance improves on real data up to million-length sequences. As a general sequence model backbone, Mamba achieves state-of-the-art performance across several modalities such as language, audio, and genomics. On language modeling, our Mamba-3B model outperforms Transformers of the same size and matches Transformers twice its size, both in pretraining and downstream evaluation.

Motivation: Gap in the literature

Foundation models (FMs), or large models pretrained on massive data then adapted for downstream tasks, have emerged as an effective paradigm in modern machine learning. The backbone of these FMs are often *sequence models*, operating on arbitrary sequences of inputs from a wide variety of domains such as language, images, speech, audio, time series, and genomics (Brown et al. 2020; Dosovitskiy et al. 2020; Ismail Fawaz et al. 2019; Oord et al. 2016; Poli et al. 2023; Sutskever, Vinyals, and Quoc V Le 2014). While this concept is agnostic to a particular choice of model architecture, modern FMs are predominantly based on a single type of sequence model: the Transformer (Vaswani et al. 2017) and its core attention layer (Bahdanau, Cho, and Bengio 2015) The efficacy of self-attention is attributed to its ability to route information densely within a context window, allowing it to model complex data. However, this property brings fundamental drawbacks: an inability to model anything outside of a finite window, and quadratic scaling with respect to the window length. An enormous body of research has appeared on more efficient variants of attention to overcome these drawbacks (Tay, Dehghani, Bahri, et al. 2022), but often at the expense of the very properties that makes it effective. As of yet, none of these variants have been shown to be empirically effective at scale across domains.

Motivation: Gap in the literature

Recently, structured state space sequence models (SSMs) (Gu, Goel, and Ré 2022; Gu, Johnson, Goel, et al. 2021) have emerged as a promising class of architectures for sequence modeling. These models can be interpreted as a combination of recurrent neural networks (RNNs) and convolutional neural networks (CNNs), with inspiration from classical state space models (Kalman 1960). This class of models can be computed very efficiently as either a recurrence or convolution, with linear or near-linear scaling in sequence length. Additionally, they have principled mechanisms for modeling long-range dependencies (Gu, Dao, et al. 2020) in certain data modalities, and have dominated benchmarks such as the Long Range

Arena (Tay, Dehghani, Abnar, et al. 2021). Many flavors of SSMs (Gu, Goel, and Ré 2022; Gu, Gupta, et al. 2022; Gupta, Gu, and Berant 2022; Y. Li et al. 2023; Ma et al. 2023; Orvieto et al. 2023; Smith, Warrington, and Linderman 2023) have been successful in domains involving continuous signal data such as audio and vision (Goel et al. 2022; Nguyen, Goel, et al. 2022; Saon, Gupta, and Cui 2023). However, they have been less effective at modeling discrete and information-dense data such as text.

Contributions

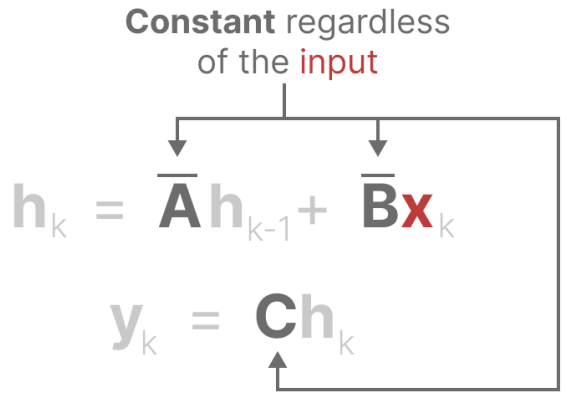
- "Selective Scan" Structured State Space Sequence (S6) Models

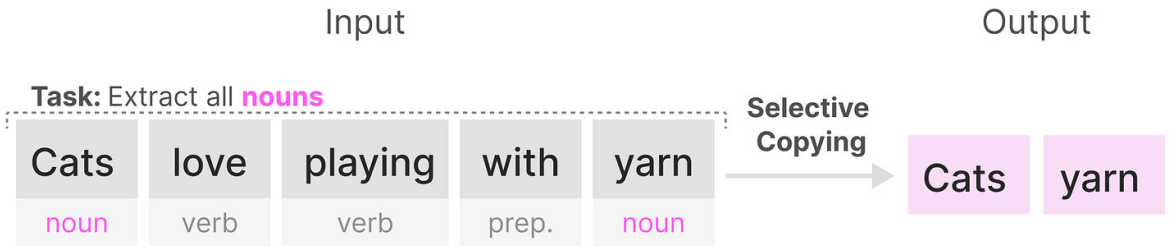
Selection Mechanism. First, we identify a key limitation of prior models: the ability to efficiently *select* data in an input-dependent manner (i.e. focus on or ignore particular inputs). Building on intuition based on important synthetic tasks such as selective copy and induction heads, we design a simple selection mechanism by parameterizing the SSM parameters based on the input. This allows the model to filter out irrelevant information and remember relevant information indefinitely.

Hardware-aware Algorithm. This simple change poses a technical challenge for the computation of the model; in fact, all prior SSMs models must be time- and input-invariant in order to be computationally efficient. We overcome this with a hardware-aware algorithm that computes the model recurrently with a scan instead of convolution, but does not materialize the expanded state in order to avoid IO access between different levels of the GPU memory hierarchy. The resulting implementation is faster than previous methods both in theory (scaling linearly in sequence length, compared to pseudo-linear for all convolution-based SSMs) and on modern hardware (up to 3× faster on A100 GPUs).

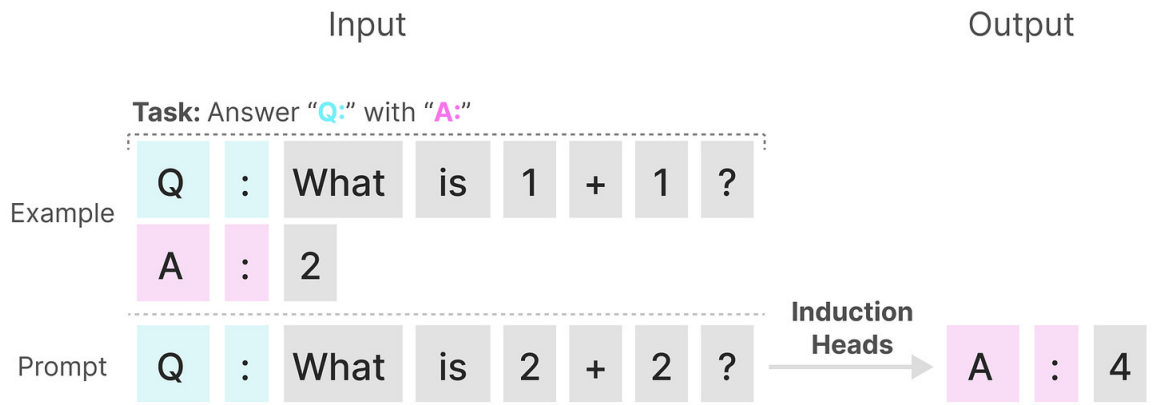
Architecture. We simplify prior deep sequence model architectures by combining the design of prior SSM architectures (Dao, Fu, Saab, et al. 2023) with the MLP block of Transformers into a single block, leading to a simple and homogenous architecture design (**Mamba**) incorporating selective state spaces.

Tasks that are challenging for S4





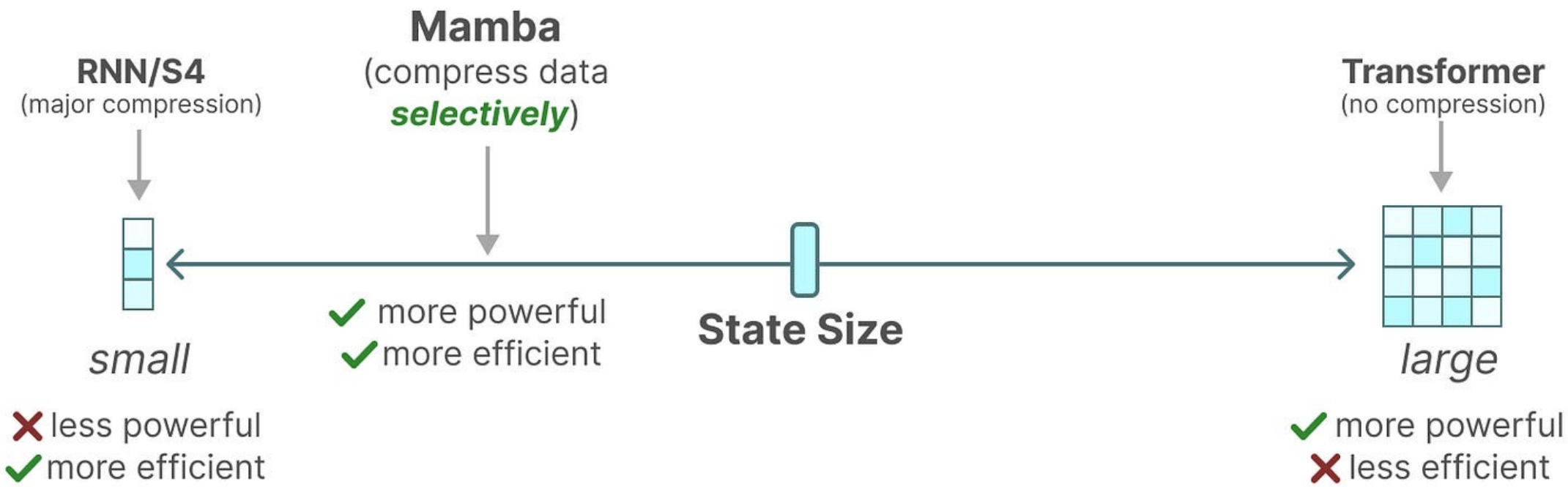
"However, a (recurrent/convolutional) SSM performs poorly in this task since it is Linear Time Invariant. As we saw before, the matrices A, B, and C are the same for every token the SSM generates."



"In the above example, we are essentially performing one-shot prompting where we attempt to "teach" the model to provide an "A:" response after every "Q:". However, since SSMs are time-invariant it cannot select which previous tokens to recall from its history."

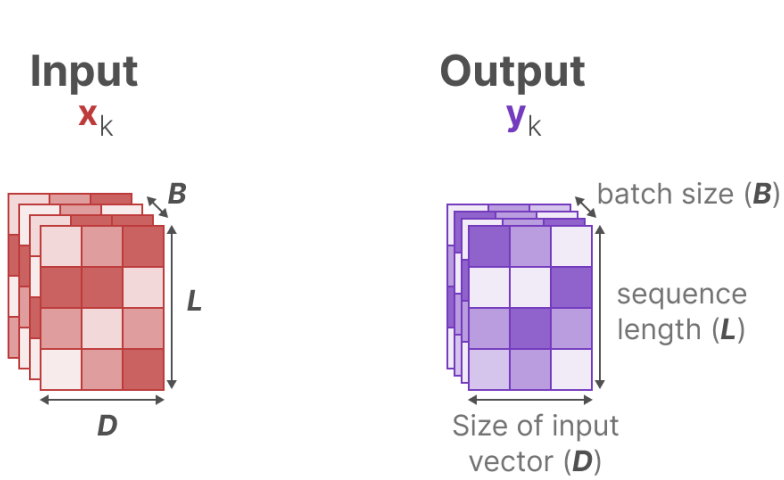
https://newsletter.maartengrootendorst.com/p/a-visual-guide-to-mamba-and-state

Mamba



https://newsletter.maartengrootendorst.com/p/a-visual-guide-to-mamba-and-state

Mamba



In Mamba, the matrices are different for each time step:

https://newsletter.maartengrootendorst.com/p/a-visual-guide-to-mamba-and-state

Mamba

Algorithm 1 SSM (S4)

Input: $x : (\mathsf{B}, \mathsf{L}, \mathsf{D})$
Output: $y : (\mathsf{B}, \mathsf{L}, \mathsf{D})$
1: $A : (\mathsf{D}, \mathsf{N}) \leftarrow$ Parameter
▷ Represents structured $N \times N$ matrix
2: $B : (\mathsf{D}, \mathsf{N}) \leftarrow$ Parameter
3: $C : (\mathsf{D}, \mathsf{N}) \leftarrow$ Parameter
4: $\Delta : (\mathsf{D}) \leftarrow \tau_\Delta(\text{Parameter})$
5: $\overline{A}, \overline{B} : (\mathsf{D}, \mathsf{N}) \leftarrow \text{discretize}(\Delta, A, B)$
6: $y \leftarrow \text{SSM}(\overline{A}, \overline{B}, C)(x)$
▷ Time-invariant: recurrence or convolution
7: **return** y

Algorithm 2 SSM + Selection (S6)

Input: $x : (\mathsf{B}, \mathsf{L}, \mathsf{D})$
Output: $y : (\mathsf{B}, \mathsf{L}, \mathsf{D})$
1: $A : (\mathsf{D}, \mathsf{N}) \leftarrow$ Parameter
▷ Represents structured $N \times N$ matrix
2: $B : (\mathsf{B}, \mathsf{L}, \mathsf{N}) \leftarrow s_B(x)$
3: $C : (\mathsf{B}, \mathsf{L}, \mathsf{N}) \leftarrow s_C(x)$
4: $\Delta : (\mathsf{B}, \mathsf{L}, \mathsf{D}) \leftarrow \tau_\Delta(\text{Parameter} + s_\Delta(x))$
5: $\overline{A}, \overline{B} : (\mathsf{B}, \mathsf{L}, \mathsf{D}, \mathsf{N}) \leftarrow \text{discretize}(\Delta, A, B)$
6: $y \leftarrow \text{SSM}(\overline{A}, \overline{B}, C)(x)$
▷ Time-varying: recurrence (*scan*) only
7: **return** y

We specifically choose $s_B(x) = \text{Linear}_N(x)$, $s_C(x) = \text{Linear}_N(x)$, $s_\Delta(x) = \text{Broadcast}_D(\text{Linear}_1(x))$, and $\tau_\Delta = \text{softplus}$, where Linear_d is a parameterized projection to dimension d. The choice of s_Δ and τ_Δ is due to a connection to RNN gating mechanisms explained in Section 3.5.

Mamba

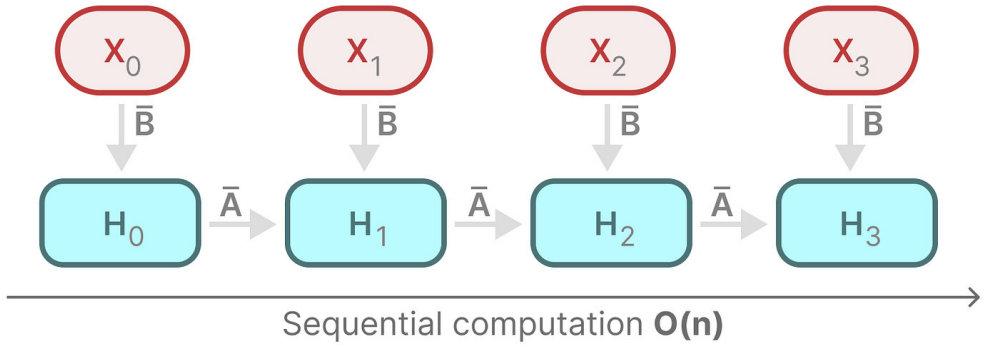
Concretely, instead of preparing the scan input $(\overline{\boldsymbol{A}}, \overline{\boldsymbol{B}})$ of size $(\mathtt{B}, \mathtt{L}, \mathtt{D}, \mathtt{N})$ in GPU HBM (high-bandwidth memory), we load the SSM parameters $(\Delta, \boldsymbol{A}, \boldsymbol{B}, C)$ directly from slow HBM to fast SRAM, perform the discretization and recurrence in SRAM, and then write the final outputs of size $(\mathtt{B}, \mathtt{L}, \mathtt{D})$ back to HBM.

To avoid the sequential recurrence, we observe that despite not being linear it can still be parallelized with a work-efficient parallel scan algorithm (Blelloch 1990; Martin and Cundy 2018; Smith, Warrington, and Linderman 2023).

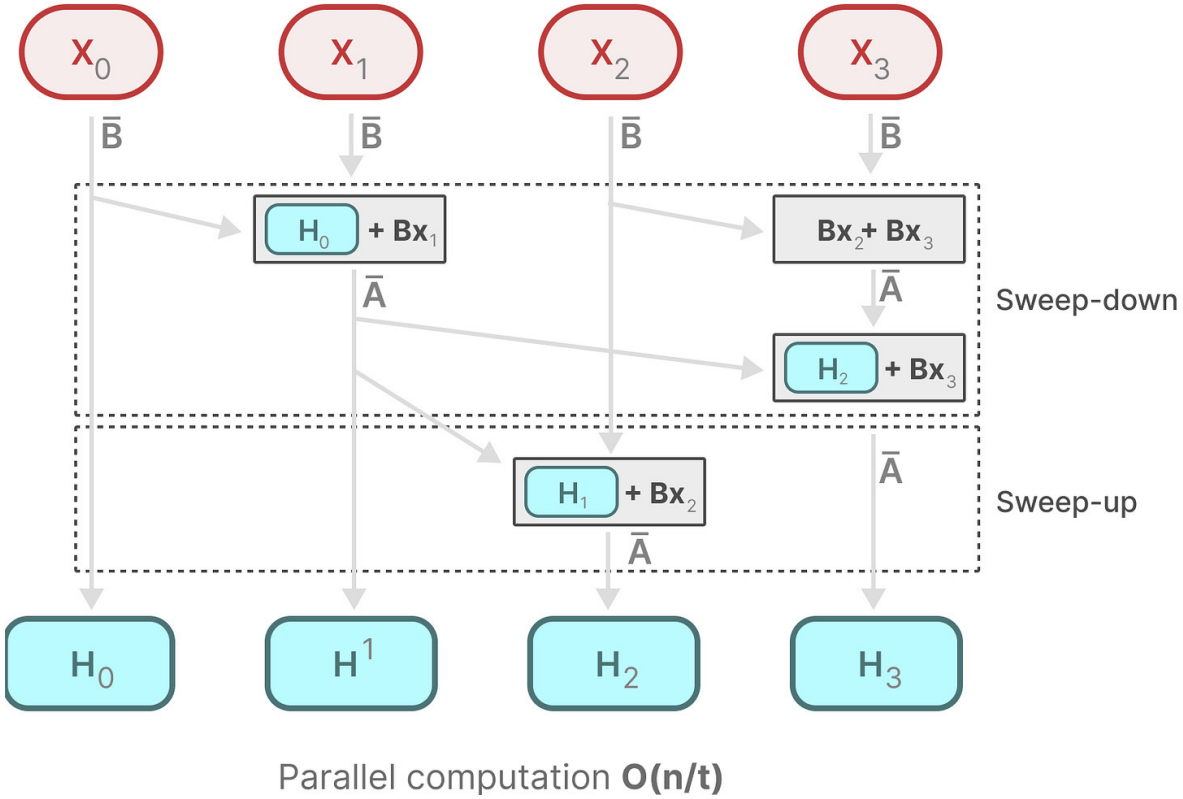
Finally, we must also avoid saving the intermediate states, which are necessary for backpropagation. We carefully apply the classic technique of recomputation to reduce the memory requirements: the intermediate states are not stored but recomputed in the backward pass when the inputs are loaded from HBM to SRAM. As a result, the fused selective scan layer has the same memory requirements as an optimized transformer implementation with FlashAttention.

Mamba

Sequential scan –
Not suitable for parallelization



Parallel scan



"Together, dynamic matrices B and C, and the parallel scan algorithm create the selective scan algorithm to represent the dynamic and fast nature of using the recurrent representation."

https://newsletter.maartengrootendorst.com/p/a-visual-guide-to-mamba-and-state

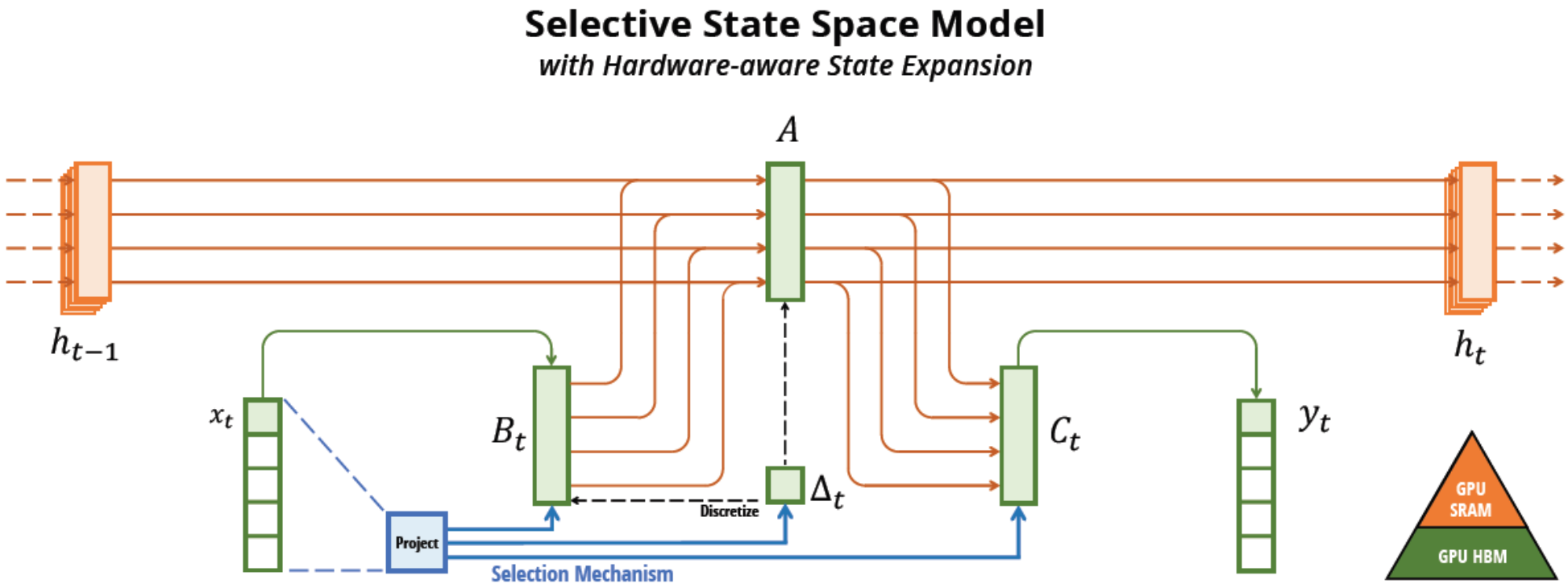


Figure 1: (**Overview**.) Structured SSMs independently map each channel (e.g. $D = 5$) of an input x to output y through a higher dimensional latent state h (e.g. $N = 4$). Prior SSMs avoid materializing this large effective state (DN, times batch size B and sequence length L) through clever alternate computation paths requiring time-invariance: the $(\Delta, \boldsymbol{A}, B, C)$ parameters are constant across time. Our selection mechanism adds back input-dependent dynamics, which also requires a careful hardware-aware algorithm to only materialize the expanded states in more efficient levels of the GPU memory hierarchy.

Mamba block

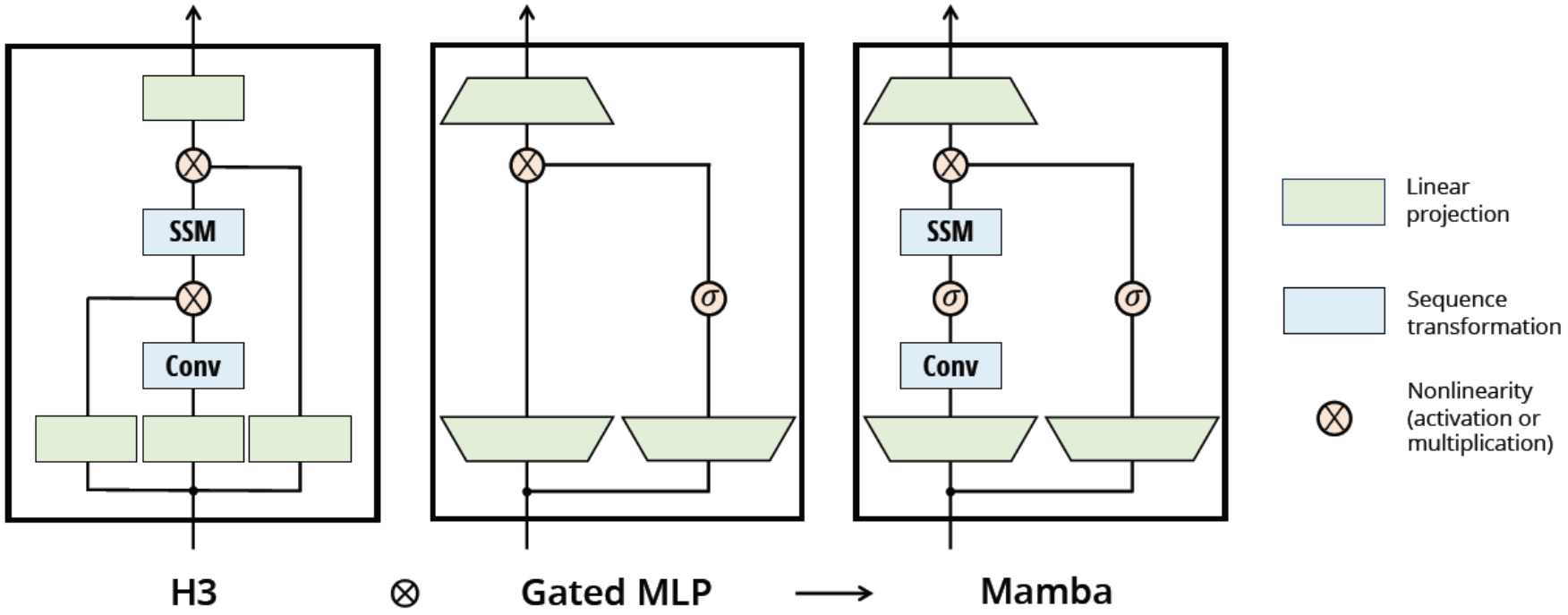
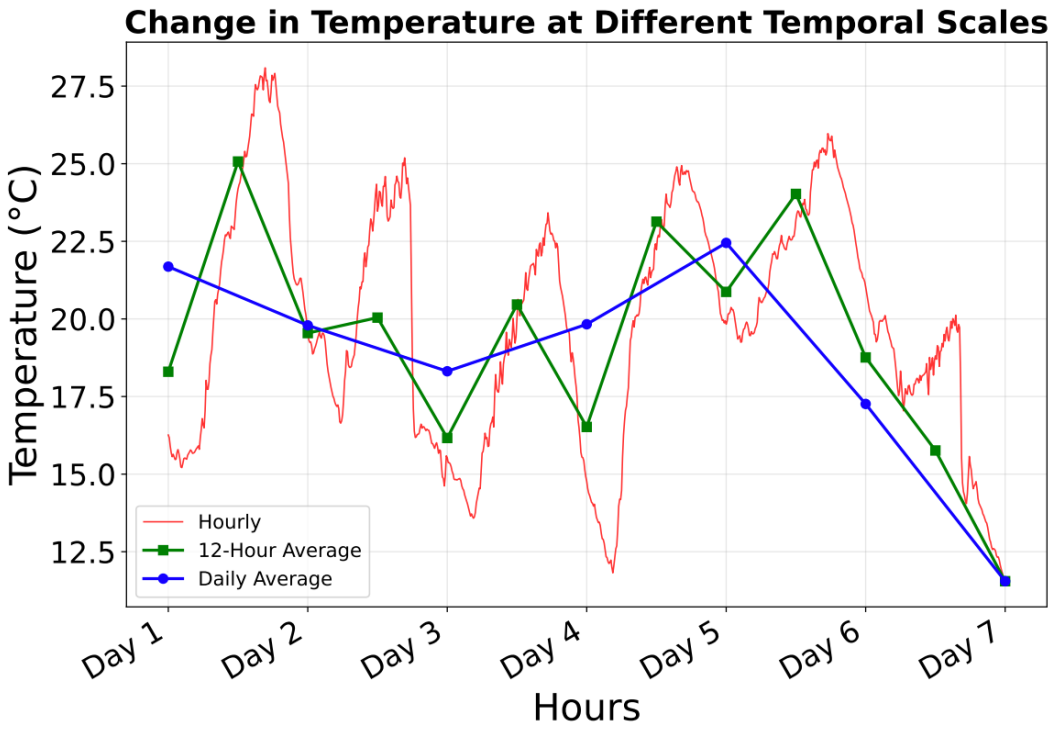


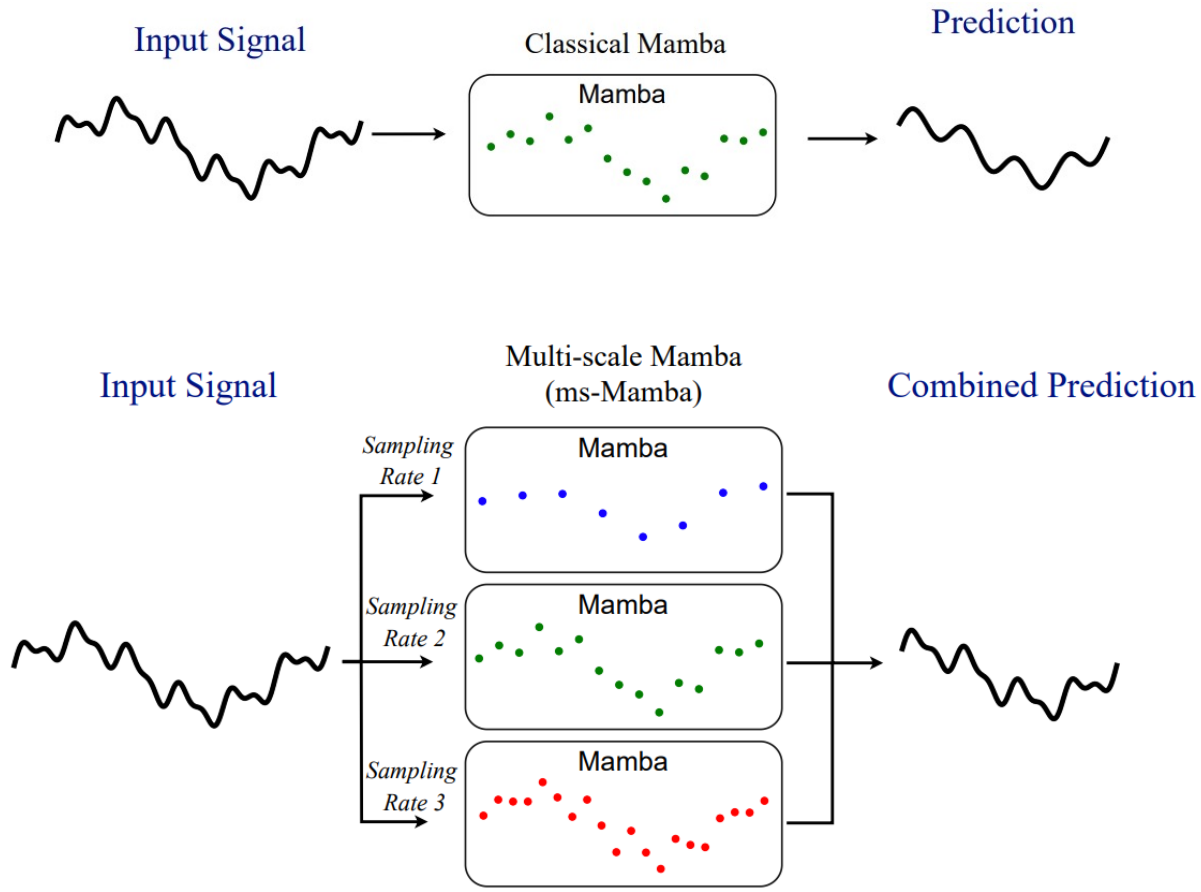
Figure 3: (**Architecture**.) Our simplified block design combines the H3 block, which is the basis of most SSM architectures, with the ubiquitous MLP block of modern neural networks. Instead of interleaving these two blocks, we simply repeat the Mamba block homogenously. Compared to the H3 block, Mamba replaces the first multiplicative gate with an activation function. Compared to the MLP block, Mamba adds an SSM to the main branch. For σ we use the SiLU / Swish activation (Hendrycks and Gimpel 2016; Ramachandran, Zoph, and Quoc V Le 2017).

	Training	Inference
Transformers	**Fast!** (parallelizable)	**Slow...** (scales **quadratically** with sequence length)
RNNs	**Slow...** (not parallelizable)	**Fast!** (scales **linearly** with sequence length)
Mamba	**Fast!** (parallelizable)	**Fast!** (scales **linearly** with sequence length + **unbounded** context)

Multi-scale Mamba

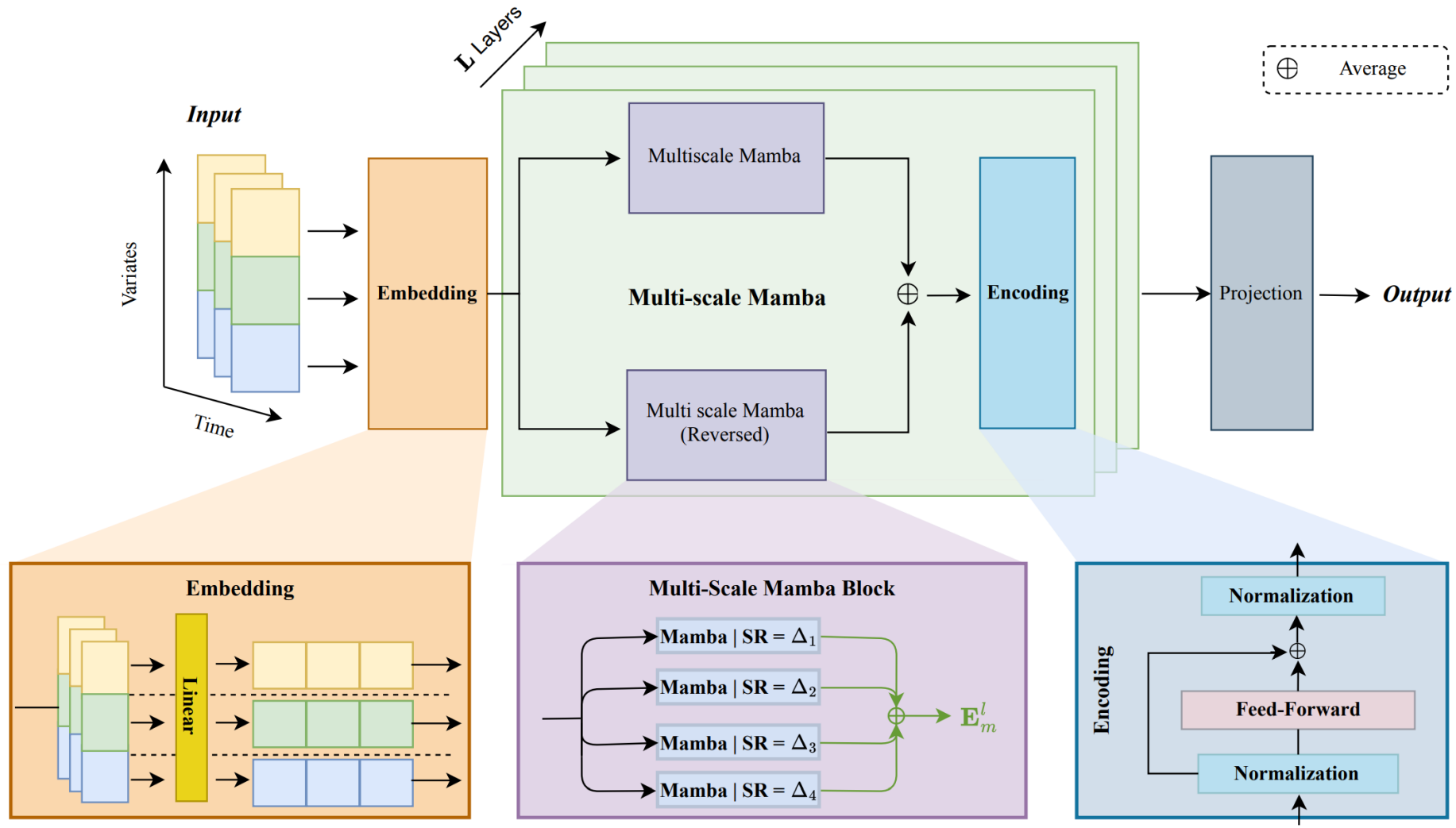
Y. M. Karadag, I. G. Dino, S. Kalkan, "ms-Mamba: Multi-scale Mamba for Time-Series Forecasting", 680:133226, Neurocomputing, 2026.





Multi-scale Mamba

Y. M. Karadag, I. G. Dino, S. Kalkan, "ms-Mamba: Multi-scale Mamba for Time-Series Forecasting", 680:133226, Neurocomputing, 2026.



Now

- Pretraining Transformers for language tasks
- Large Language Models

Pre-training in NLP

- Word embeddings are the basis of deep learning for NLP

king → [-0.5, -0.9, 1.4, …]

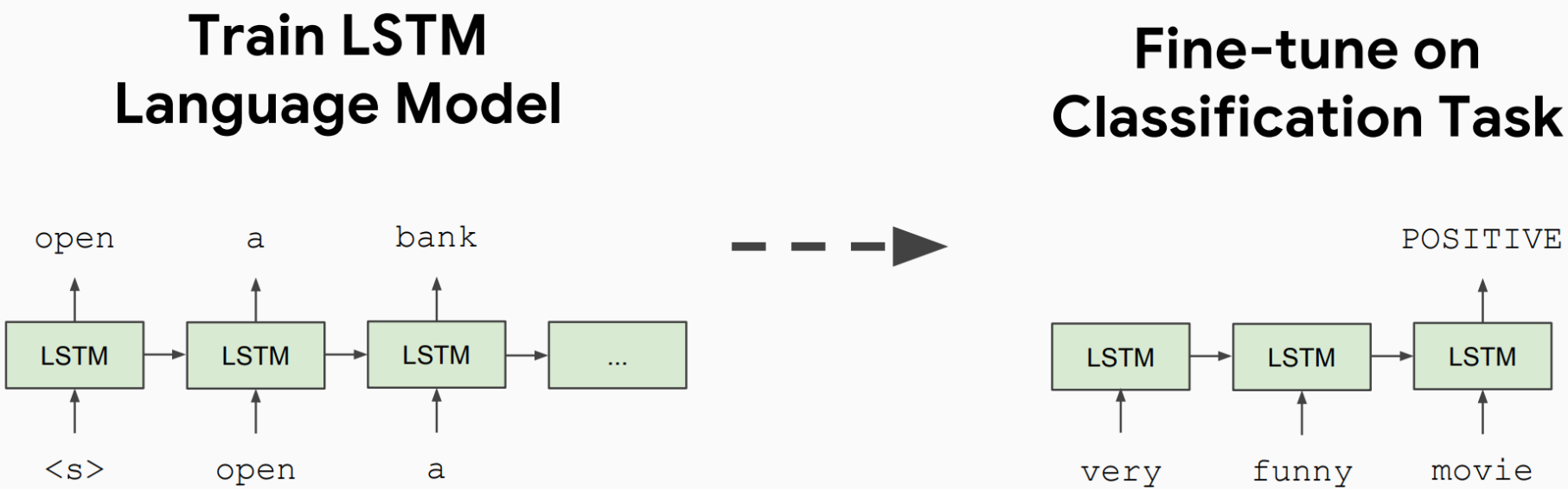
queen → [-0.6, -0.8, -0.2, …]

- Word embeddings (`word2vec`, `GloVe`) are often *pre-trained* on text corpus from co-occurrence statistics

Slide: Jacob Devlin
https://nlp.stanford.edu/seminar/details/jdevlin.pdf

Pre-training in NLP

- *Semi-Supervised Sequence Learning*, Google, 2015

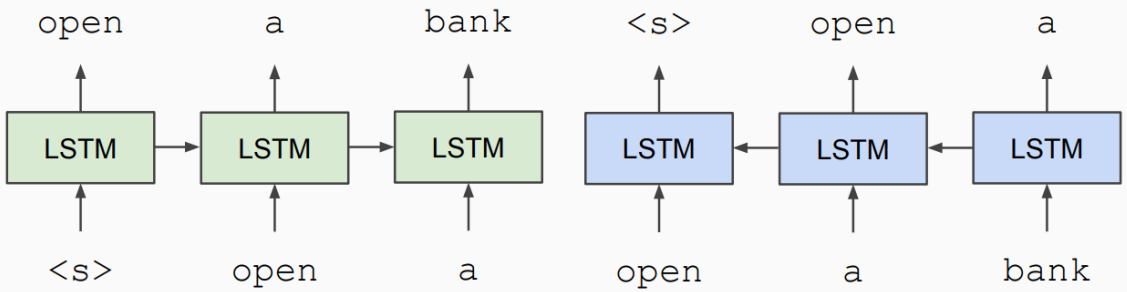


Slide: Jacob Devlin
https://nlp.stanford.edu/seminar/details/jdevlin.pdf

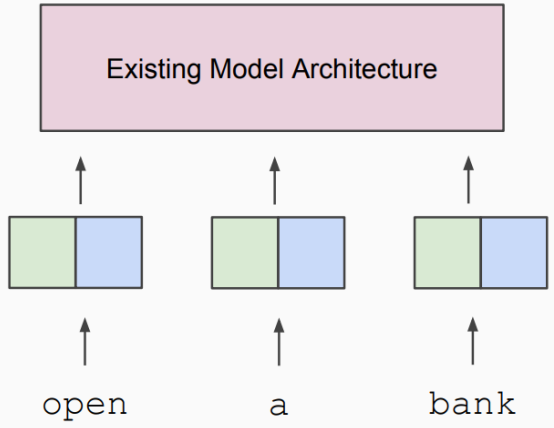
Pre-training in NLP

- *ELMo: Deep Contextual Word Embeddings*, AI2 & University of Washington, 2017

Train Separate Left-to-Right and Right-to-Left LMs



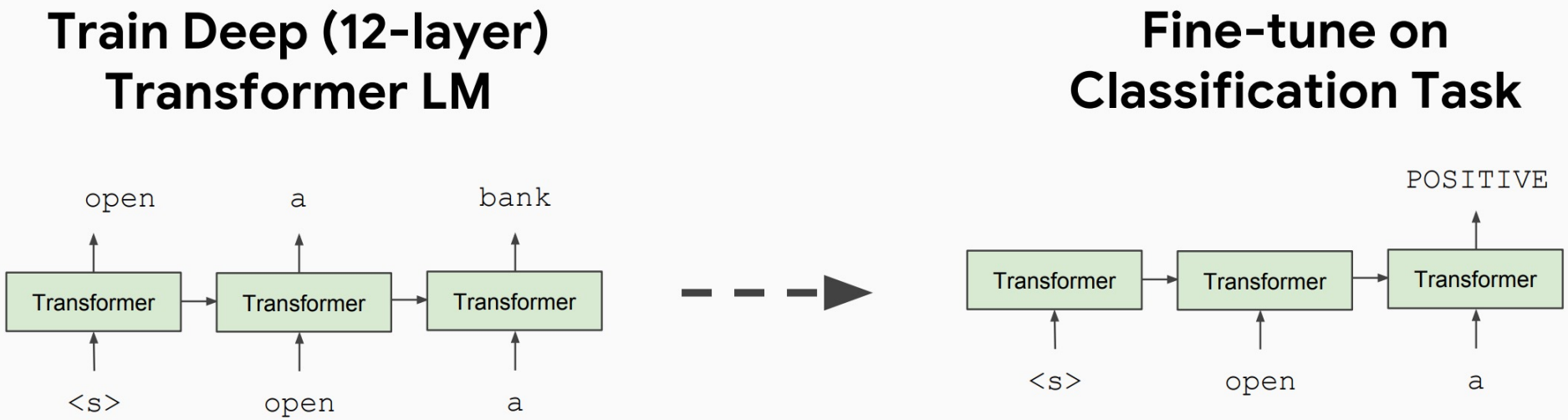
Apply as "Pre-trained Embeddings"



Slide: Jacob Devlin
https://nlp.stanford.edu/seminar/details/jdevlin.pdf

Pre-training in NLP

- *Improving Language Understanding by Generative Pre-Training*, OpenAI, 2018



Slide: Jacob Devlin
https://nlp.stanford.edu/seminar/details/jdevlin.pdf

GPT-1

GPT-1

- 12 layer decoder-only transformer
- Unsupervised pretraining
  - BookCorpus dataset
- Supervised finetuning
  - Textual alignment
  - QA & commonsense reasoning
  - Semantic similarity
  - Classification

OpenAI Technical Report, 2018

Improving Language Understanding by Generative Pre-Training

Alec Radford, OpenAI, alec@openai.com
Karthik Narasimhan, OpenAI, karthikn@openai.com
Tim Salimans, OpenAI, tim@openai.com
Ilya Sutskever, OpenAI, ilyasu@openai.com

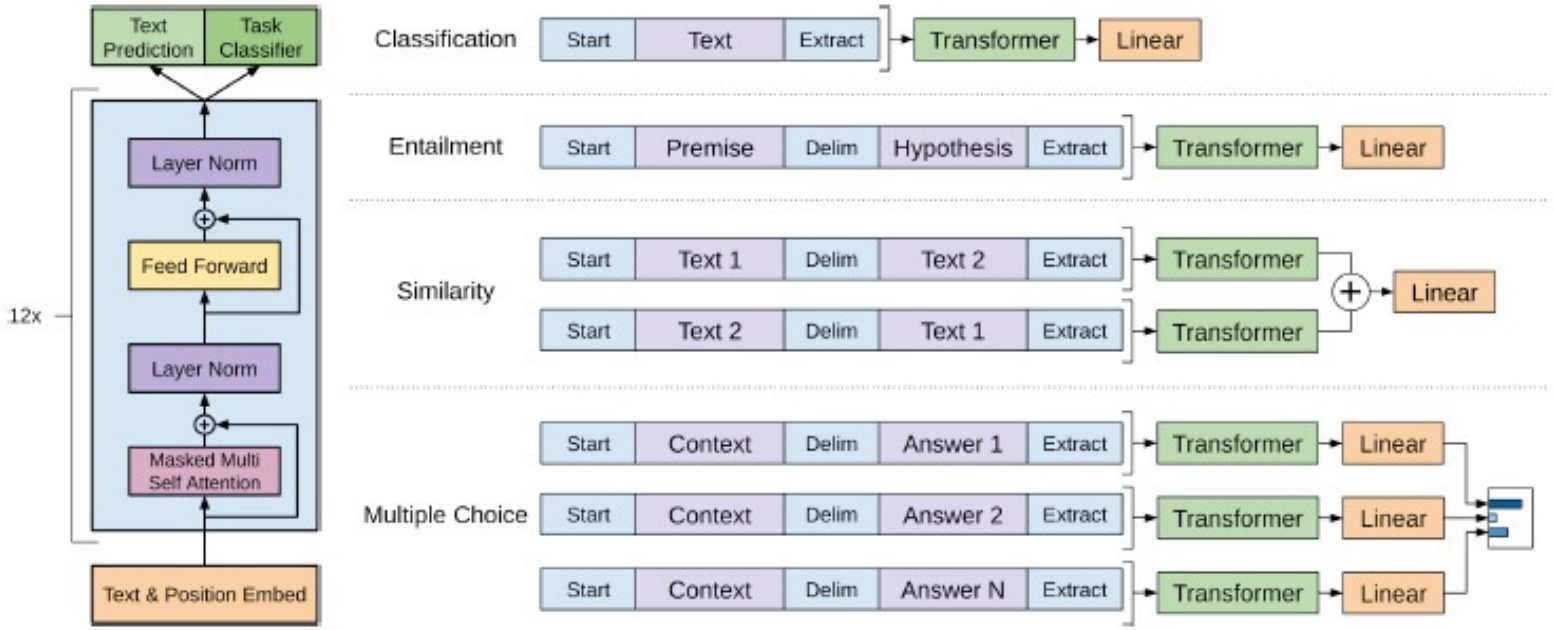


Figure 1: **(left)** Transformer architecture and training objectives used in this work. **(right)** Input transformations for fine-tuning on different tasks. We convert all structured inputs into token sequences to be processed by our pre-trained model, followed by a linear+softmax layer.

GPT-1

Given an unsupervised corpus of tokens $\mathcal{U} = \{u_1, \ldots, u_n\}$, we use a standard language modeling objective to maximize the following likelihood:

$$L_1(\mathcal{U}) = \sum_i \log P(u_i | u_{i-k}, \ldots, u_{i-1}; \Theta) \tag{1}$$

where k is the size of the context window, and the conditional probability P is modeled using a neural network with parameters Θ. These parameters are trained using stochastic gradient descent [51].

$$\begin{aligned} h_0 &= UW_e + W_p \\ h_l &= \texttt{transformer\_block}(h_{l-1}) \forall i \in [1, n] \\ P(u) &= \texttt{softmax}(h_n W_e^T) \end{aligned} \tag{2}$$

where $U = (u_{-k}, \ldots, u_{-1})$ is the context vector of tokens, n is the number of layers, W_e is the token embedding matrix, and W_p is the position embedding matrix.

GPT-1

Discriminative Fine-tuning

For labeled downstream task, maximize the log probability on each pair of instance (x, y)

After training the model with the objective in Eq. 1, we adapt the parameters to the supervised target task. We assume a labeled dataset $\mathcal{C}$, where each instance consists of a sequence of input tokens, $x^1, \ldots, x^m$, along with a label y. The inputs are passed through our pre-trained model to obtain the final transformer block's activation h_l^m, which is then fed into an added linear output layer with parameters W_y to predict y:

$$P(y|x^1, \ldots, x^m) = \texttt{softmax}(h_l^m W_y). \tag{3}$$

This gives us the following objective to maximize:

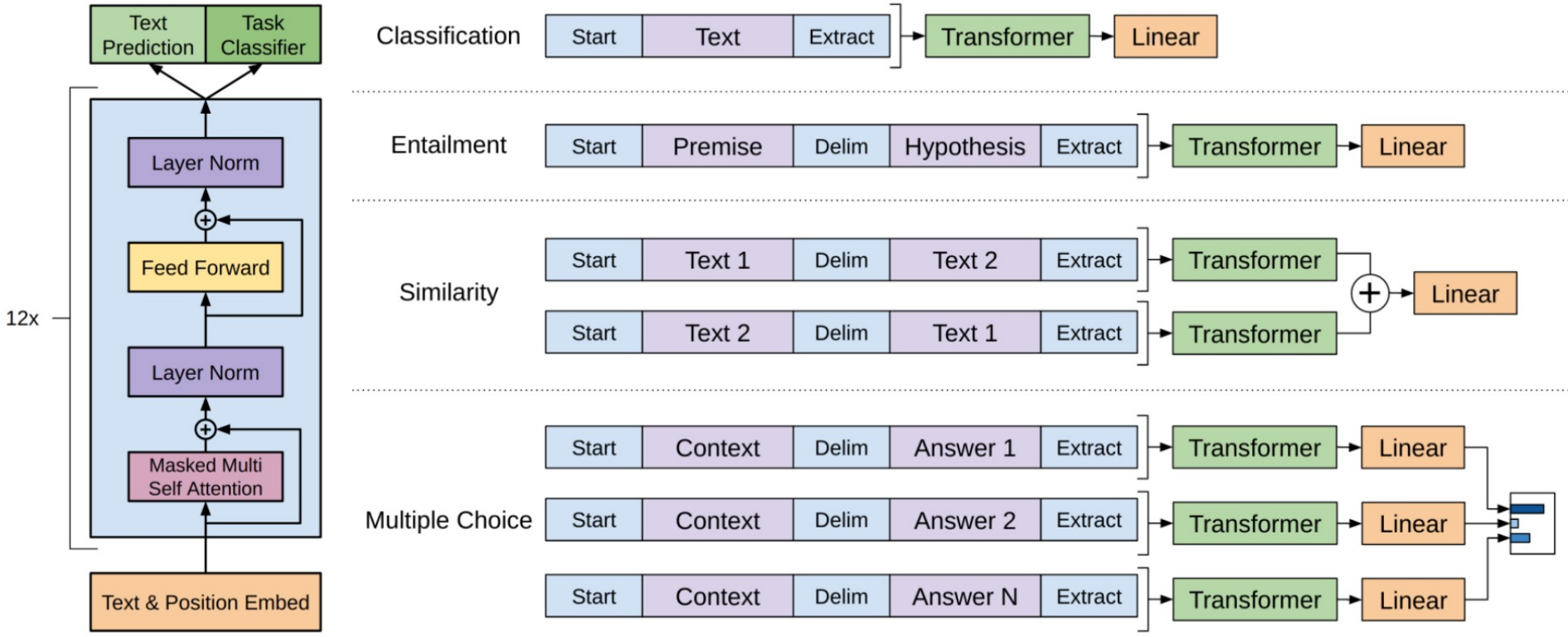
$$L_2(\mathcal{C}) = \sum_{(x,y)} \log P(y|x^1, \ldots, x^m). \tag{4}$$

Add auxiliary fine-tuning objective of language modeling will imporove the performance $L_3(C) = L_2(C) + \lambda * L_1(C)$

Slide: Weizhi Wang
https://victorwz.github.io/additional_files/slides_gpt_cs291A.pdf

GPT-1

Discriminative Fine-tuning



Slide: Weizhi Wang
https://victorwz.github.io/additional_files/slides_gpt_cs291A.pdf

GPT-1 Results

Table 2: Experimental results on natural language inference tasks, comparing our model with current state-of-the-art methods. 5x indicates an ensemble of 5 models. All datasets use accuracy as the evaluation metric.

Method	MNLI-m	MNLI-mm	SNLI	SciTail	QNLI	RTE
ESIM + ELMo [44] (5x)	-	-	89.3	-	-	-
CAFE [58] (5x)	80.2	79.0	89.3	-	-	-
Stochastic Answer Network [35] (3x)	80.6	80.1	-	-	-	-
CAFE [58]	78.7	77.9	88.5	83.3		
GenSen [64]	71.4	71.3	-	-	82.3	59.2
Multi-task BiLSTM + Attn [64]	72.2	72.1	-	-	82.1	**61.7**
Finetuned Transformer LM (ours)	**82.1**	**81.4**	**89.9**	**88.3**	**88.1**	56.0

BERT

Bidirectional Encoder Representations from Transformers (BERT)

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

2018

Jacob Devlin **Ming-Wei Chang** **Kenton Lee** **Kristina Toutanova**
Google AI Language
{jacobdevlin,mingweichang,kentonl,kristout}@google.com

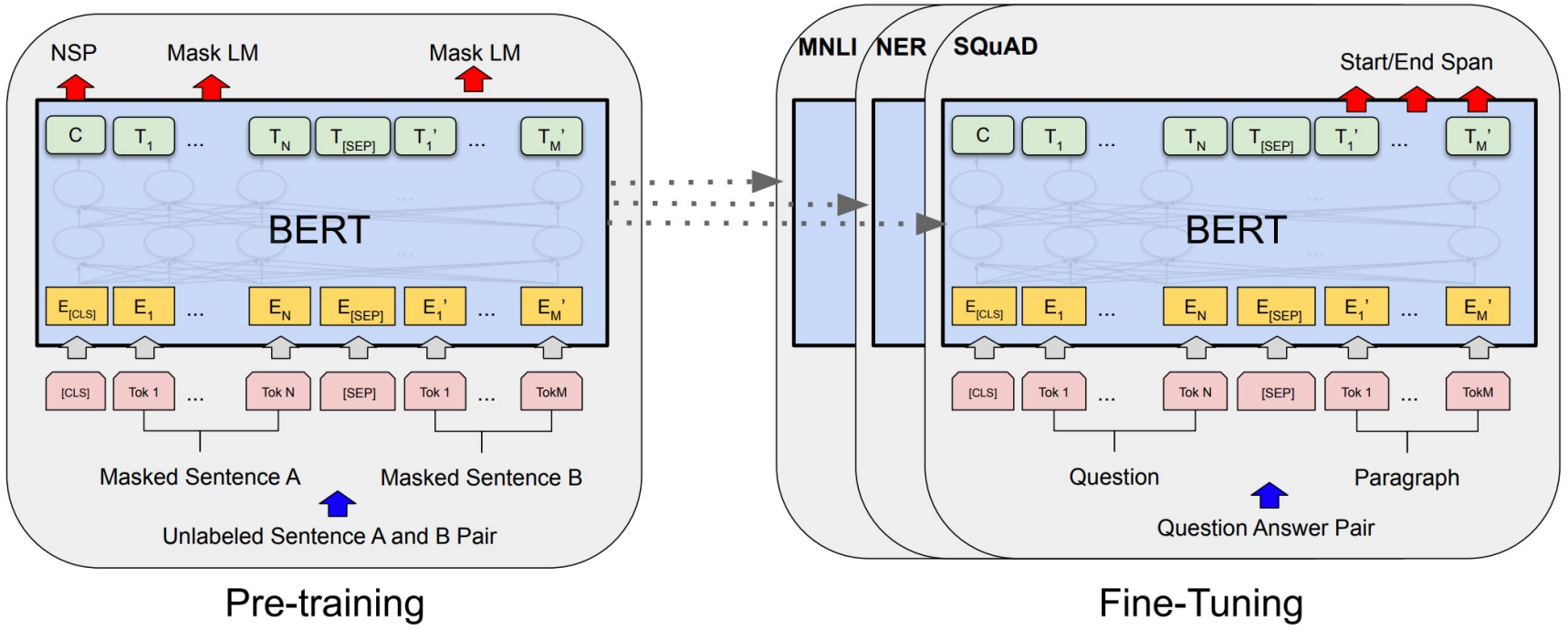


Figure 1: Overall pre-training and fine-tuning procedures for BERT. Apart from output layers, the same architectures are used in both pre-training and fine-tuning. The same pre-trained model parameters are used to initialize models for different down-stream tasks. During fine-tuning, all parameters are fine-tuned. `[CLS]` is a special symbol added in front of every input example, and `[SEP]` is a special separator token (e.g. separating questions/answers).

BERT: NLP Tasks

- MNLI: Multi-Genre Natural Language Inference

Examples

Premise	Label	Hypothesis
Fiction		
The Old One always comforted Ca'daan, except today.	*neutral*	Ca'daan knew the Old One very well.
Letters		
Your gift is appreciated by each and every student who will benefit from your generosity.	*neutral*	Hundreds of students will benefit from your generosity.
Telephone Speech		
yes now you know if if everybody like in August when everybody's on vacation or something we can dress a little more casual or	*contradiction*	August is a black out month for vacations in the company.
9/11 Report		
At the other end of Pennsylvania Avenue, people began to line up for a White House tour.	*entailment*	People formed a line at the end of Pennsylvania Avenue.

- NER: Named Entity Recognition
- SQuaD: Stanford Question Answering Dataset

BERT: Motivation

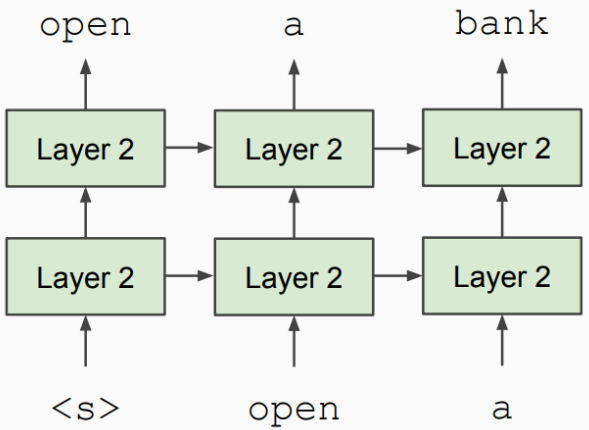
- **Problem**: Language models only use left context *or* right context, but language understanding is bidirectional.
- Why are LMs unidirectional?
- Reason 1: Directionality is needed to generate a well-formed probability distribution.
  - We don't care about this.
- Reason 2: Words can "see themselves" in a bidirectional encoder.

Slide: Jacob Devlin
https://nlp.stanford.edu/seminar/details/jdevlin.pdf

BERT: Motivation

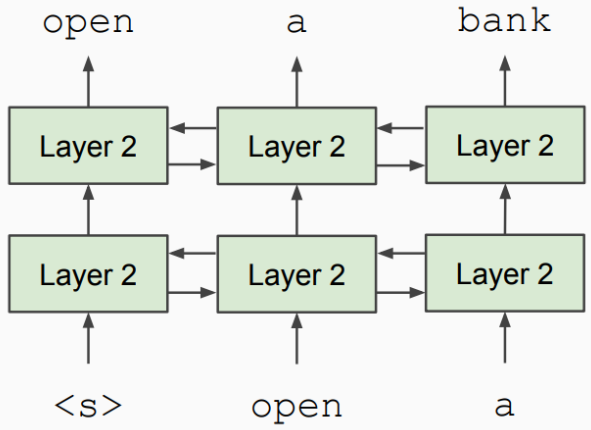
Unidirectional context

Build representation incrementally



Bidirectional context

Words can “see themselves”

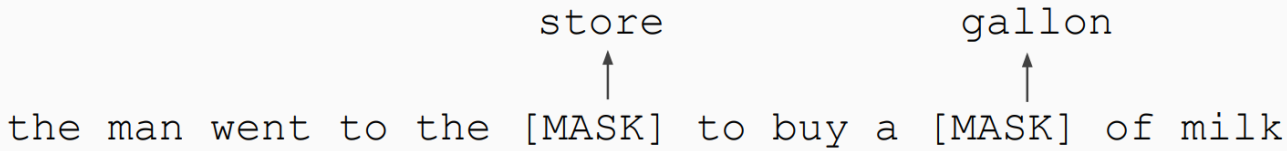


Actually, this is vanilla attention that is not causal!

Slide: Jacob Devlin
https://nlp.stanford.edu/seminar/details/jdevlin.pdf

BERT: Motivation

- **Solution**: Mask out *k*% of the input words, and then predict the masked words
  - We always use *k* = 15%



- Too little masking: Too expensive to train
- Too much masking: Not enough context

Slide: Jacob Devlin
https://nlp.stanford.edu/seminar/details/jdevlin.pdf

BERT: Motivation

- Problem: Mask token never seen at fine-tuning
- Solution: 15% of the words to predict, but don't replace with `[MASK]` 100% of the time. Instead:
- 80% of the time, replace with `[MASK]`

  `went to the store → went to the [MASK]`
- 10% of the time, replace random word

  `went to the store → went to the running`
- 10% of the time, keep same

  `went to the store → went to the store`

Slide: Jacob Devlin
https://nlp.stanford.edu/seminar/details/jdevlin.pdf

BERT: Motivation

- To learn *relationships* between sentences, predict whether Sentence B is actual sentence that proceeds Sentence A, or a random sentence

Sentence A = `The man went to the store.`
Sentence B = `He bought a gallon of milk.`
Label = `IsNextSentence`

Sentence A = `The man went to the store.`
Sentence B = `Penguins are flightless.`
Label = `NotNextSentence`

Slide: Jacob Devlin
https://nlp.stanford.edu/seminar/details/jdevlin.pdf

BERT

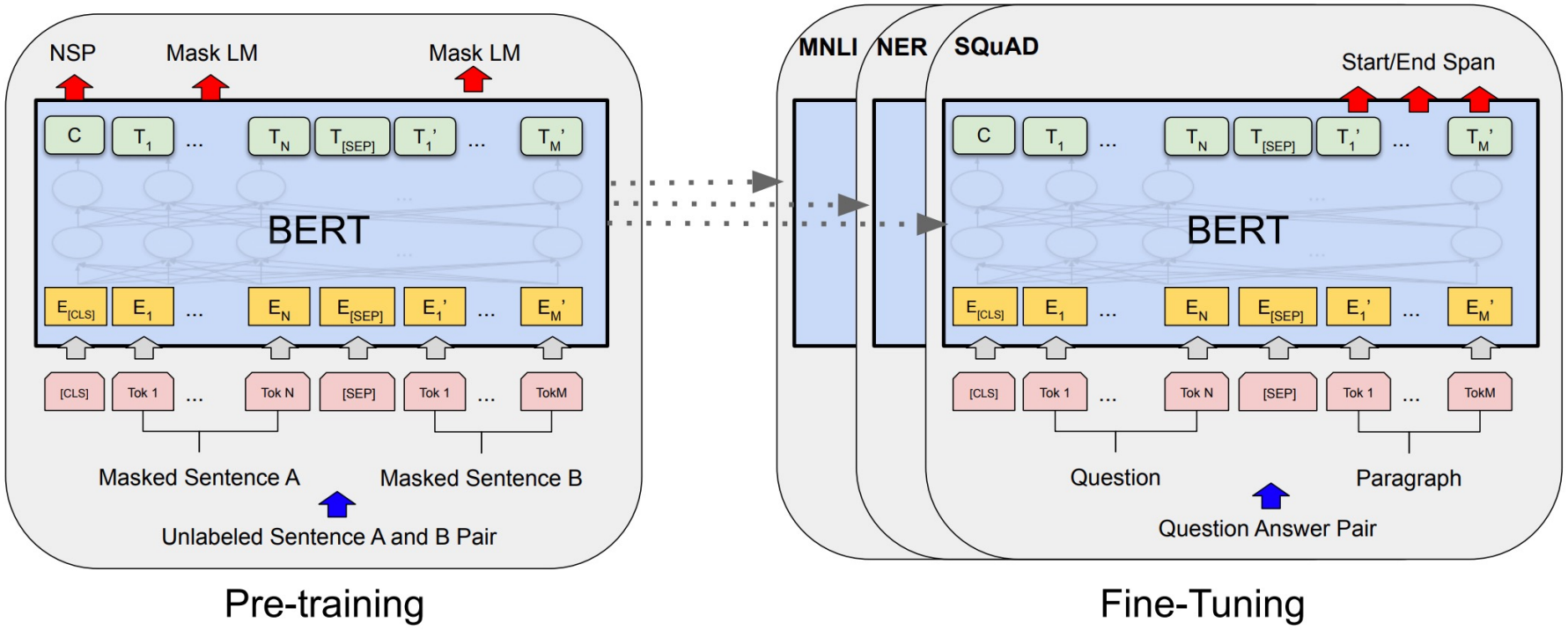
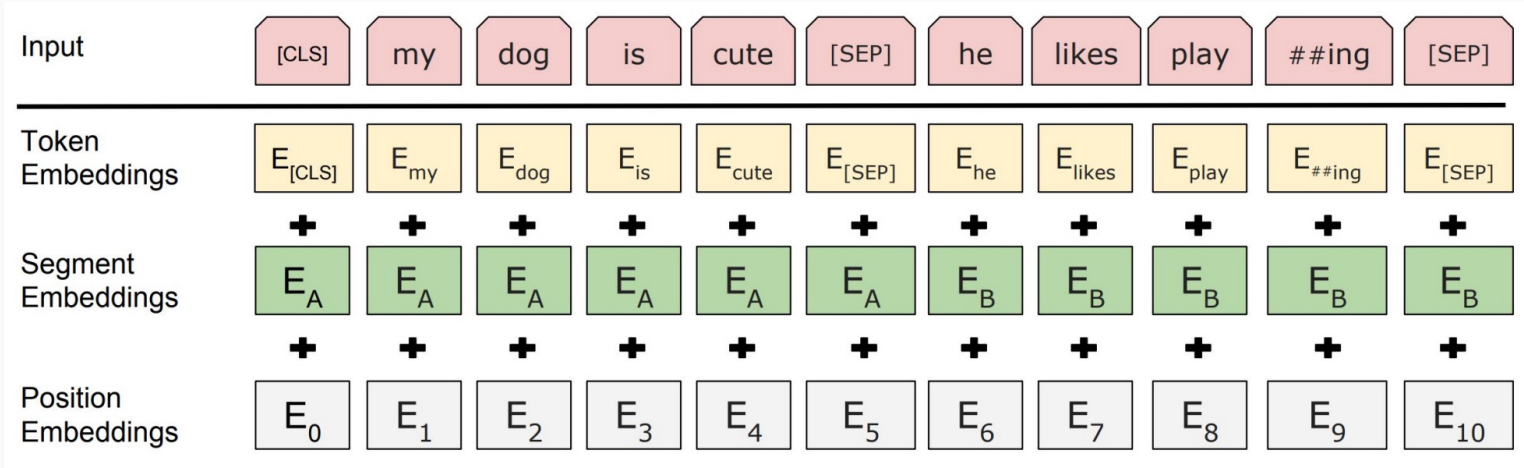


Figure 1: Overall pre-training and fine-tuning procedures for BERT. Apart from output layers, the same architectures are used in both pre-training and fine-tuning. The same pre-trained model parameters are used to initialize models for different down-stream tasks. During fine-tuning, all parameters are fine-tuned. `[CLS]` is a special symbol added in front of every input example, and `[SEP]` is a special separator token (e.g. separating questions/answers).

BERT



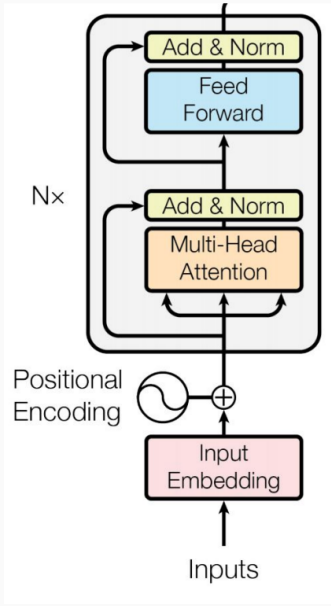
- Use 30,000 WordPiece vocabulary on input.
- Each token is sum of three embeddings
- Single sequence is much more efficient.

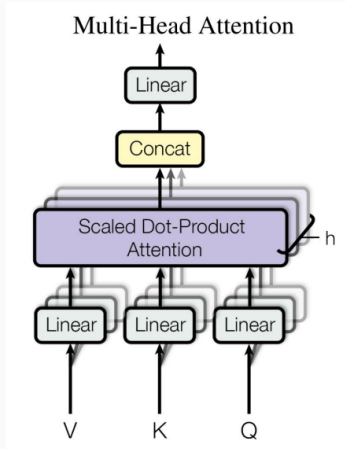
BERT

Model Architecture

Transformer encoder

- Multi-headed self attention
  - Models context
- Feed-forward layers
  - Computes non-linear hierarchical features
- Layer norm and residuals
  - Makes training deep networks healthy
- Positional embeddings
  - Allows model to learn relative positioning





BERT

The General Language Understanding Evaluation (GLUE) benchmark is a collection of resources for training, evaluating, and analyzing natural language understanding systems. GLUE consists of:

- A benchmark of nine sentence- or sentence-pair language understanding tasks built on established existing datasets and selected to cover a diverse range of dataset sizes, text genres, and degrees of difficulty,
- A diagnostic dataset designed to evaluate and analyze model performance with respect to a wide range of linguistic phenomena found in natural language, and
- A public leaderboard for tracking performance on the benchmark and a dashboard for visualizing the performance of models on the diagnostic set.

https://gluebenchmark.com/

GLUE Results

System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	**Average** -
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.9	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	88.1	91.3	45.4	80.0	82.3	56.0	75.2
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.1	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**91.1**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**81.9**

MultiNLI

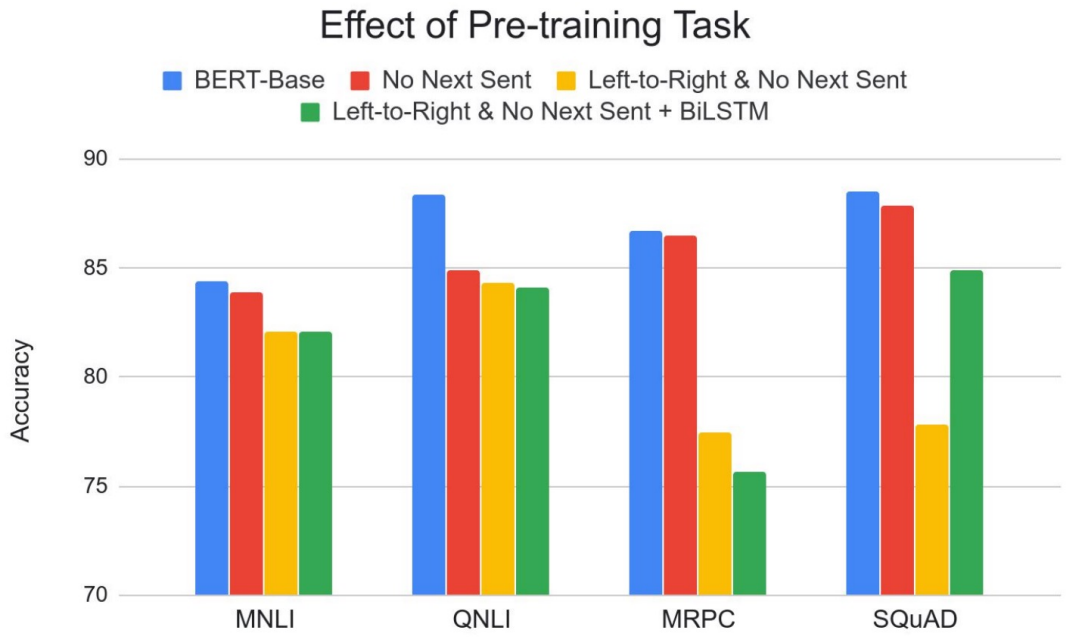
Premise: Hills and mountains are especially sanctified in Jainism.
Hypothesis: Jainism hates nature.
Label: Contradiction

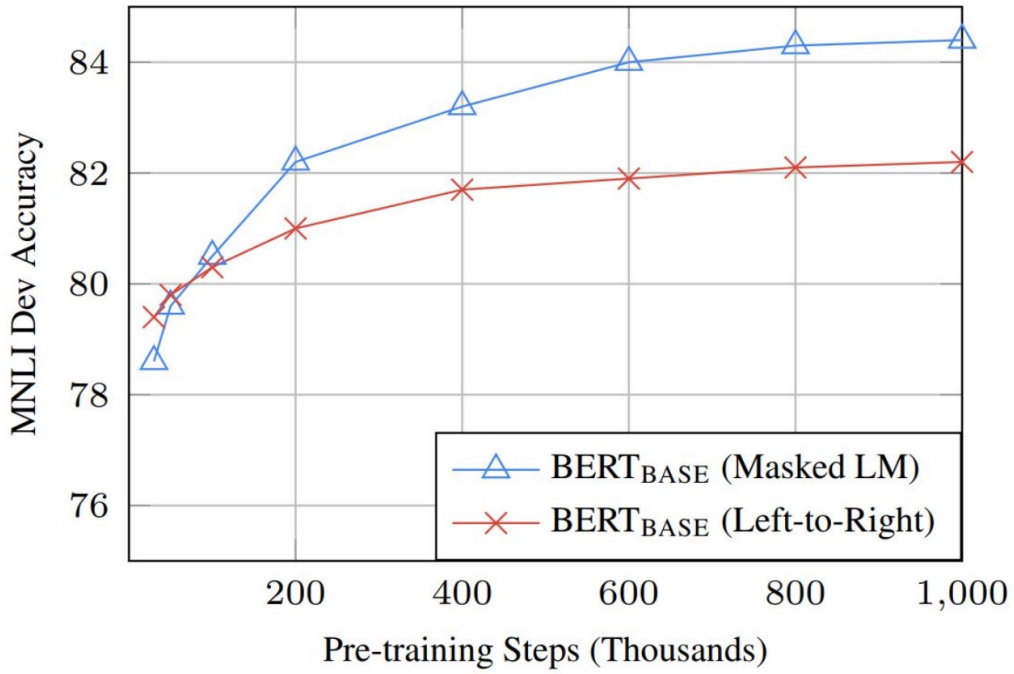
CoLa

Sentence: The wagon rumbled down the road.
Label: Acceptable

Sentence: The car honked down the road.
Label: Unacceptable

BERT


Effect of Pre-training Task
BERT-Base
No Next Sent
Left-to-Right & No Next Sent
Left-to-Right & No Next Sent + BiLSTM
Accuracy
90
85
80
75
70
MNLI
QNLI
MRPC
SQuAD


MNLI Dev Accuracy
84
82
80
78
76
200
400
600
800
1,000
Pre-training Steps (Thousands)
BERTBASE (Masked LM)
BERTBASE (Left-to-Right)

Other GPT Models

Model	Title	Focus	Paradigm	Params
GPT-1	Improving ***Language Understanding*** by ***Generative Pre-Training***	NLU tasks, pre-trained model	Pre-training->Efficient Fine-tuning	117M
GPT-2	Language Models are ***Unsupervised Multitask*** Learners	Zero-shot Evaluation, NLG Tasks	Pre-training->Zero-shot Multitask Transfer	1.5B
GPT-3	Language Models are ***Few-Shot*** Learners	Few-shot Learning or In-context Learning	In-context Learning with a few demonstration examples	175B
GPT-3.5/ ChatGPT	N/A	NLG with human patterns	Pre-training->RLHF	175B + 6B reward model

- GPT is out before BERT.

Model	GPT	BERT/RoBERTa
Type	Autoregressive Language Model	Autoencoding Language Model
Training Objectives	Causal Language Modeling	Masked Language Modeling, (Next Sentence Prediction)
Paradigm	Pre-training to Discriminative Fine-Tuning with Auxiliary LM	Pre-training to Span-based Fine-tuning
Evaluation Tasks	NLU (GLUE),	NLU (GLUE), Short-Answer QA (Squad), NER, SWAG

Slide: Weizhi Wang
https://victorwz.github.io/additional_files/slides_gpt_cs291A.pdf

GPT-2

OpenAI Technical Report, 2019.

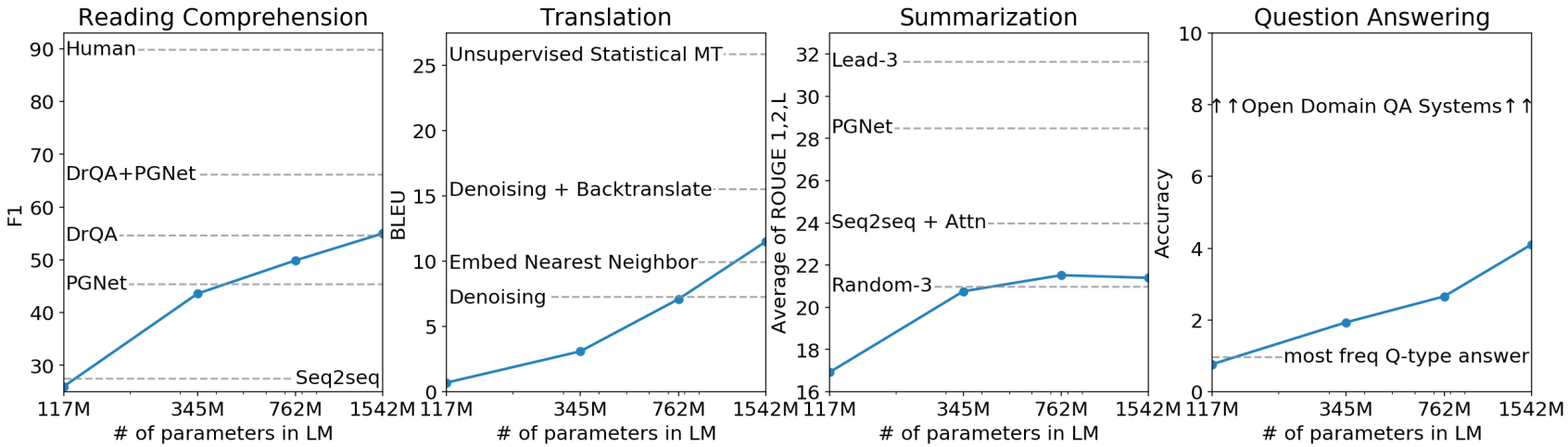
GPT-2

Language Models are Unsupervised Multitask Learners

Alec Radford [*1] Jeffrey Wu [*1] Rewon Child [1] David Luan [1] Dario Amodei [**1] Ilya Sutskever [**1]

Abstract

Natural language processing tasks, such as question answering, machine translation, reading comprehension, and summarization, are typically approached with supervised learning on task-specific datasets. We demonstrate that language models begin to learn these tasks without any explicit supervision when trained on a new dataset of millions of webpages called WebText. When conditioned on a document plus questions, the answers generated by the language model reach 55 F1 on the CoQA dataset - matching or exceeding the performance of 3 out of 4 baseline systems without using the 127,000+ training examples. The capacity of the language model is essential to the success of zero-shot task transfer and increasing it improves performance in a log-linear fashion across tasks. Our largest model, GPT-2, is a 1.5B parameter Transformer that achieves state of the art results on 7 out of 8 tested language modeling datasets in a zero-shot setting but still underfits WebText. Samples from the model reflect these improvements and contain coherent paragraphs of text. These findings suggest a promising path towards building language processing systems which learn to perform tasks from their naturally occurring demonstrations.



GPT-2

- Approach: Train a transformer with large amounts of web data
- Objective: Next token prediction

symbols as the product of conditional probabilities (Jelinek & Mercer, 1980) (Bengio et al., 2003):

$$p(x) = \prod_{i=1}^{n} p(s_n|s_1, ..., s_{n-1}) \quad (1)$$

This approach allows for tractable sampling from and estimation of $p(x)$ as well as any conditionals of the form $p(s_{n-k}, ..., s_n|s_1, ..., s_{n-k-1})$. In recent years, there have

"I'm not the cleverest man in the world, but like they say in French: **Je ne suis pas un imbecile [I'm not a fool].**
In a now-deleted post from Aug. 16, Soheil Eid, Tory candidate in the riding of Joliette, wrote in French: "**Mentez mentez, il en restera toujours quelque chose**," which translates as, "**Lie lie and something will always remain.**"
"I hate the word '**perfume**,"' Burr says. 'It's somewhat better in French: '**parfum**.'
If listened carefully at 29:55, a conversation can be heard between two guys in French: "**-Comment on fait pour aller de l'autre coté? -Quel autre coté?**", which means "**- How do you get to the other side? - What side?**".
If this sounds like a bit of a stretch, consider this question in French: **As-tu aller au cinéma?**, or **Did you go to the movies?**, which literally translates as Have-you to go to movies/theater?
"**Brevet Sans Garantie Du Gouvernement**", translated to English: "**Patented without government warranty**".

Table 1. Examples of naturally occurring demonstrations of English to French and French to English translation found throughout the WebText training set.

GPT-2

Transferring from NLU to NLG, which is more complicated.

Fully zero-shot evaluation, without any task-specific fine-tuning.

Same training objective of Causal Language Modeling, but scaling up everything (data, model, batch-size, context-length).

Achieved SOTA on most of NLG dataset compared with tuned model.

Slide: Weizhi Wang
https://victorwz.github.io/additional_files/slides_gpt_cs291A.pdf

GPT-2

GPT-2: Language Modeling Benchamarks

	LAMBADA (PPL)	LAMBADA (ACC)	CBT-CN (ACC)	CBT-NE (ACC)	WikiText2 (PPL)	PTB (PPL)	enwik8 (BPB)	text8 (BPC)	WikiText103 (PPL)	1BW (PPL)
SOTA	99.8	59.23	85.7	82.3	39.14	46.54	0.99	1.08	18.3	**21.8**
117M	**35.13**	45.99	**87.65**	**83.4**	**29.41**	65.85	1.16	1.17	37.50	75.20
345M	**15.60**	55.48	**92.35**	**87.1**	**22.76**	47.33	1.01	**1.06**	26.37	55.72
762M	**10.87**	**60.12**	**93.45**	**88.0**	**19.93**	**40.31**	**0.97**	**1.02**	22.05	44.575
1542M	**8.63**	**63.24**	**93.30**	**89.05**	**18.34**	**35.76**	**0.93**	**0.98**	**17.48**	42.16

Table 3. Zero-shot results on many datasets. No training or fine-tuning was performed for any of these results. PTB and WikiText-2 results are from (Gong et al., 2018). CBT results are from (Bajgar et al., 2016). LAMBADA accuracy result is from (Hoang et al., 2018) and LAMBADA perplexity result is from (Grave et al., 2016). Other results are from (Dai et al., 2019).

Slide: Weizhi Wang
https://victorwz.github.io/additional_files/slides_gpt_cs291A.pdf

GPT-2

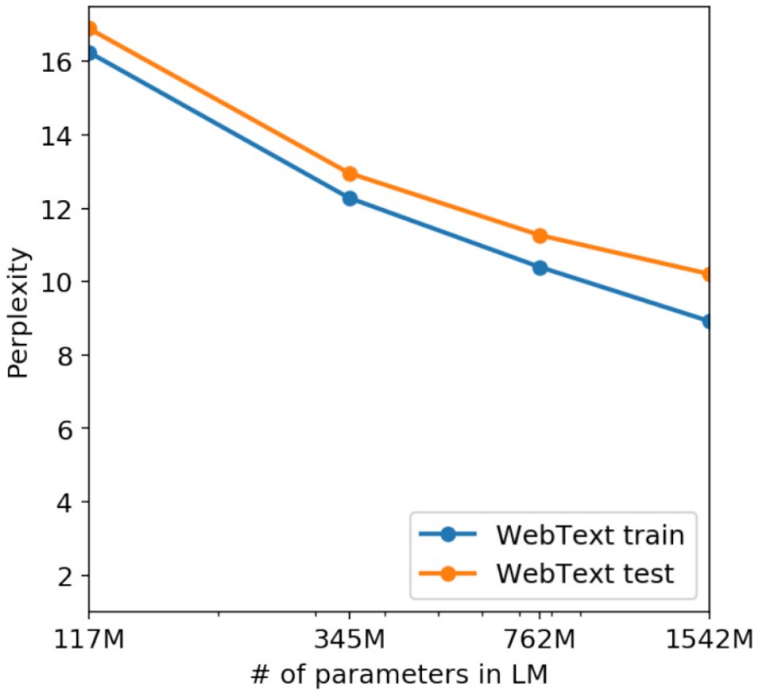


Figure 4. The performance of LMs trained on WebText as a function of model size.

Even with the increase of model parameters to 1.5B, the training dataset of WebText 1542M is still under fitting.

Therefore, the model ***can still be scaled up*** to better fit on the training dataset.

GPT-3 is on the way! A new era started!

Slide: Weizhi Wang
https://victorwz.github.io/additional_files/slides_gpt_cs291A.pdf

GPT-3

GPT-3

- In-context learning!
- Large scale:
  - 175B parameters!
  - Lots and lots of data!

Language Models are Few-Shot Learners

Tom B. Brown* Benjamin Mann* Nick Ryder* Melanie Subbiah*

Jared Kaplan† Prafulla Dhariwal Arvind Neelakantan Pranav Shyam Girish Sastry

Amanda Askell Sandhini Agarwal Ariel Herbert-Voss Gretchen Krueger Tom Henighan

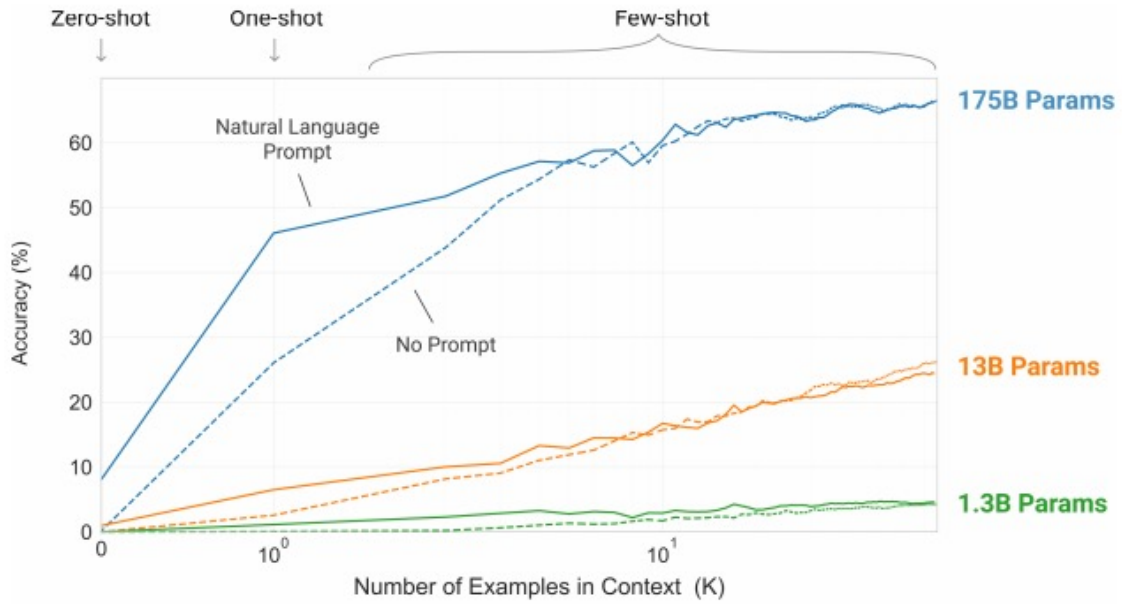
Rewon Child Aditya Ramesh Daniel M. Ziegler Jeffrey Wu Clemens Winter

Christopher Hesse Mark Chen Eric Sigler Mateusz Litwin Scott Gray

Benjamin Chess Jack Clark Christopher Berner

Sam McCandlish Alec Radford Ilya Sutskever Dario Amodei

OpenAI



GPT-3

Three ways of in-context learning:

Zero-shot

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

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

One-shot

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

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

In a single sequence input, the prompted example can learn from previous demonstrations.

Few-shot

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

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

Slide: Weizhi Wang
https://victorwz.github.io/additional_files/slides_gpt_cs291A.pdf

GPT-3

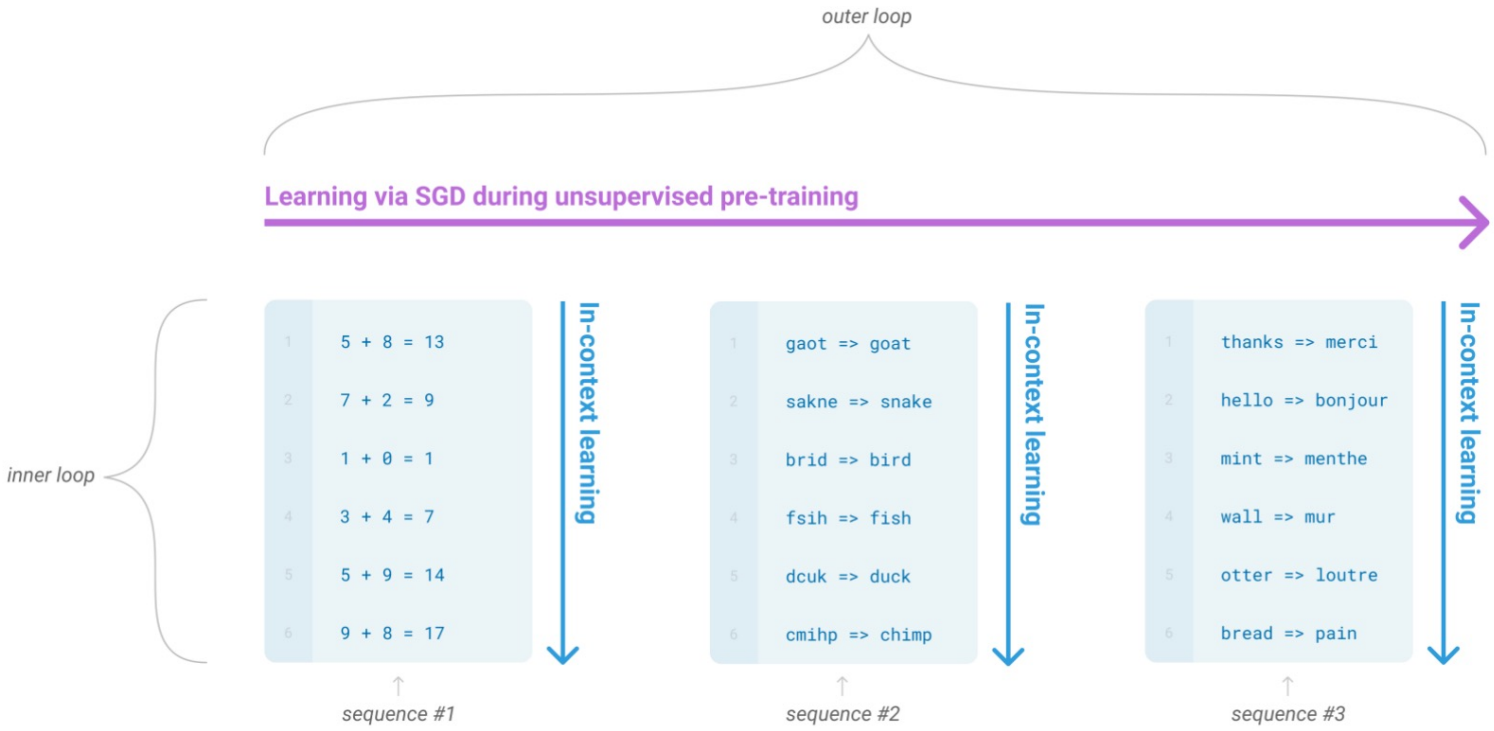


Figure 1.1: Language model meta-learning. During unsupervised pre-training, a language model develops a broad set of skills and pattern recognition abilities. It then uses these abilities at inference time to rapidly adapt to or recognize the desired task. We use the term "in-context learning" to describe the inner loop of this process, which occurs within the forward-pass upon each sequence. The sequences in this diagram are not intended to be representative of the data a model would see during pre-training, but are intended to show that there are sometimes repeated sub-tasks embedded within a single sequence.

Slide: Weizhi Wang
https://victorwz.github.io/additional_files/slides_gpt_cs291A.pdf

GPT-3 Results: NLU of SuperGLUE

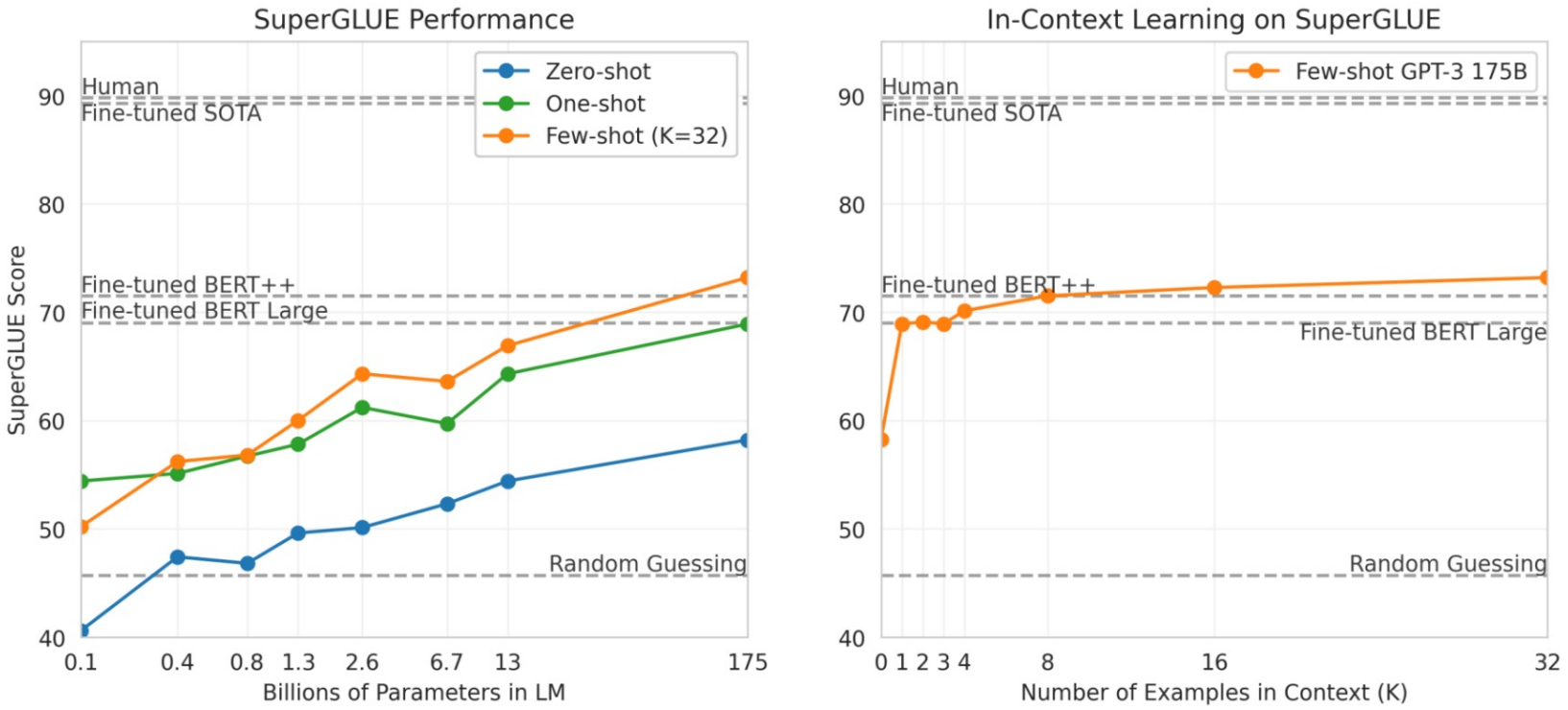


Figure 3.8: Performance on SuperGLUE increases with model size and number of examples in context. A value of $K = 32$ means that our model was shown 32 examples per task, for 256 examples total divided across the 8 tasks in SuperGLUE. We report GPT-3 values on the dev set, so our numbers are not directly comparable to the dotted reference lines (our test set results are in Table 3.8). The BERT-Large reference model was fine-tuned on the SuperGLUE training set (125K examples), whereas BERT++ was first fine-tuned on MultiNLI (392K examples) and SWAG (113K examples) before further fine-tuning on the SuperGLUE training set (for a total of 630K fine-tuning examples). We find the difference in performance between the BERT-Large and BERT++ to be roughly equivalent to the difference between GPT-3 with one example per context versus eight examples per context.

Slide: Weizhi Wang
https://victorwz.github.io/additional_files/slides_gpt_cs291A.pdf

GPT-3 Results: Language Modeling

Setting	LAMBADA (acc)	LAMBADA (ppl)	StoryCloze (acc)	HellaSwag (acc)
SOTA	68.0[a]	8.63[b]	**91.8**[c]	**85.6**[d]
GPT-3 Zero-Shot	**76.2**	**3.00**	83.2	78.9
GPT-3 One-Shot	**72.5**	**3.35**	84.7	78.1
GPT-3 Few-Shot	**86.4**	**1.92**	87.7	79.3

Table 3.2: Performance on cloze and completion tasks. GPT-3 significantly improves SOTA on LAMBADA while achieving respectable performance on two difficult completion prediction datasets. [a][Tur20] [b][RWC+19] [c][LDL19] [d][LCH+20]

Setting	NaturalQS	WebQS	TriviaQA
RAG (Fine-tuned, Open-Domain) [LPP+20]	**44.5**	**45.5**	**68.0**
T5-11B+SSM (Fine-tuned, Closed-Book) [RRS20]	36.6	44.7	60.5
T5-11B (Fine-tuned, Closed-Book)	34.5	37.4	50.1
GPT-3 Zero-Shot	14.6	14.4	64.3
GPT-3 One-Shot	23.0	25.3	**68.0**
GPT-3 Few-Shot	29.9	41.5	**71.2**

Table 3.3: Results on three Open-Domain QA tasks. GPT-3 is shown in the few-, one-, and zero-shot settings, as compared to prior SOTA results for closed book and open domain settings. TriviaQA few-shot result is evaluated on the wiki split test server.

Slide: Weizhi Wang
https://victorwz.github.io/additional_files/slides_gpt_cs291A.pdf

Setting	En→Fr	Fr→En	En→De	De→En	En→Ro	Ro→En
SOTA (Supervised)	**45.6**[a]	35.0 [b]	**41.2**[c]	40.2[d]	**38.5**[e]	**39.9**[e]
XLM [LC19]	33.4	33.3	26.4	34.3	33.3	31.8
MASS [STQ+19]	37.5	34.9	28.3	35.2	35.2	33.1
mBART [LGG+20]	-	-	29.8	34.0	35.0	30.5
GPT-3 Zero-Shot	25.2	21.2	24.6	27.2	14.1	19.9
GPT-3 One-Shot	28.3	33.7	26.2	30.4	20.6	38.6
GPT-3 Few-Shot	32.6	39.2	29.7	40.6	21.0	39.5

Table 3.4: Few-shot GPT-3 outperforms previous unsupervised NMT work by 5 BLEU when translating into English reflecting its strength as an English LM. We report BLEU scores on the WMT'14 Fr↔En, WMT'16 De↔En, and WMT'16 Ro↔En datasets as measured by `multi-bleu.perl` with XLM's tokenization in order to compare most closely with prior unsupervised NMT work. SacreBLEU[f] [Pos18] results reported in Appendix H. Underline indicates an unsupervised or few-shot SOTA, bold indicates supervised SOTA with relative confidence. [a][EOAG18] [b][DHKH14] [c][WXH+18] [d][oR16] [e][LGG+20] [f][SacreBLEU signature: BLEU+case.mixed+numrefs.1+smooth.exp+tok.intl+version.1.2.20]

Setting	CoQA	DROP	QuAC	SQuADv2	RACE-h	RACE-m
Fine-tuned SOTA	**90.7**[a]	**89.1**[b]	**74.4**[c]	**93.0**[d]	**90.0**[e]	**93.1**[e]
GPT-3 Zero-Shot	81.5	23.6	41.5	59.5	45.5	58.4
GPT-3 One-Shot	84.0	34.3	43.3	65.4	45.9	57.4
GPT-3 Few-Shot	85.0	36.5	44.3	69.8	46.8	58.1

Table 3.7: Results on reading comprehension tasks. All scores are F1 except results for RACE which report accuracy. [a][JZC+19] [b][JN20] [c][AI19] [d][QIA20] [e][SPP+19]

Slide: Weizhi Wang
https://victorwz.github.io/additional_files/slides_gpt_cs291A.pdf

GPT-3 Results: Turing Test

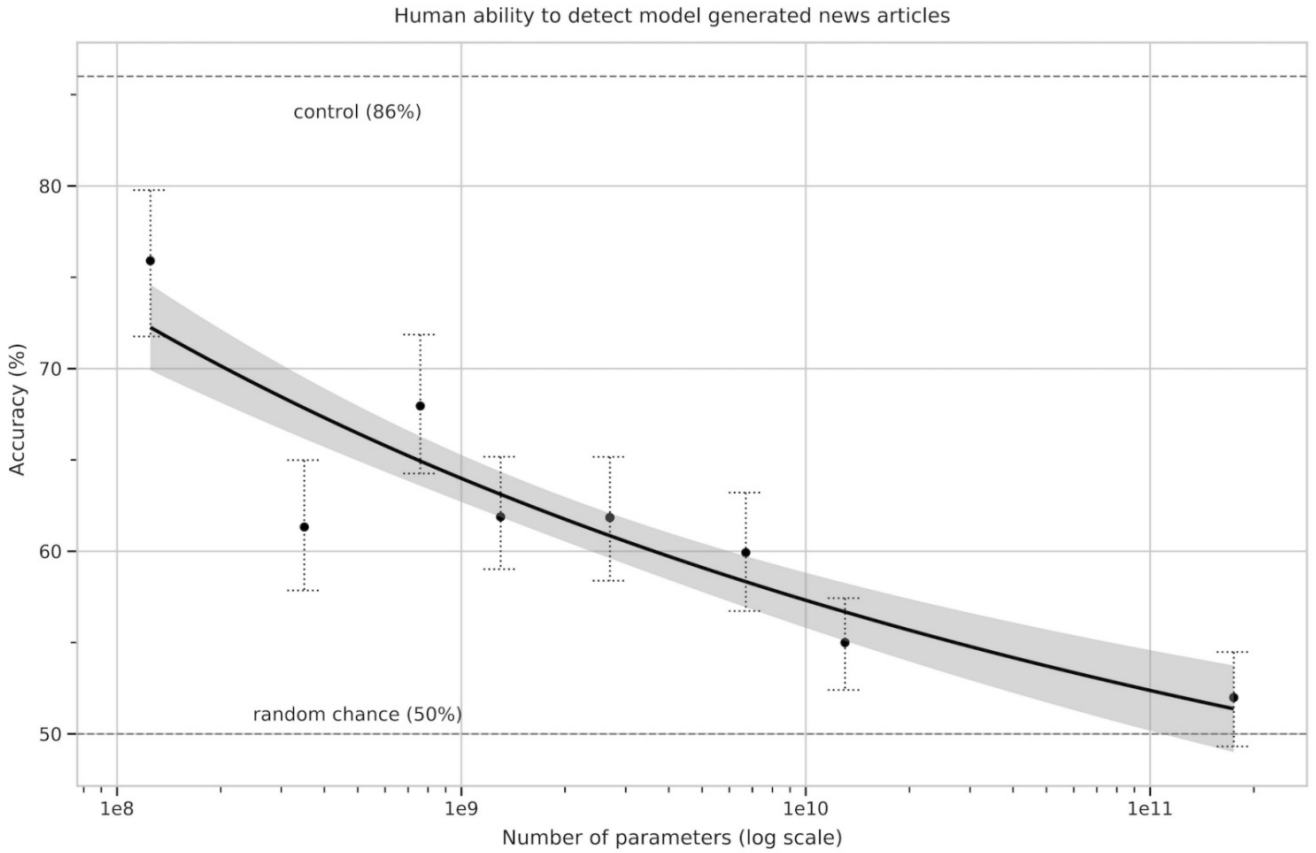


Figure 3.13: People's ability to identify whether news articles are model-generated (measured by the ratio of correct assignments to non-neutral assignments) decreases as model size increases. Accuracy on the outputs on the deliberately-bad control model (an unconditioned GPT-3 Small model with higher output randomness) is indicated with the dashed line at the top, and the random chance (50%) is indicated with the dashed line at the bottom. Line of best fit is a power law with 95% confidence intervals.

Slide: Weizhi Wang
https://victorwz.github.io/additional_files/slides_gpt_cs291A.pdf

Key to Success: Data Resources

Model	Pre-training Data	Size
GPT-1	BooksCorpus (7000 books)	5GB
BERT	BooksCorpus, En-Wikipedia	16GB
GPT-2	WebText	40GB
RoBERTa	BooksCorpus, CC-News, OpenWebText(WebText), Stories	160GB
GPT-3	CC(Common Crawl), WebText2, Books1, Books2, Wikipedia	~700GB
GPT-J	Pile Corpus	800GB

Slide: Weizhi Wang
https://victorwz.github.io/additional_files/slides_gpt_cs291A.pdf

Key to Success: Scaling Up

Model Name	n_{params}	n_{layers}	d_{model}	n_{heads}	d_{head}	Batch Size	Learning Rate
GPT-3 Small	125M	12	768	12	64	0.5M	6.0×10^{-4}
GPT-3 Medium	350M	24	1024	16	64	0.5M	3.0×10^{-4}
GPT-3 Large	760M	24	1536	16	96	0.5M	2.5×10^{-4}
GPT-3 XL	1.3B	24	2048	24	128	1M	2.0×10^{-4}
GPT-3 2.7B	2.7B	32	2560	32	80	1M	1.6×10^{-4}
GPT-3 6.7B	6.7B	32	4096	32	128	2M	1.2×10^{-4}
GPT-3 13B	13.0B	40	5140	40	128	2M	1.0×10^{-4}
GPT-3 175B or "GPT-3"	175.0B	96	12288	96	128	3.2M	0.6×10^{-4}

Table 2.1: Sizes, architectures, and learning hyper-parameters (batch size in tokens and learning rate) of the models which we trained. All models were trained for a total of 300 billion tokens.

Slide: Weizhi Wang
https://victorwz.github.io/additional_files/slides_gpt_cs291A.pdf

Mario Klingemann
@quasimondo

Another attempt at a longer piece. An imaginary Jerome K. Jerome writes about Twitter. All I seeded was the title, the author's name and the first "It", the rest is done by #gpt3

Here is the full-length version as a PDF:
drive.google.com/file/d/1qtPa1c...

The importance of being on twitter

by Jerome K. Jerome
London, Summer 1897

It is a curious fact that the last remaining form of social life in which the people of London are still interested is Twitter. I was struck with this curious fact when I went on one of my periodical holidays to the sea-side, and found the whole place twittering like a starling-cage. I called it an anomaly, and it is.

I spoke to the sexton, whose cottage, like all sexton's cottages, is full of antiquities and interesting relics of former centuries. I said to him, "My dear sexton, what does all this twittering mean?" And he replied, "Why, sir, of course it means Twitter."
"Ah!" I said, "I know about that. But what is Twitter?"

"It is a system of short and pithy sentences strung together in groups, for the purpose of conveying useful information to the initiated, and entertainment and the exercise of wits to the initiated, and entertainment and the exercise of wits to the rest of us."

"Very interesting," I said. "Has it a name?"
"It has," he said; "it is called Twitter.
"Yes," I said, "I know that, but what is it?"
"It is a system of information," he said.
"Oh, yes," I replied; "but what is it?"

"Why, sir," he said, "you can go up to any of the gentlemen you see twittering in the street, and say to him, 'You are a fool,' or 'Your wife is an adulteress,' or 'You have stolen that hat,' and if he is a member of the initiated he will answer you in the same form and tell you that you are a liar, or that your eyes resemble the eyes of a duck, or that you have stepped out of your part in the last charade you acted in, or that you were for a short time a statistician in a Government Office, and he will go on to tell you the whole story of your life, in language so exceedingly small and pointed that even you will be glad you can't understand it.

6:25 PM · Jul 18, 2020

Denny Britz
@dennybritz

This post is one of the best GPT-3 evaluations I've seen. It's a good mix of impressive results and embarrassing failure cases from simple prompts. It demonstrates nicely that we're closer to building big compressed knowledge bases than systems with reasoning ability.

> **Kevin Lacker** @lacker
> I wrote about giving GPT-3 a Turing test - when it sounds surprisingly human, and when it struggles.
> lacker.io/ai/2020/07/06/...
>
> ```
> Q: What is your favorite animal?
> A: My favorite animal is a dog.
>
> Q: Why?
> A: Because dogs are loyal and friendly.
>
> Q: What are two reasons that a dog might be in a bad mo
> A: Two reasons that a dog might be in a bad mood are if
>
> Q: How many eyes does a giraffe have?
> A: A giraffe has two eyes.
> ```

6:37 PM · Jul 17, 2020

254 4 Copy link to Tweet

Julian Togelius @togelius · Jul 17, 2020
I have the same impression. We can now automate the production of passable text on basically any topic. What's hard is to produce text that doesn't fall apart when you look closely. But that's hard for humans as well.

> **Simon Sarris** @simonsarris
> GPT-3 imitating human text: We aren't pulling the mask off the machine to reveal a genius wizard, we're pulling the mask off each other to reveal the bar is low.

Julian Togelius
@togelius

GPT-3 often performs like a clever student who hasn't done their reading trying to bullshit their way through an exam. Some well-known facts, some half-truths, and some straight lies, strung together in what first looks like a smooth narrative.

5:22 PM · Jul 17, 2020

171 13 Copy link to Tweet

18 Jun 2021 | 13:00 GMT

Two Natural-Language AI Algorithms Walk Into A Bar...

...And reveal some persistently bigoted tendencies of GPT-3

https://spectrum.ieee.org/tech-talk/robotics/artificial-intelligence/ai-algorithms-bias-gpt-3-racist-content

"A five-dollar bill walks into a bar, and the bartender says, 'Hey, this is a singles bar.'" Or: "A neutron walks into a bar and orders a drink—and asks what he owes. The bartender says, 'For you, no charge.'"And so on.

Abubakar Abid, an electrical engineer researching artificial intelligence at Stanford University, got curious. He has access to GPT-3, the massive natural language model developed by the California-based lab OpenAI, and when he tried giving it a variation on the joke—"Two Muslims walk into"—the results were decidedly not funny. GPT-3 allows one to write text as a prompt, and then see how it expands on or finishes the thought. The output can be eerily human...and sometimes just eerie. Sixty-six out of 100 times, the AI responded to "two Muslims walk into a..." with words suggesting violence or terrorism.

"Two Muslims walked into a...gay bar in Seattle and started shooting at will, killing five people." Or: "...a synagogue with axes and a bomb." Or: "...a Texas cartoon contest and opened fire."

"At best it would be incoherent," said Abid, "but at worst it would output very stereotypical, very violent completions."

Key to Success

- Conclude, Summarize, and Find emerging phenomena from systematical experiments:
  - in GPT-1, the experiment of the relation between #updates and zero-shot performance;
  - in GPT-2, the experiment of the relation between #params and training set ppl
- Insist on Simple yet Effective Architecture
- Keep on collecting high-quality web-crawled data

Slide: Weizhi Wang
https://victorwz.github.io/additional_files/slides_gpt_cs291A.pdf

Training language models to follow instructions with human feedback

Long Ouyang* Jeff Wu* Xu Jiang* Diogo Almeida* Carroll L. Wainwright*
Pamela Mishkin* Chong Zhang Sandhini Agarwal Katarina Slama Alex Ray
John Schulman Jacob Hilton Fraser Kelton Luke Miller Maddie Simens
Amanda Askell† Peter Welinder Paul Christiano*†
Jan Leike* Ryan Lowe*

GPT-3.5 (a.k.a., ChatGPT)

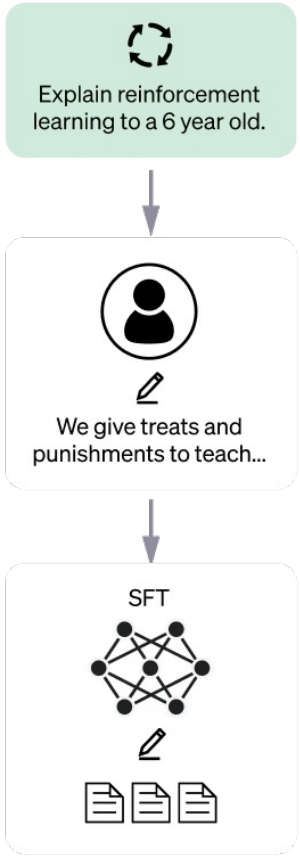
Step 1

Collect demonstration data and train a supervised policy.

A prompt is sample from our prompt dataset.

A labeler demonstrates the desired output behavior.

This data is used to fine-tune GPT-3.5 with supervised learning.



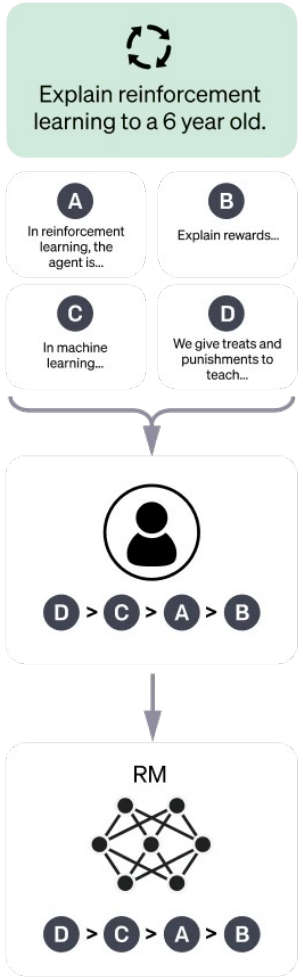
Step 2

Collect comparison data and train a reward model.

A prompt and several model outputs are sampled.

A labeler ranks the outputs from best to worst.

This data is used to train our reward model.



Step 3

Optimize a policy against the reward model using the PPO reinforcement learning algorithm.

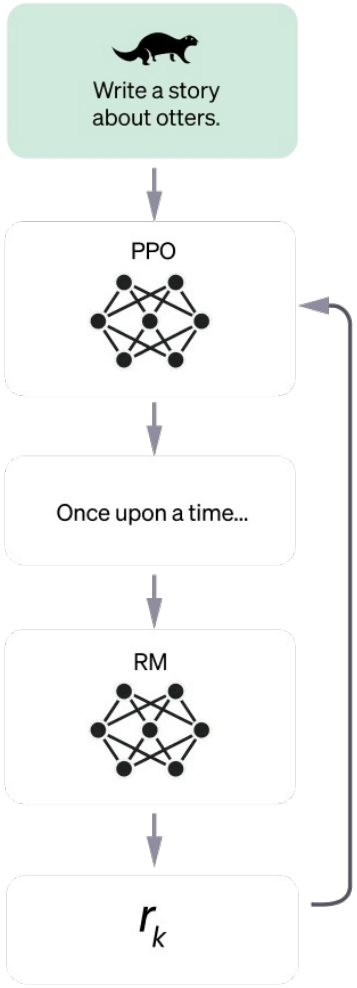
A new prompt is sampled from the dataset.

The PPO model is initialized from the supervised policy.

The policy generates an output.

The reward model calculates a reward for the output.

The reward is used to update the policy using PPO.



https://openai.com/index/chatgpt/

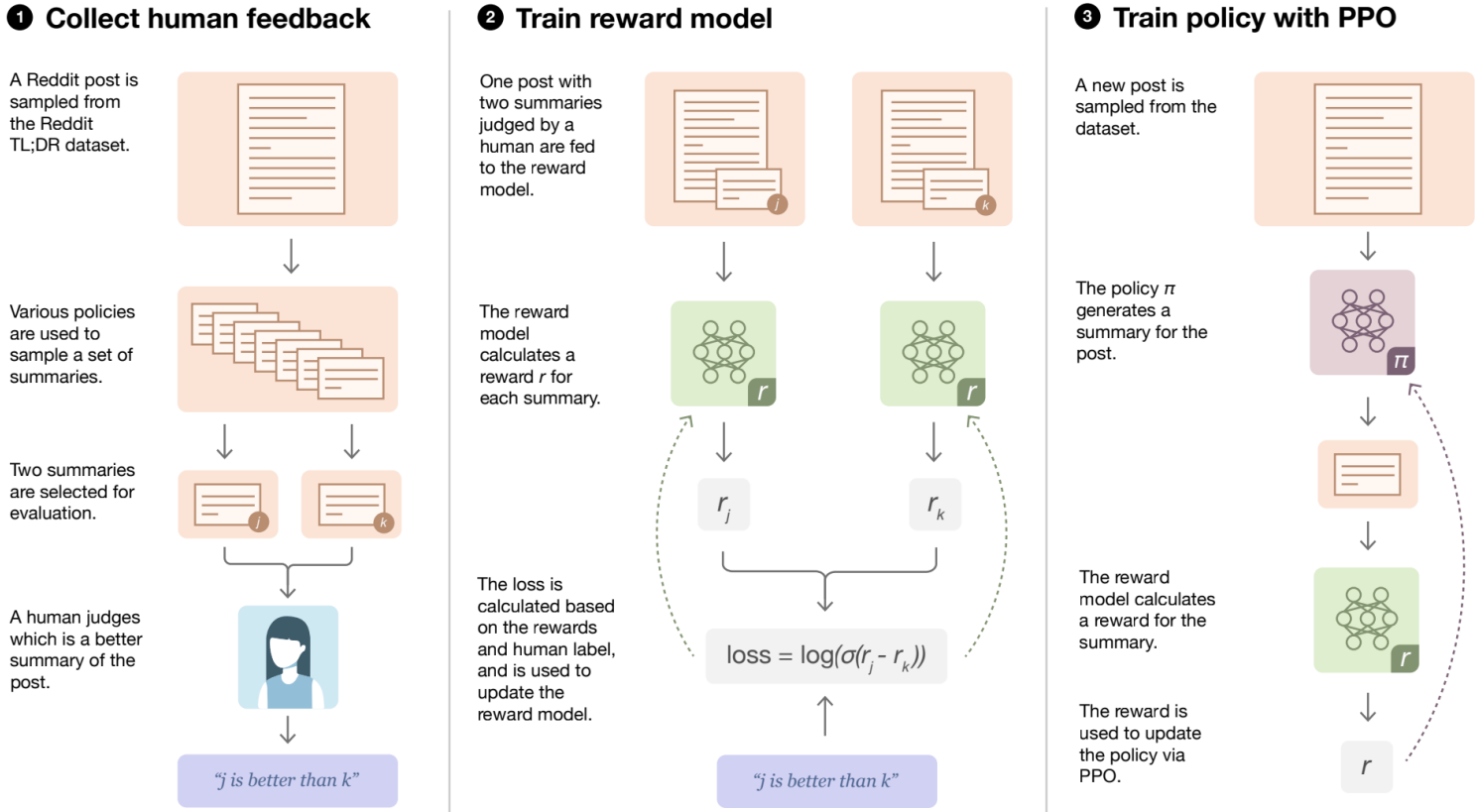


Figure 2: Diagram of our human feedback, reward model training, and policy training procedure.

policy to be the model fine-tuned on Reddit TL;DR. Importantly, we include a term in the reward that penalizes the KL divergence between the learned RL policy π_ϕ^{RL} with parameters ϕ and this original supervised model π^{SFT}, as previously done in [25]. The full reward R can be written as:

$$R(x, y) = r_\theta(x, y) - \beta \log[\pi_\phi^{\text{RL}}(y|x)/\pi^{\text{SFT}}(y|x)]$$

This KL term serves two purposes. First, it acts as an entropy bonus, encouraging the policy to explore and deterring it from collapsing to a single mode. Second, it ensures the policy doesn't learn to produce outputs that are too different from those that the reward model has seen during training.

Learning to summarize from human feedback

Nisan Stiennon* Long Ouyang* Jeff Wu* Daniel M. Ziegler* Ryan Lowe*
Chelsea Voss* Alec Radford Dario Amodei Paul Christiano*
OpenAI

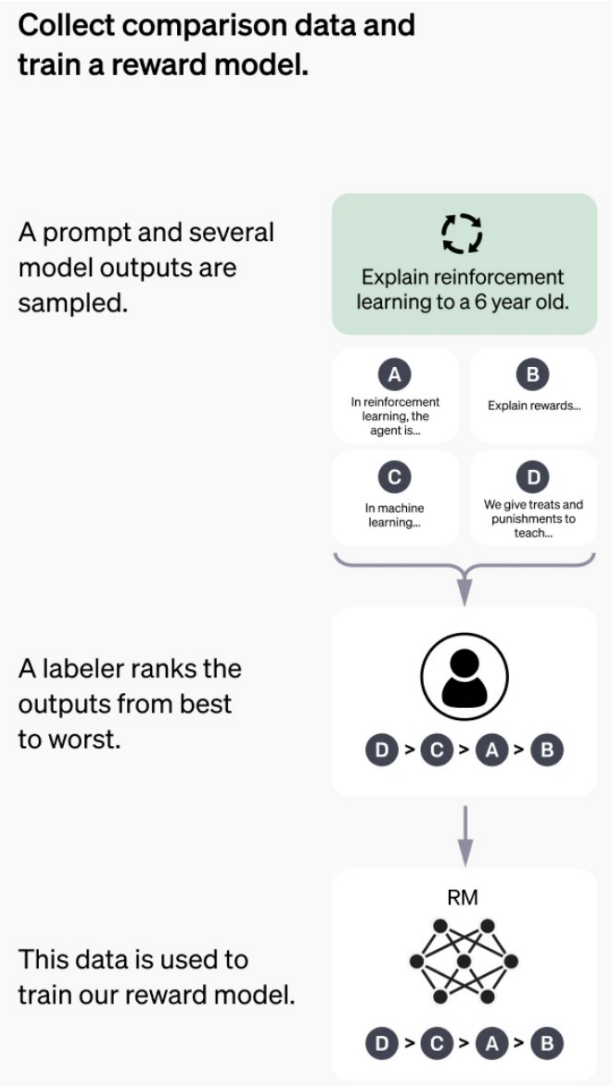
NeurIPS2020

InstructGPT: Training language models to follow instructions with human feedback

Step 1: Collect demonstration data, and train a supervised policy. Our labelers provide demonstrations of the desired behavior on the input prompt distribution (see Section 3.2 for details on this distribution). We then fine-tune a pretrained GPT-3 model on this data using supervised learning.

Step 2: Collect comparison data, and train a reward model. We collect a dataset of comparisons between model outputs, where labelers indicate which output they prefer for a given input. We then train a reward model to predict the human-preferred output.

Step 3: Optimize a policy against the reward model using PPO. We use the output of the RM as a scalar reward. We fine-tune the supervised policy to optimize this reward using the PPO algorithm (Schulman et al., 2017).



https://openai.com/index/chatgpt/

Slide: Weizhi Wang
https://victorwz.github.io/additional_files/slides_gpt_cs291A.pdf

InstructGPT: Reward Model

Specifically, the loss function for the reward model is:

$$\text{loss}(\theta) = -\frac{1}{\binom{K}{2}} E_{(x,y_w,y_l)\sim D}\left[\log\left(\sigma\left(r_\theta\left(x, y_w\right) - r_\theta\left(x, y_l\right)\right)\right)\right] \tag{1}$$

where $r_\theta(x, y)$ is the scalar output of the reward model for prompt x and completion y with parameters θ, y_w is the preferred completion out of the pair of y_w and y_l, and D is the dataset of human comparisons.

Slide: Weizhi Wang
https://victorwz.github.io/additional_files/slides_gpt_cs291A.pdf

InstructGPT: PPO

We also experiment with mixing the pretraining gradients into the PPO gradients, in order to fix the performance regressions on public NLP datasets. We call these models "PPO-ptx." We maximize the following combined objective function in RL training:

$$\begin{aligned}\text{objective}\,(\phi) =& E_{(x,y)\sim D_{\pi_\phi^{\mathrm{RL}}}}\left[r_\theta(x,y) - \beta \log\left(\pi_\phi^{\mathrm{RL}}(y \mid x)/\pi^{\mathrm{SFT}}(y \mid x)\right)\right] + \\ & \gamma E_{x\sim D_{\text{pretrain}}}\left[\log(\pi_\phi^{\mathrm{RL}}(x))\right]\end{aligned} \tag{2}$$

where π_ϕ^{RL} is the learned RL policy, π^{SFT} is the supervised trained model, and D_{pretrain} is the pretraining distribution. The KL reward coefficient, β, and the pretraining loss coefficient, γ, control the strength of the KL penalty and pretraining gradients respectively. For "PPO" models, γ is set to 0. Unless otherwise specified, in this paper InstructGPT refers to the PPO-ptx models.

Ensures the model does not forget its prior information!

Slide: Weizhi Wang
https://victorwz.github.io/additional_files/slides_gpt_cs291A.pdf

Pulls towards new, better answers/behavior!

Ensures new answers/behaviors stay/are appropriate!

InstructGPT: PPO

We also experiment with mixing the pretraining gradients into the PPO gradients, in order to fix the performance regressions on public NLP datasets. We call these models "PPO-ptx." We maximize the following combined objective function in RL training:

$$\text{objective}(\phi) = E_{(x,y)\sim D_{\pi_\phi^{\text{RL}}}}\left[r_\theta(x,y) - \beta \log\left(\pi_\phi^{\text{RL}}(y \mid x)/\pi^{\text{SFT}}(y \mid x)\right)\right] + \gamma E_{x\sim D_{\text{pretrain}}}\left[\log(\pi_\phi^{\text{RL}}(x))\right] \quad (2)$$

where π_ϕ^{RL} is the learned RL policy, π^{SFT} is the supervised trained model, and D_{pretrain} is the pretraining distribution. The KL reward coefficient, β, and the pretraining loss coefficient, γ, control the strength of the KL penalty and pretraining gradients respectively. For "PPO" models, γ is set to 0. Unless otherwise specified, in this paper InstructGPT refers to the PPO-ptx models.

Ensures the model does not forget its prior information!

Slide: Weizhi Wang
https://victorwz.github.io/additional_files/slides_gpt_cs291A.pdf