

CME 295: Transformers & Large Language Models

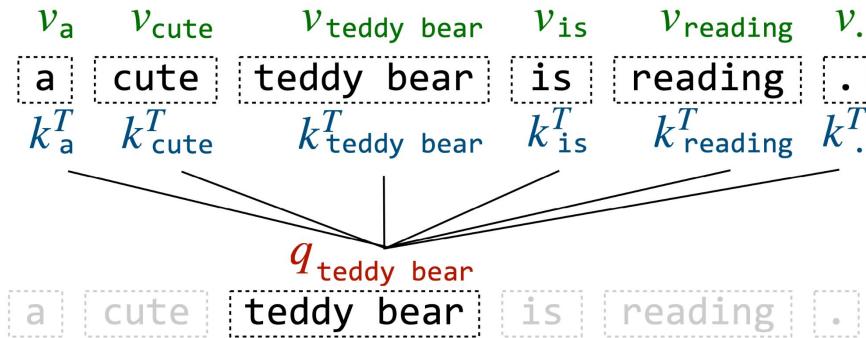


Afshine Amidi & Shervine Amidi

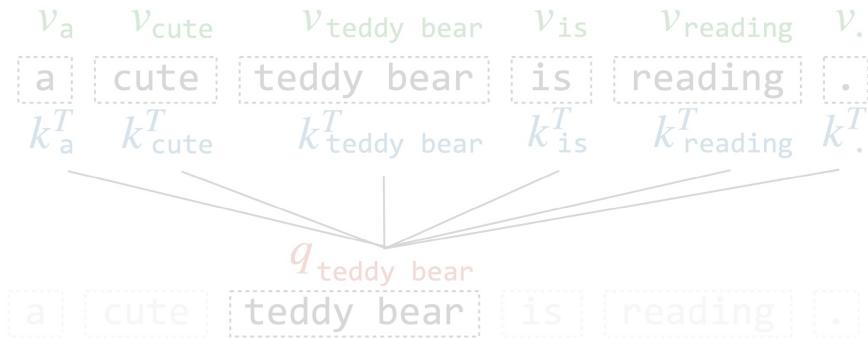


Recap of last episode...

Recap of last episode...



Recap of last episode...

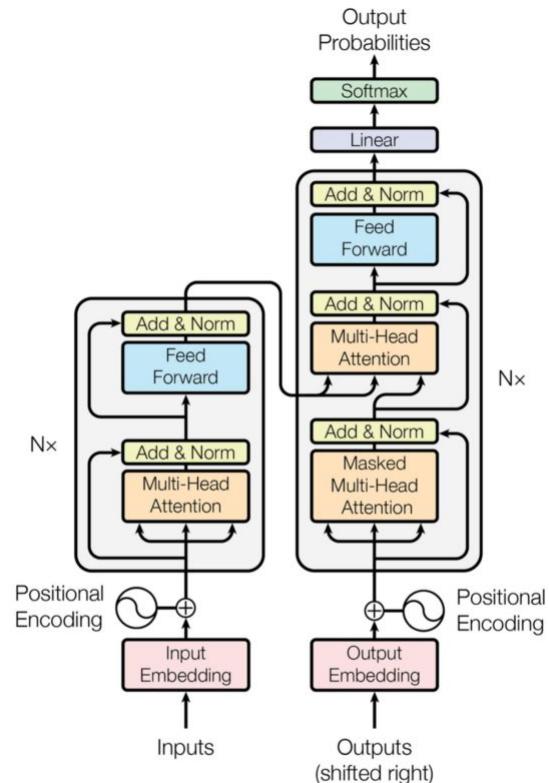


$$\text{softmax} \left(\frac{QK^T}{\sqrt{d_k}} \right) V$$

Recap of last episode...



$$\text{softmax} \left(\frac{QK^T}{\sqrt{d_k}} \right) V$$

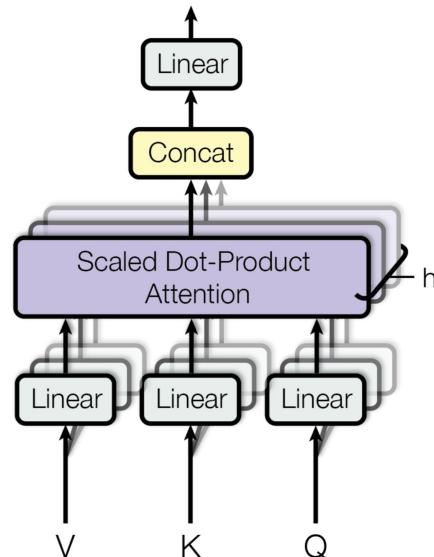


Recap of last episode...

$$\text{softmax} \left(\frac{QK^T}{\sqrt{d_k}} \right) V$$

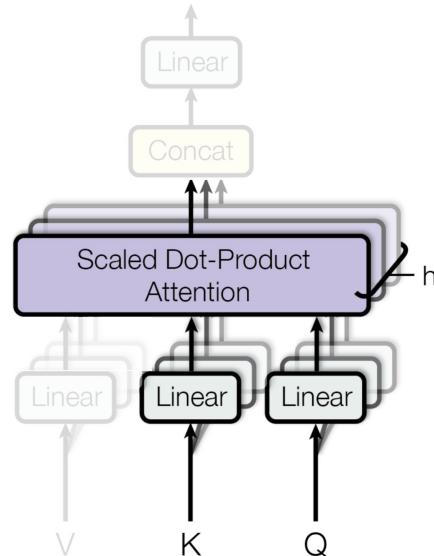
Recap of last episode...

$$\text{softmax} \left(\frac{QK^T}{\sqrt{d_k}} \right) V$$



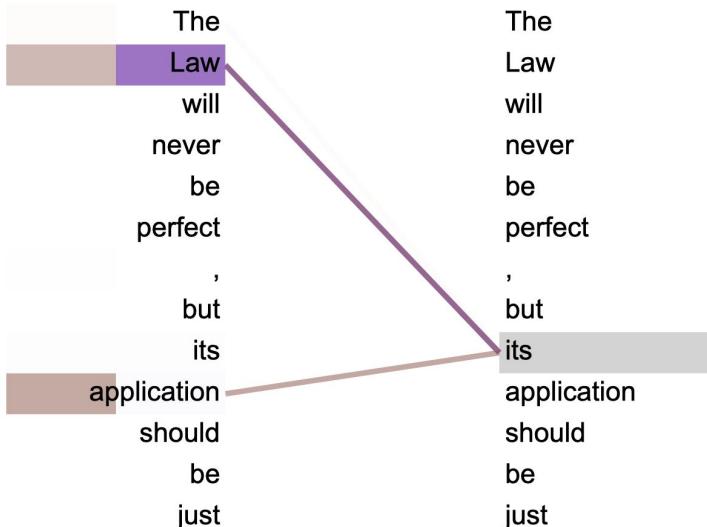
Recap of last episode...

$$\text{softmax} \left(\frac{QK^T}{\sqrt{d_k}} \right) V$$

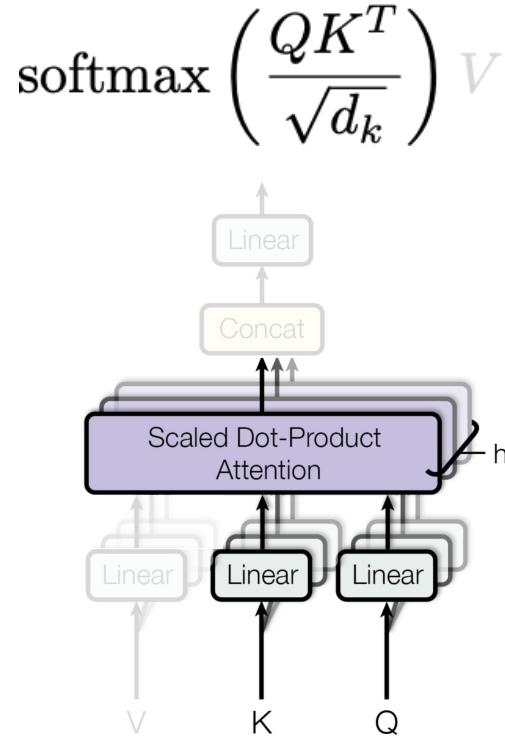


Recap of last episode...

Attention map



"Two attention heads, also in layer 5 of 6, apparently involved in anaphora resolution. [...] Isolated attentions from just the word 'its' for attention heads 5 and 6."



Suggested reading: original Transformer paper

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

Lukasz Kaiser*

Google Brain

lukaszkaiser@google.com

Illia Polosukhin* ‡

illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best



Transformers & Large Language Models

Position embeddings

Layer normalization

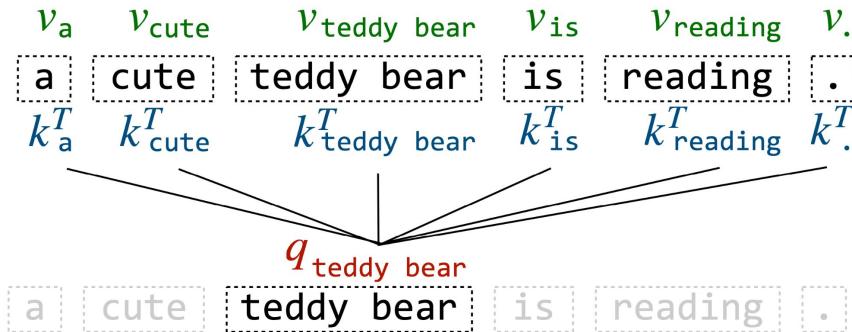
Attention approximation

Transformer-based models

BERT deep dive

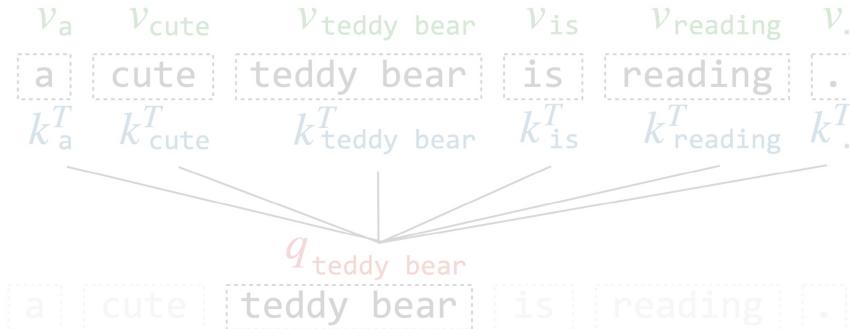
Need for position information

Motivation. Direct links "lose" position info

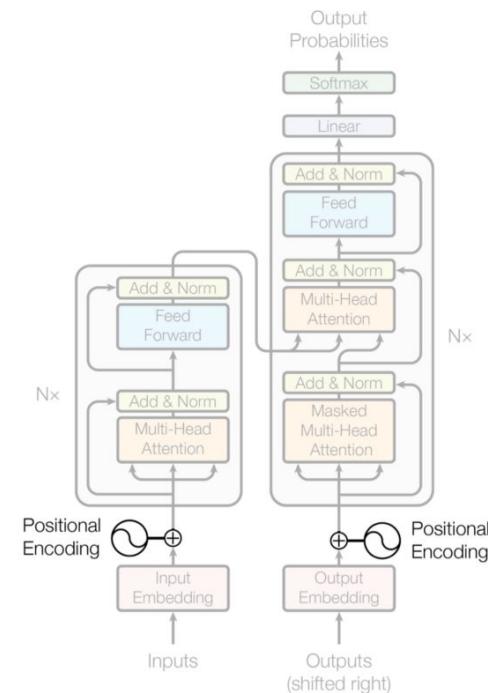


Need for position information

Motivation. Direct links "lose" position info

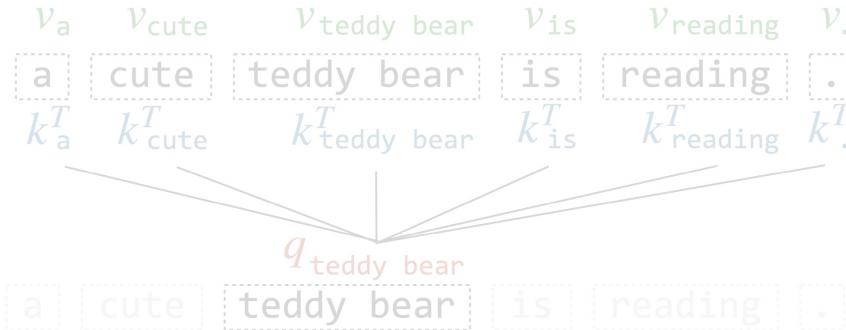


Idea. Add learned position-specific embedding to token vector



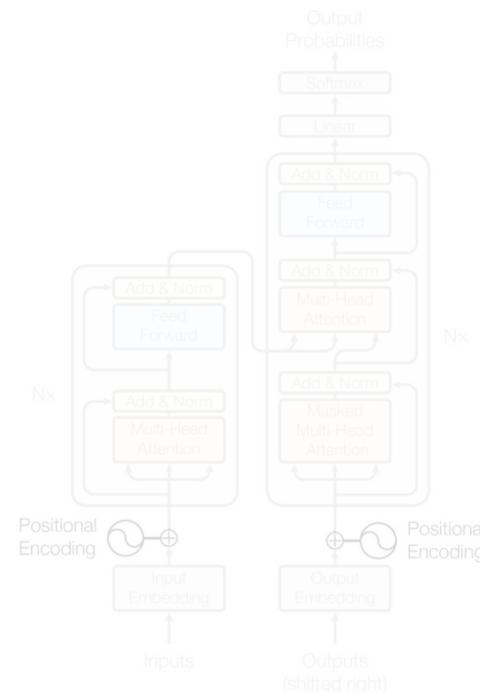
Need for position information

Motivation. Direct links "lose" position info



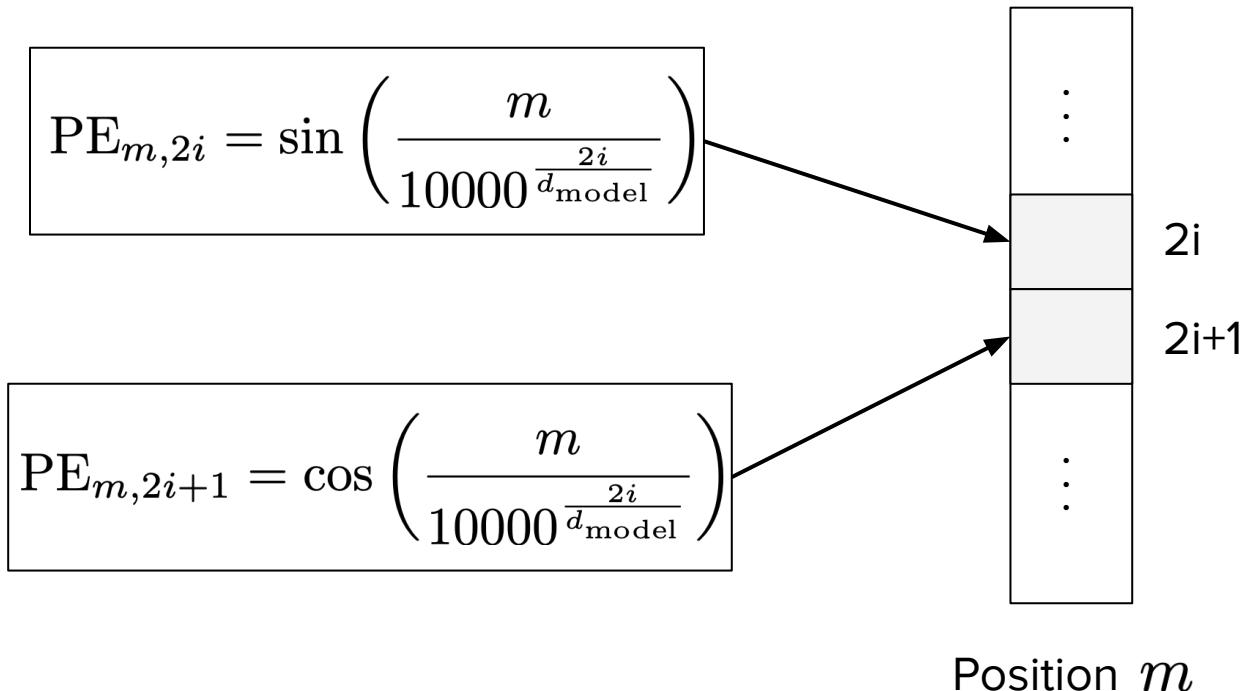
Idea. Add learned position-specific embedding to token vector

Limitations. Need to retrain for longer sequences



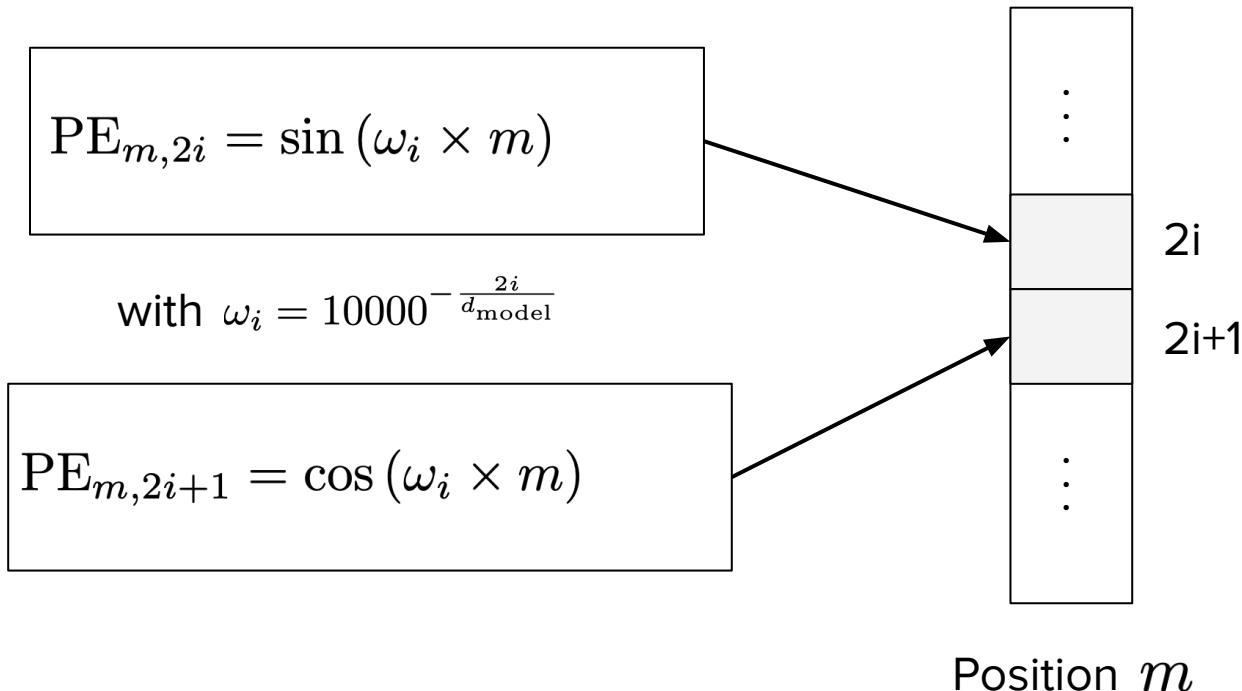
Hardcoded position embeddings

Idea. Hardcode values of position embedding



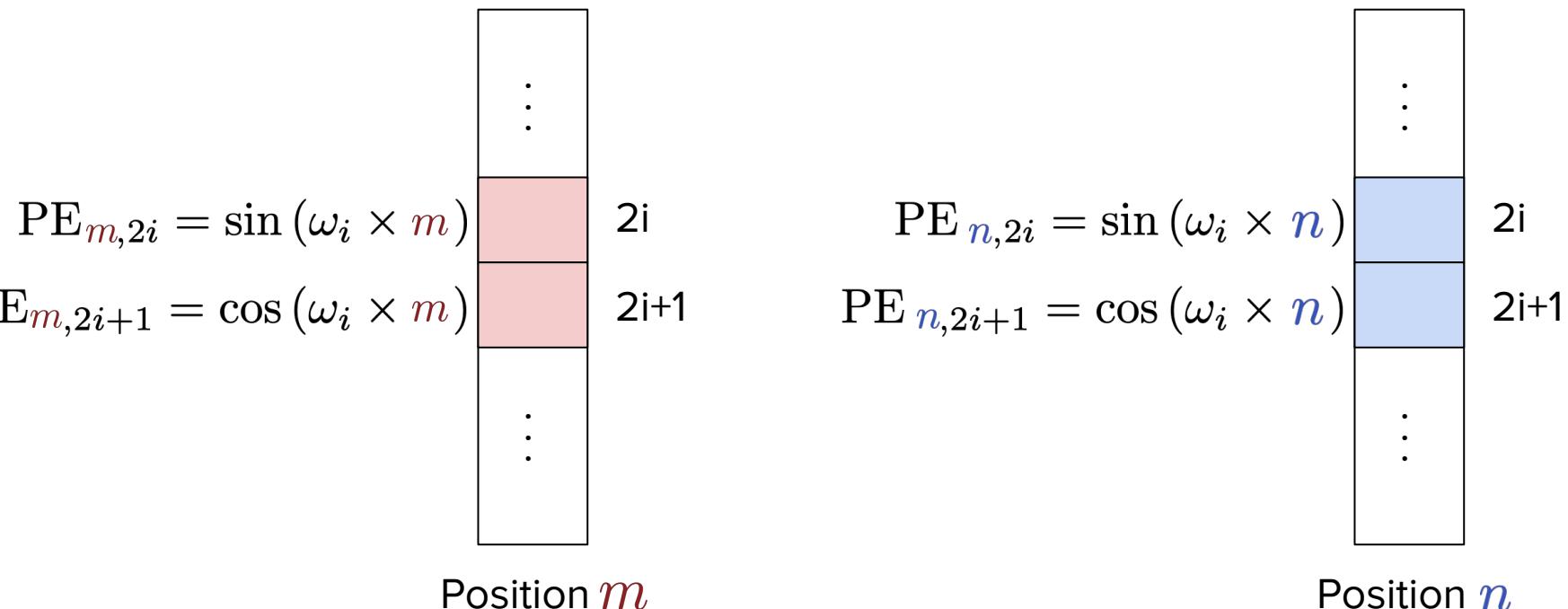
Hardcoded position embeddings

Idea. Hardcode values of position embedding



Hardcoded position embeddings

Idea. Hardcode values of position embedding



Hardcoded position embeddings

Idea. Hardcode values of position embedding

$$\cos(a - b) = \cos(a) \cos(b) + \sin(a) \sin(b)$$

Hardcoded position embeddings

Idea. Hardcode values of position embedding

$$\cos(a - b) = \cos(a) \cos(b) + \sin(a) \sin(b)$$

$$\cos(\omega_i(m - n)) = \cos(\omega_i \times m) \cos(\omega_i \times n) + \sin(\omega_i \times m) \sin(\omega_i \times n)$$

Hardcoded position embeddings

Idea. Hardcode values of position embedding

$$\cos(a - b) = \cos(a) \cos(b) + \sin(a) \sin(b)$$

$$\cos(\omega_i(m - n)) = \cos(\omega_i \times m) \cos(\omega_i \times n) + \sin(\omega_i \times m) \sin(\omega_i \times n)$$

$$\langle \text{PE}_{\textcolor{brown}{m}}, \text{PE}_{\textcolor{blue}{n}} \rangle = \dots + \cos(\omega_i(\textcolor{brown}{m} - \textcolor{blue}{n})) + \dots$$

Hardcoded position embeddings

Idea. Hardcode values of position embedding

$$\cos(a - b) = \cos(a) \cos(b) + \sin(a) \sin(b)$$

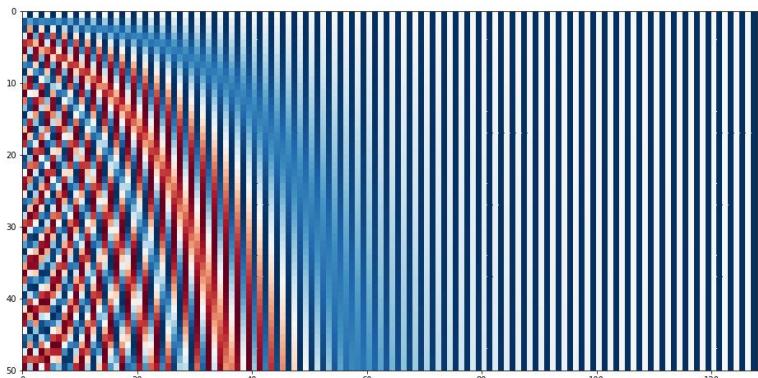
$$\cos(\omega_i(m - n)) = \cos(\omega_i \times m) \cos(\omega_i \times n) + \sin(\omega_i \times m) \sin(\omega_i \times n)$$

$$\langle \text{PE}_{\textcolor{brown}{m}}, \text{PE}_{\textcolor{brown}{n}} \rangle = \dots + \cos(\omega_i(m - n)) + \dots$$

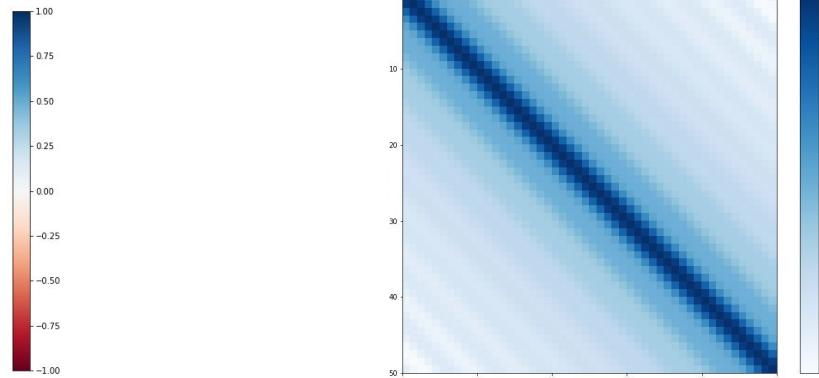
$$\langle \text{PE}_{\textcolor{brown}{m}}, \text{PE}_{\textcolor{blue}{n}} \rangle = f(\textcolor{brown}{m} - \textcolor{blue}{n})$$

Hardcoded position embeddings

Idea. Hardcode values of position embedding



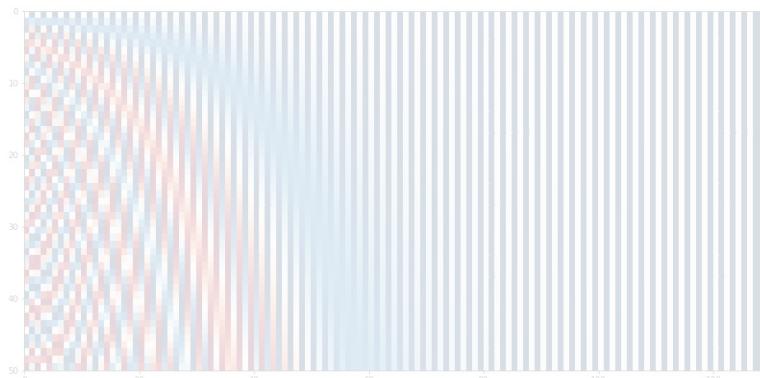
Value of embeddings



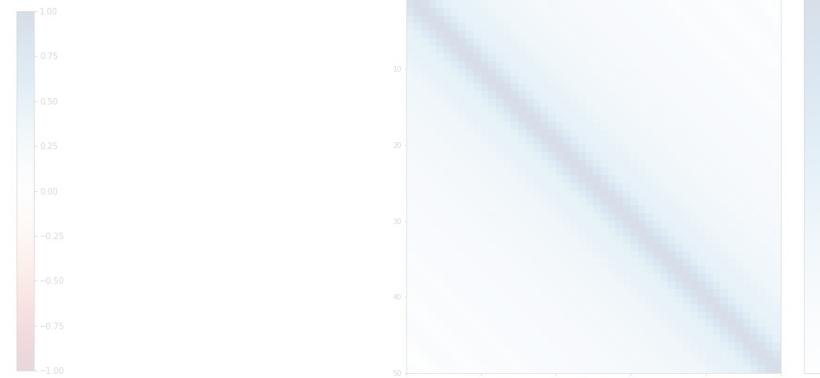
Similarity between positions

Hardcoded position embeddings

Idea. Hardcode values of position embedding



Value of embeddings

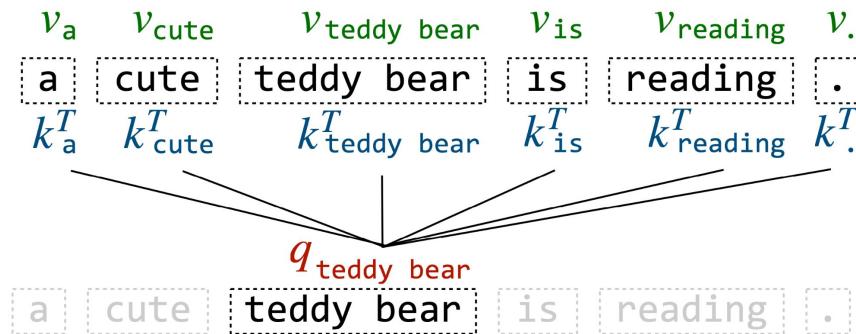


Similarity between positions

Benefits. Can extend to any sequence length

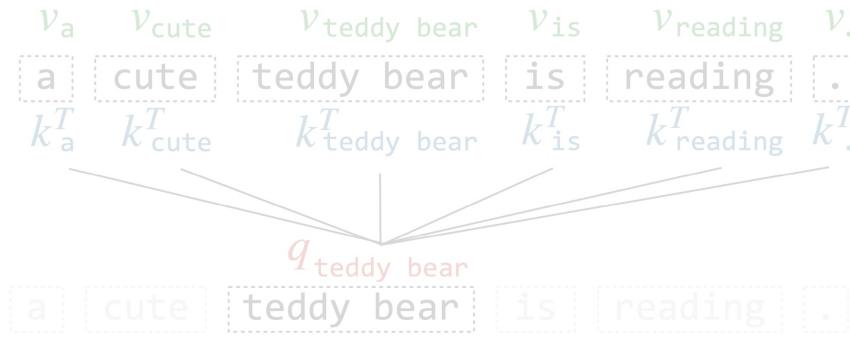
From absolute to relative position info

Motivation. Actually, we really care about relative position **with attention layer**

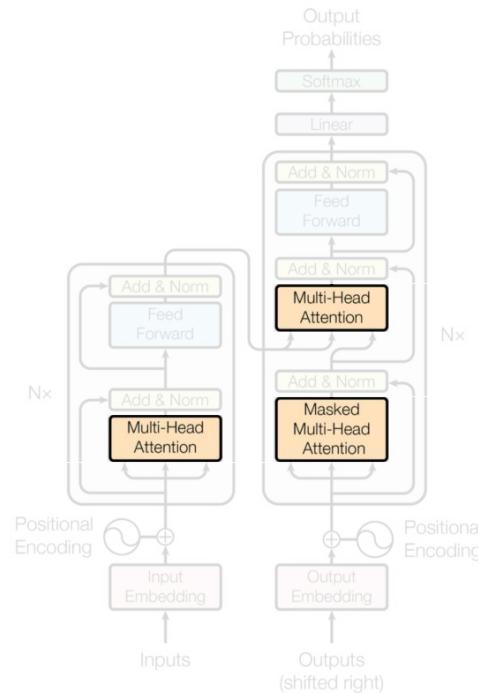


From absolute to relative position info

Motivation. Actually, we really care about relative position **with attention layer**

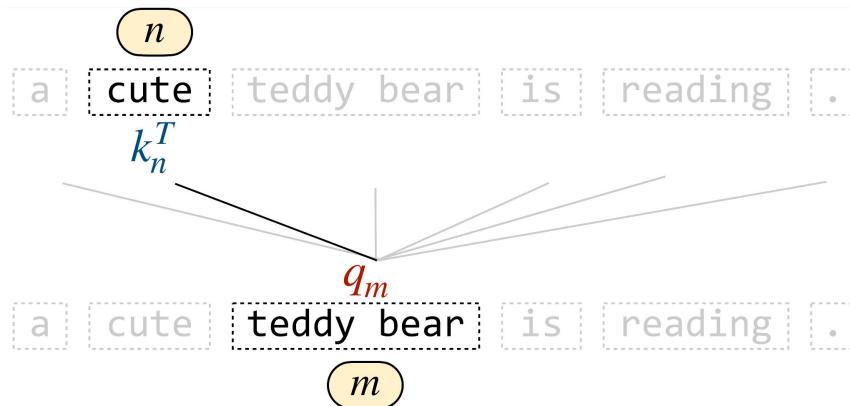


Idea. Let's change the attention layer instead



Linear bias in attention layer

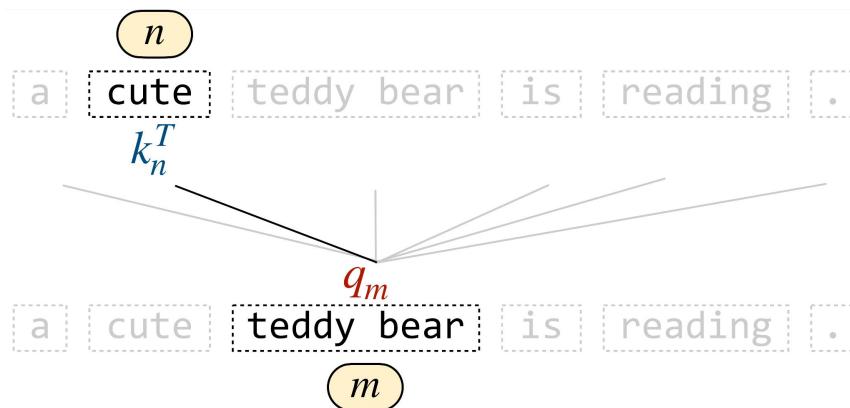
Idea. Add bias to query-key scores



$$\text{softmax} \left(\frac{\langle q_m, k_n \rangle}{\sqrt{d_k}} + \text{bias}(m, n) \right)$$

Linear bias in attention layer

Idea. Add bias to query-key scores



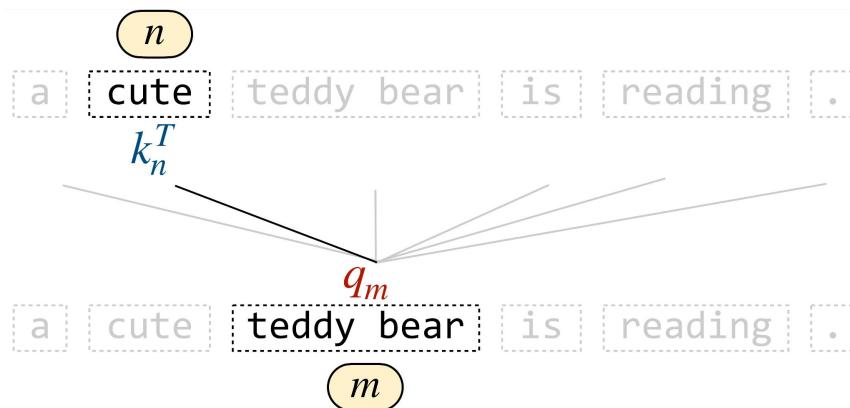
$$\text{softmax} \left(\frac{\langle q_m, k_n \rangle}{\sqrt{d_k}} + \text{bias}(m, n) \right)$$

T5 bias. Bias is **learned per head**

$$\text{bias}(m, n) = \beta_{\text{bucket}(m-n)}$$

Linear bias in attention layer

Idea. Add bias to query-key scores



$$\text{softmax}\left(\frac{\langle q_m, k_n \rangle}{\sqrt{d_k}} + \text{bias}(m, n)\right)$$

T5 bias. Bias is learned per head

$$\text{bias}(m, n) = \beta_{\text{bucket}(m-n)}$$

ALiBi. Bias is **linear, deterministic**, not bounded

$$\text{bias}(m, n) = \mu \times (n - m)$$

Default choice nowadays: rotations in attention layer

Idea. Rotate query and key vectors with rotation matrix



Default choice nowadays: rotations in attention layer

Idea. Rotate query and key vectors with rotation matrix



Formula. Use rotation matrix: $R_{\theta,m} = \begin{pmatrix} \cos(m\theta) & -\sin(m\theta) \\ \sin(m\theta) & \cos(m\theta) \end{pmatrix}$

Default choice nowadays: rotations in attention layer

RoPE = Rotary Position Embeddings

Idea. Rotate query and key vectors with rotation matrix



Formula. Use rotation matrix: $R_{\theta,m} = \begin{pmatrix} \cos(m\theta) & -\sin(m\theta) \\ \sin(m\theta) & \cos(m\theta) \end{pmatrix}$

Default choice nowadays: rotations in attention layer

Extension to dimension $d > 2$. Rotate every block of dimension 2

$$R_{\theta,m} = \begin{pmatrix} \text{block}_1 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \text{block}_{\frac{d_k}{2}} \end{pmatrix} \quad \text{block}_i = \begin{pmatrix} \cos(m\theta_i) & -\sin(m\theta_i) \\ \sin(m\theta_i) & \cos(m\theta_i) \end{pmatrix}$$

Default choice nowadays: rotations in attention layer

Extension to dimension $d > 2$. Rotate every block of dimension 2

$$R_{\theta,m} = \begin{pmatrix} \text{block}_1 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \text{block}_{\frac{d_k}{2}} \end{pmatrix} \quad \text{block}_i = \begin{pmatrix} \cos(m\theta_i) & -\sin(m\theta_i) \\ \sin(m\theta_i) & \cos(m\theta_i) \end{pmatrix}$$

Benefits. Relative distance nicely captured:

$$q_m k_n^T = x_m W_q R_{\theta,n-m} W_k^T x_n^T$$

Default choice nowadays: rotations in attention layer

Extension to dimension $d > 2$. Rotate every block of dimension 2

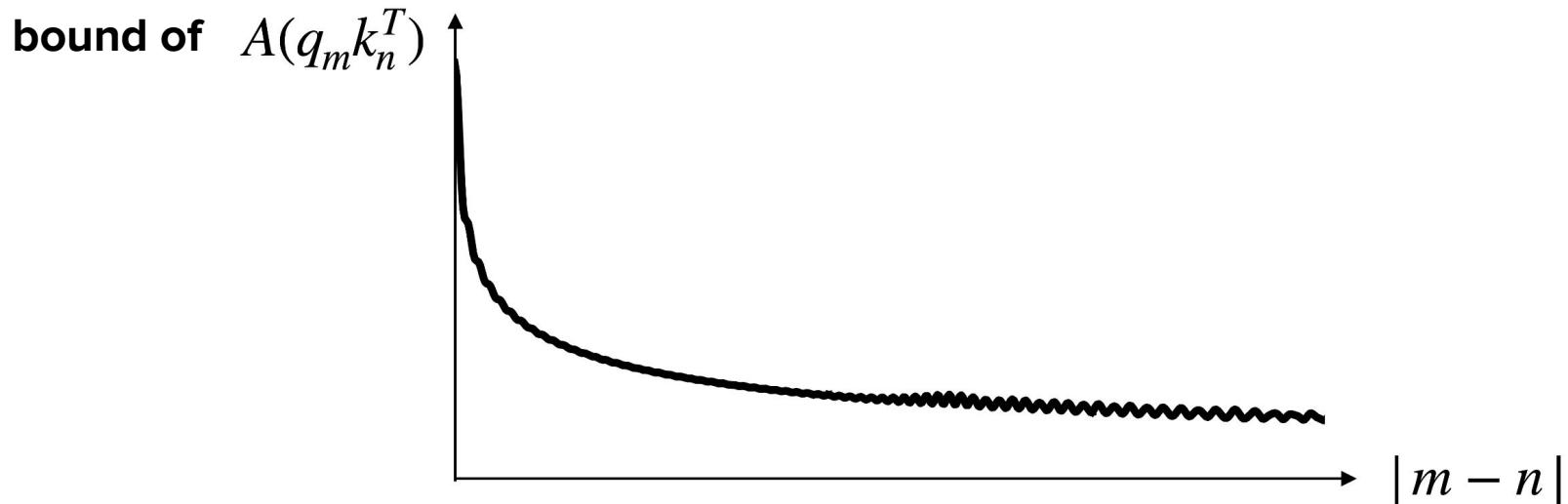
$$R_{\theta,m} = \begin{pmatrix} \text{block}_1 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \text{block}_{\frac{d_k}{2}} \end{pmatrix} \quad \text{block}_i = \begin{pmatrix} \cos(m\theta_i) & -\sin(m\theta_i) \\ \sin(m\theta_i) & \cos(m\theta_i) \end{pmatrix}$$

Benefits. Relative distance nicely captured:

$$q_{\mathbf{m}} k_{\mathbf{n}}^T = x_{\mathbf{m}} W_q R_{\theta,\mathbf{n}-\mathbf{m}} W_k^T x_{\mathbf{n}}^T$$

Attention weight has a long-term decay

Relative upper





Transformers & Large Language Models

Position embeddings

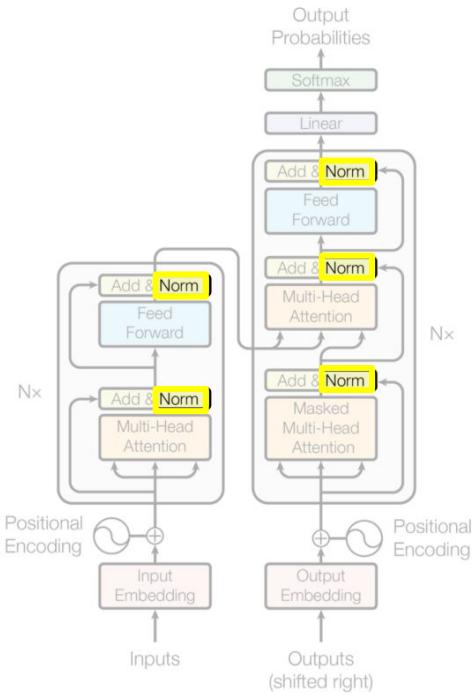
Layer normalization

Attention approximation

Transformer-based models

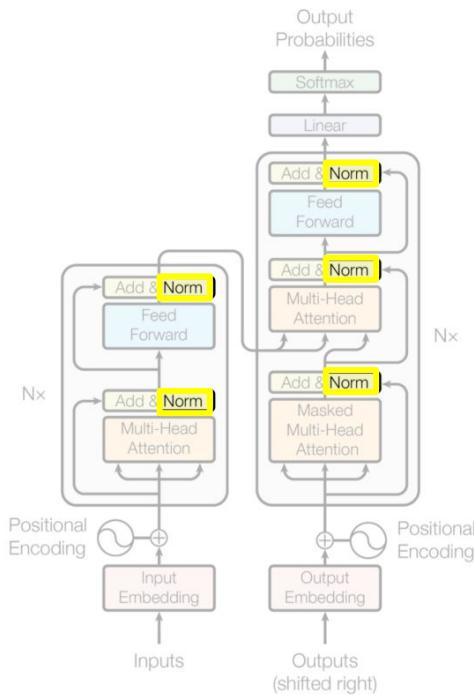
BERT deep dive

Layer normalization



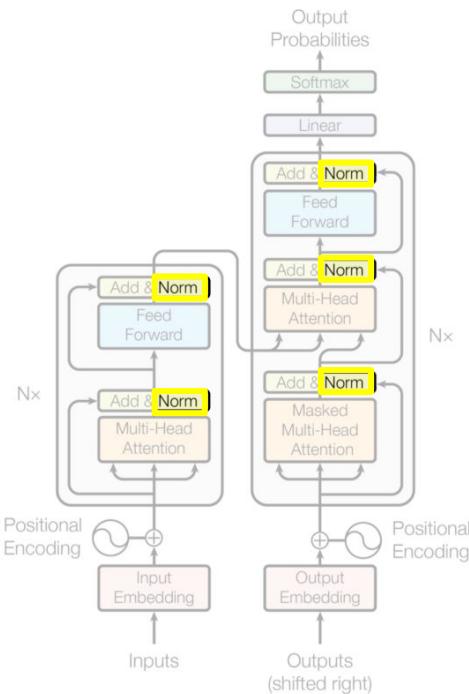
Layer normalization

LN = Layer Normalization



Layer normalization

LN = Layer Normalization



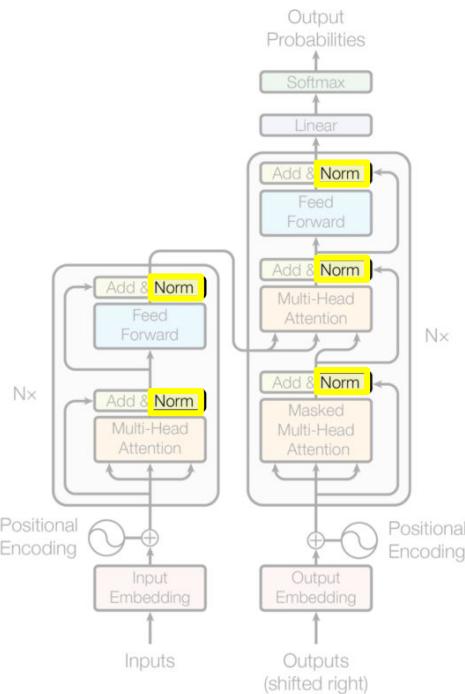
$$\text{LN}(x) = \gamma \hat{x} + \beta$$

$$\text{with } \hat{x} = \frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}}$$

$$\text{where } \mu = \frac{1}{d} \sum_{i=1}^d x_i \quad \text{and} \quad \sigma^2 = \frac{1}{d} \sum_{i=1}^d (x_i - \mu)^2$$

Layer normalization

LN = Layer Normalization



$$\text{LN}(x) = \gamma \hat{x} + \beta$$

$$\text{with } \hat{x} = \frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}}$$

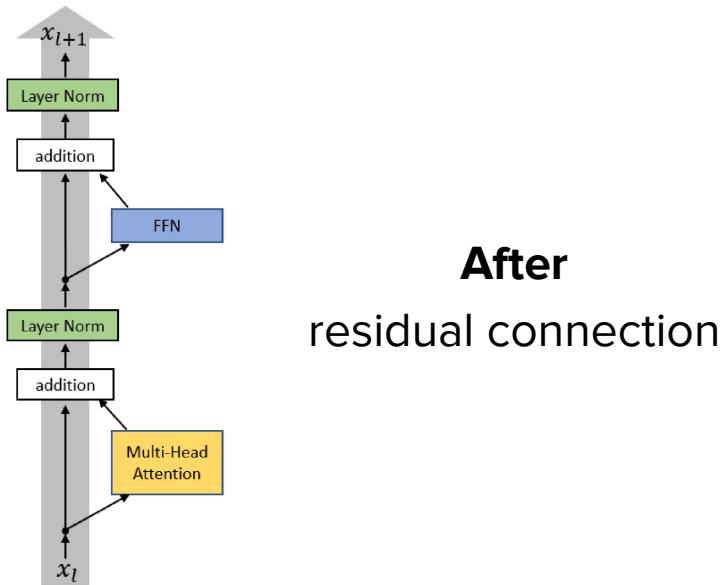
$$\text{where } \mu = \frac{1}{d} \sum_{i=1}^d x_i \quad \text{and} \quad \sigma^2 = \frac{1}{d} \sum_{i=1}^d (x_i - \mu)^2$$

Benefits. Helps with training stability and convergence.

Different kinds of layer normalization

Post-Norm

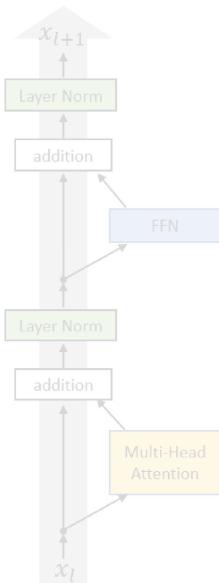
$$\text{Output} = \text{LayerNorm}(x + \text{SubLayer}(x))$$



Different kinds of layer normalization

Post-Norm

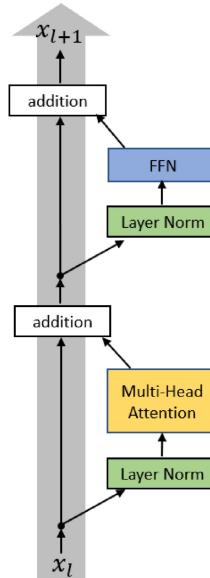
$$\text{Output} = \text{LayerNorm}(x + \text{SubLayer}(x))$$



After
residual connection

Pre-Norm

$$\text{Output} = x + \text{SubLayer}(\text{LayerNorm}(x))$$

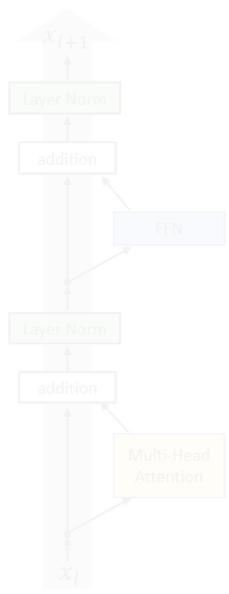


Before
residual connection

Different kinds of layer normalization

Post-LN

$$\text{Output} = \text{LayerNorm}(x + \text{SubLayer}(x))$$



Nowadays:

$$\text{Output} = x + \text{SubLayer}(\text{LayerNorm}(x))$$

Pre-Norm

After
residual connection

Pre-LN

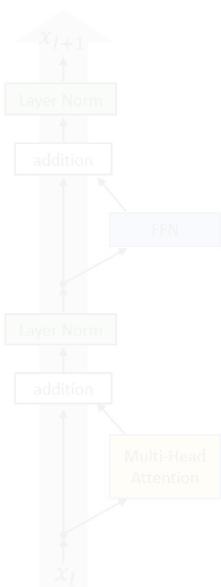
Before
residual connection



Different kinds of layer normalization

Post-LN

$$\text{Output} = \text{LayerNorm}(x + \text{SubLayer}(x))$$



Pre-LN

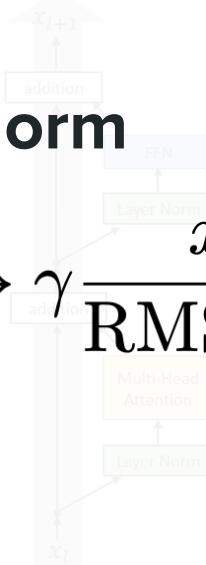
$$\text{Output} = x + \text{SubLayer}(\text{LayerNorm}(x))$$

Nowadays:

Pre-Norm +

RMSNorm

$$\text{After residual connection: } \gamma \hat{x} + \beta \longrightarrow \gamma \frac{x}{\text{RMS}(x)}$$





Transformers & Large Language Models

Position embeddings

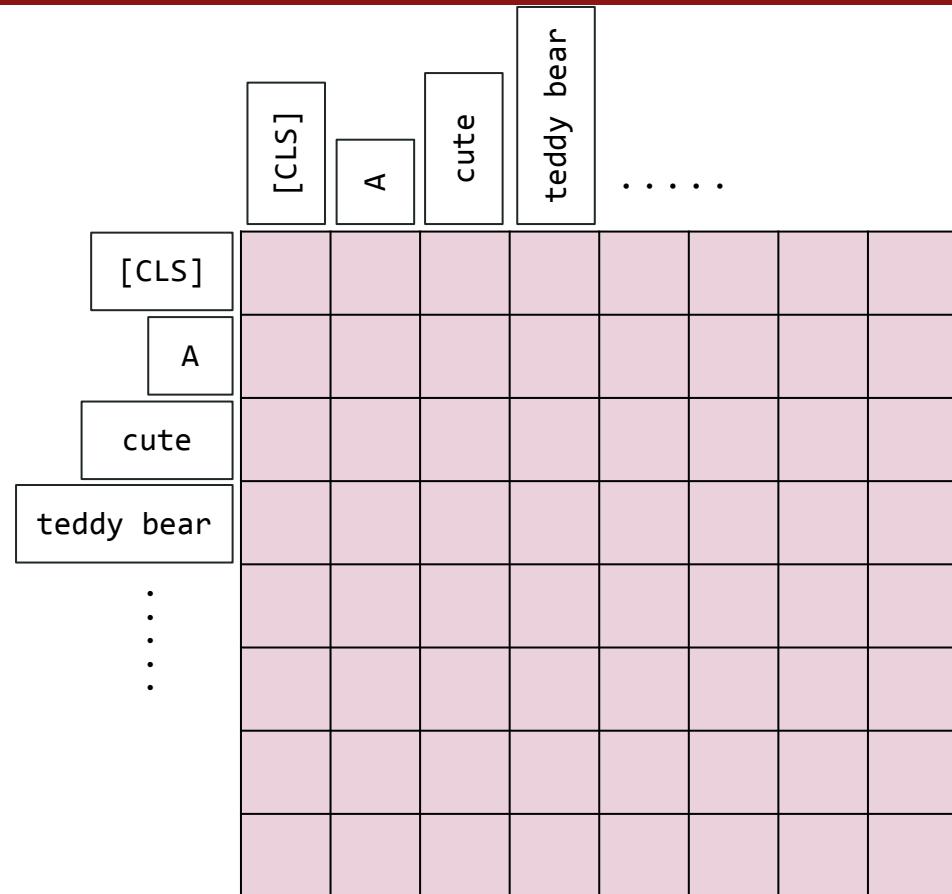
Layer normalization

Attention approximation

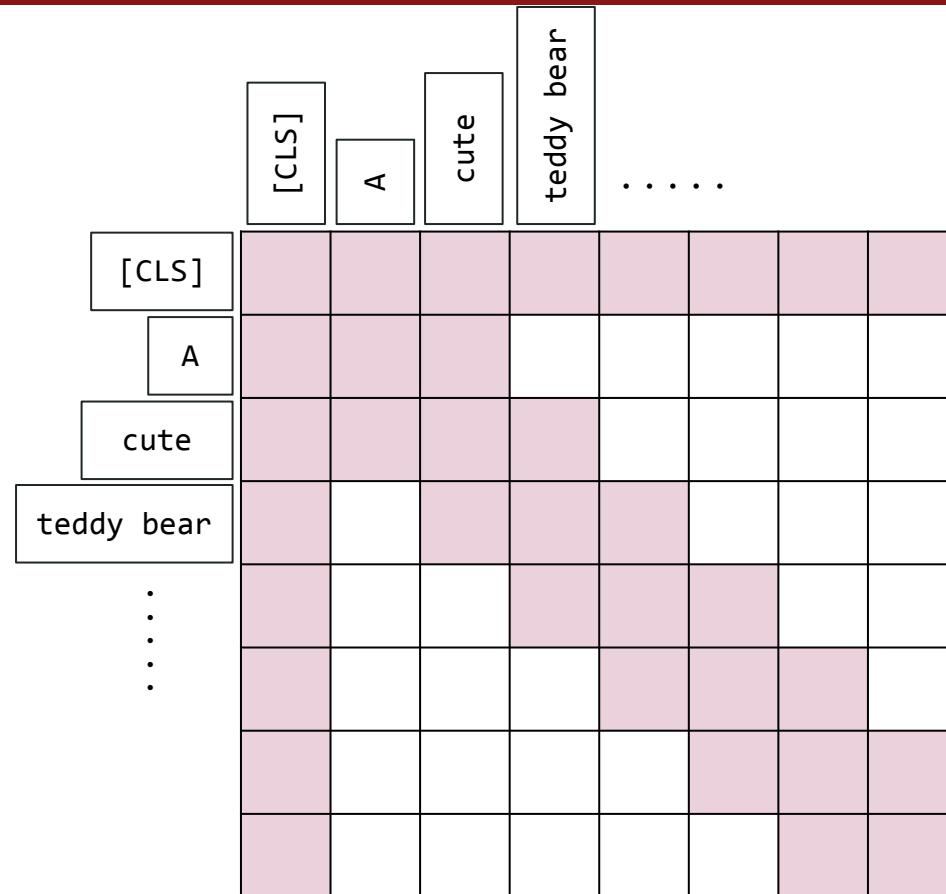
Transformer-based models

BERT deep dive

Sparse attention: Longformer

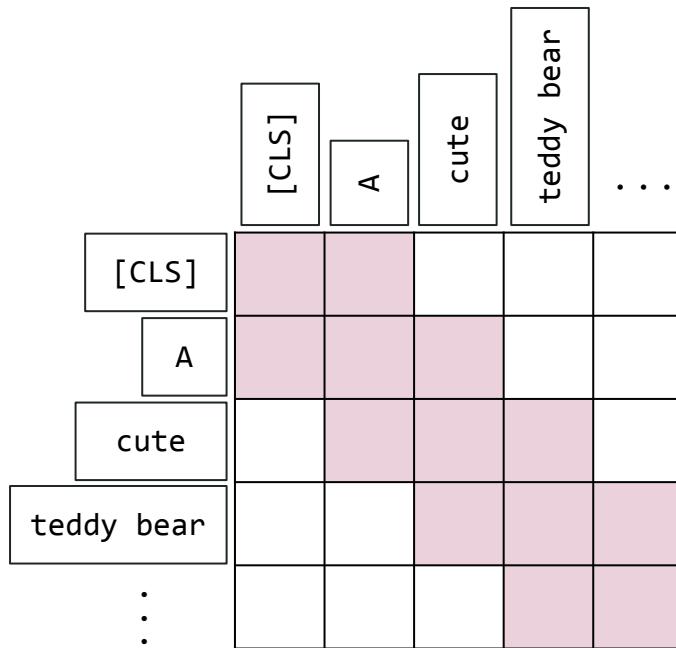


Sparse attention: Longformer



Leveraging local and global attention

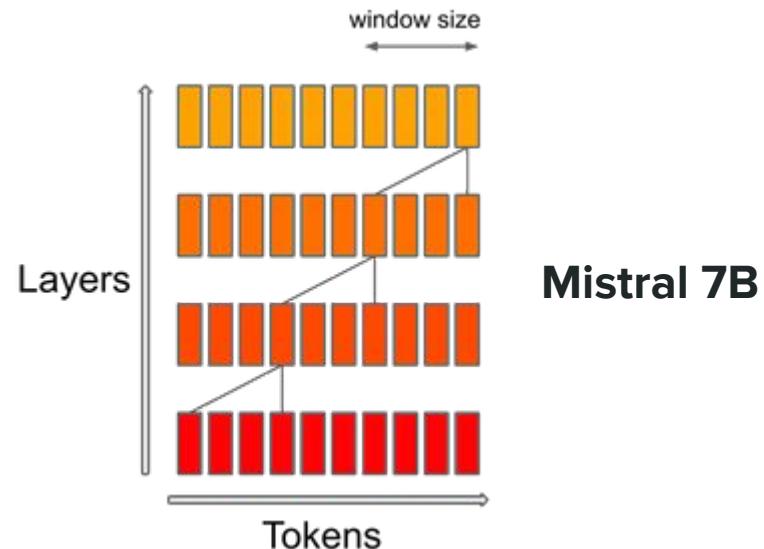
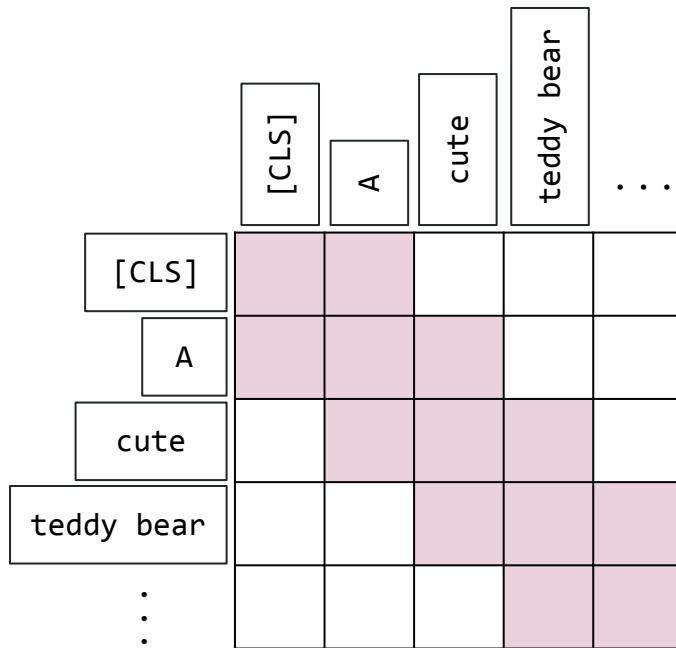
SWA = Sliding Window Attention



Variations include **interleaving local and global attention layers**

Leveraging local and global attention

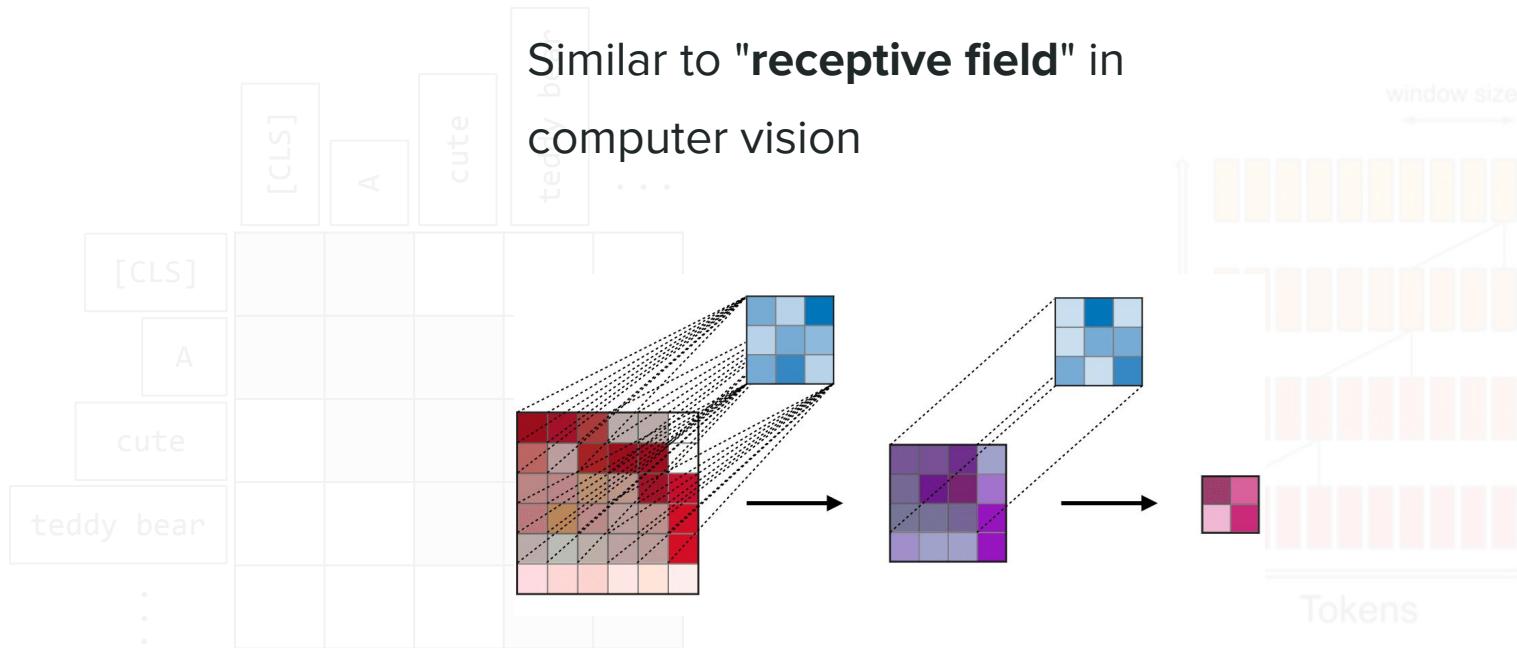
SWA = Sliding Window Attention



Leveraging local and global attention

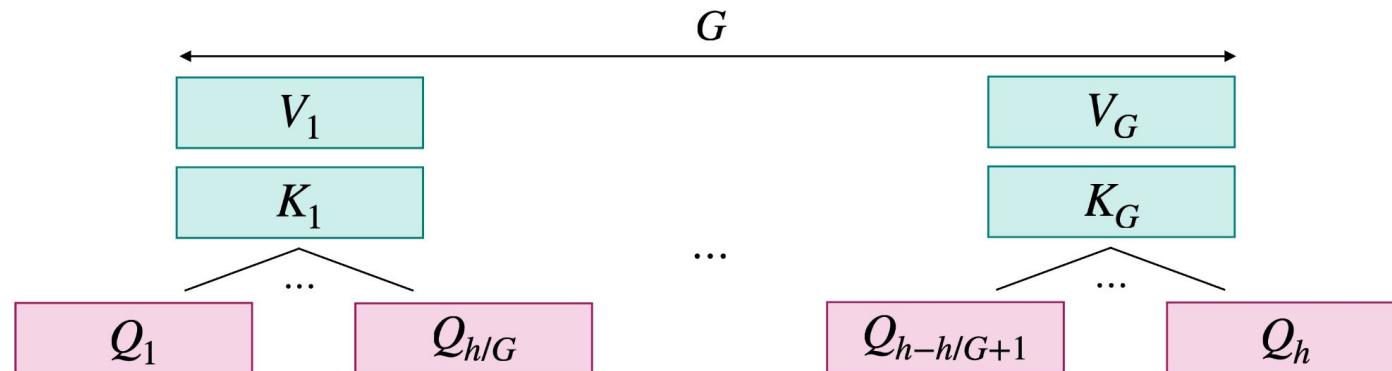
SWA = Sliding Window Attention

Similar to "**receptive field**" in
computer vision



Sharing attention heads

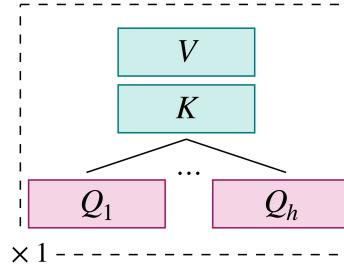
Idea. Share key/value attention heads within groups of queries



Sharing attention heads

$$G = 1$$

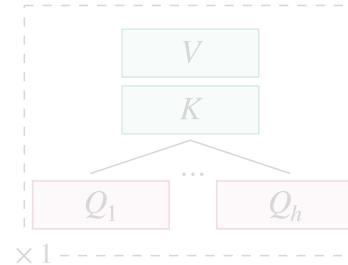
**Multi-Query Attention
(MQA)**



Sharing attention heads

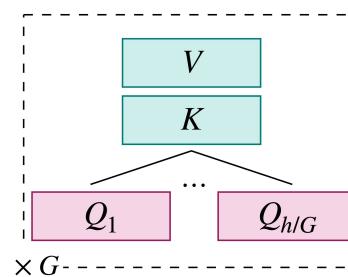
$G = 1$

**Multi-Query Attention
(MQA)**

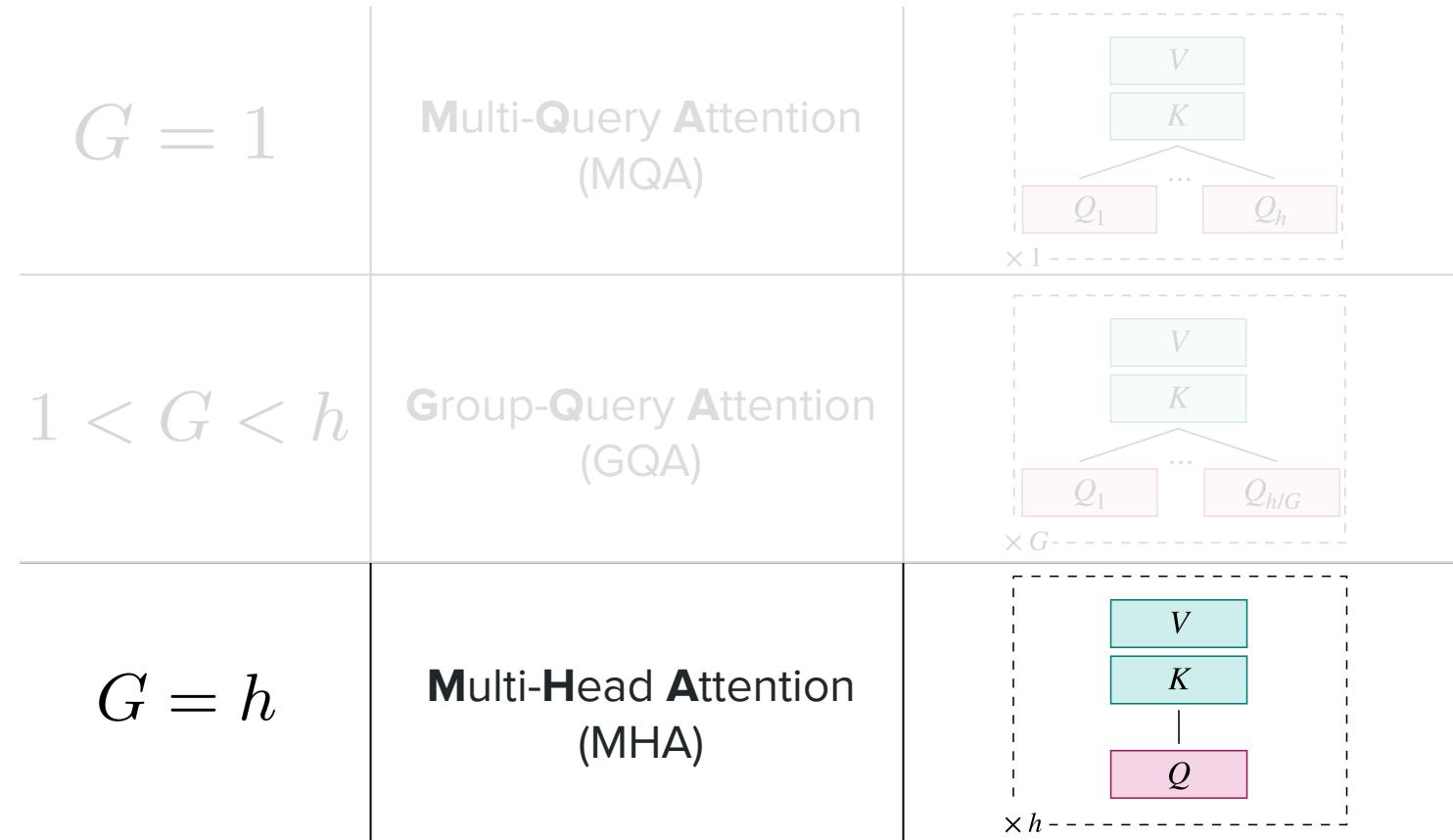


$1 < G < h$

**Group-Query Attention
(GQA)**



Sharing attention heads





Transformers & Large Language Models

Attention approximation

Position embeddings

Layer normalization

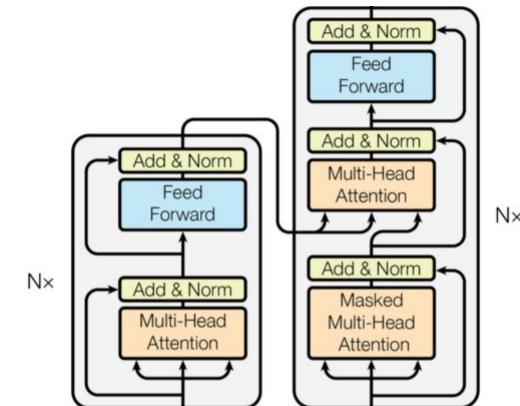
Transformer-based models

BERT deep dive

Overview of Transformer-based models

3 categories of Transformer-based models:

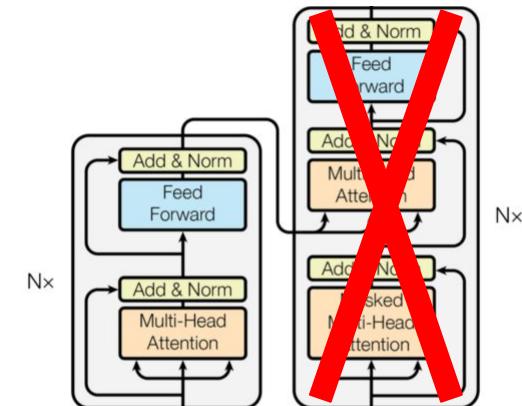
Encoder-decoder	Text to text	T5, mT5, ByT5
------------------------	--------------	---------------



Overview of Transformer-based models

3 categories of Transformer-based models:

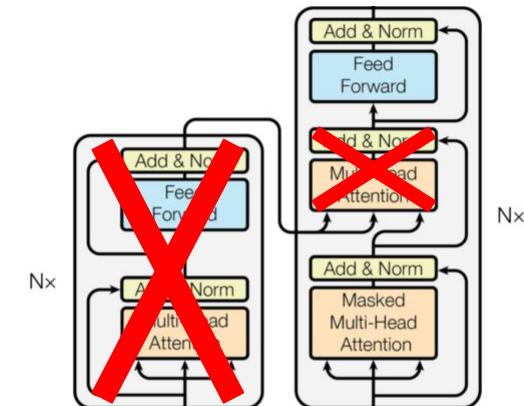
Encoder-decoder	Text to text	T5, mT5, ByT5
Encoder-only	Projection of embedding for class prediction (e.g. sentiment extraction)	BERT, DistilBERT, RoBERTa



Overview of Transformer-based models

3 categories of Transformer-based models:

Encoder-decoder	Text to text	T5, mT5, ByT5
Encoder-only	Projection of embedding for class prediction (e.g. sentiment extraction)	BERT, DistilBERT, RoBERTa
Decoder-only	Text to text	GPT series



Overview of Transformer-based models

3 categories of Transformer-based models:

Encoder-decoder	Text to text	T5, mT5, ByT5
Encoder-only	Projection of embedding for class prediction (e.g. sentiment extraction)	BERT, DistilBERT, RoBERTa
Decoder-only	Text to text	GPT series

Popular in ~2018-2022

```
graph LR; A[Popular in ~2018-2022] --> B[Encoder-decoder]; A --> C[Encoder-only]
```

Overview of Transformer-based models

3 categories of Transformer-based models:

Encoder-decoder	Text to text	T5, mT5, ByT5
Encoder-only	Projection of embedding for class prediction (e.g. sentiment extraction)	BERT, DistilBERT, RoBERTa
Decoder-only	Text to text	GPT series

← **Popular now!**



Transformers & Large Language Models

Attention approximation

Position embeddings

Layer normalization

Transformer-based models

BERT deep dive

Why "BERT"?

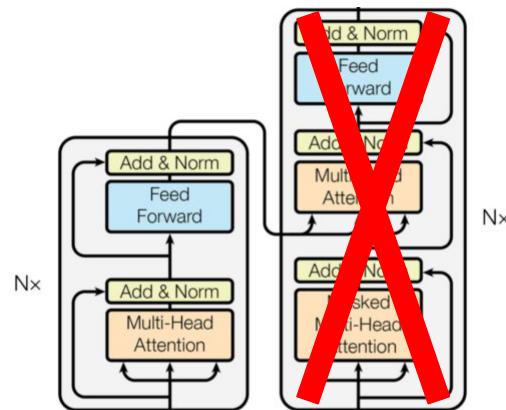
BERT = Bidirectional Encoder Representations from Transformers

Why "BERT"?

BERT = Bidirectional Encoder Representations from Transformers

Why "BERT"?

BERT = Bidirectional Encoder Representations from Transformers

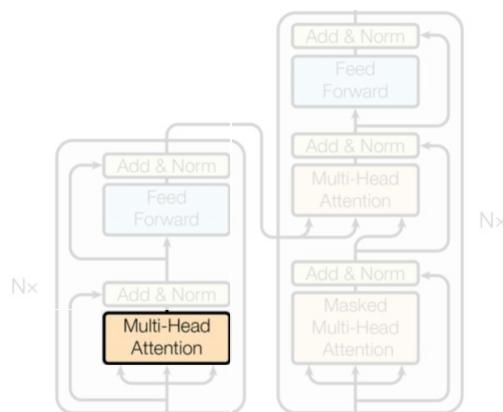


Why "BERT"?

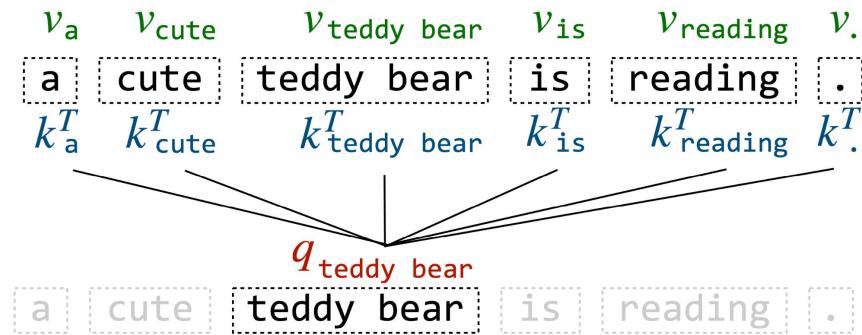
BERT = Bidirectional Encoder Representations from Transformers

Why "BERT"?

BERT = Bidirectional Encoder Representations from Transformers

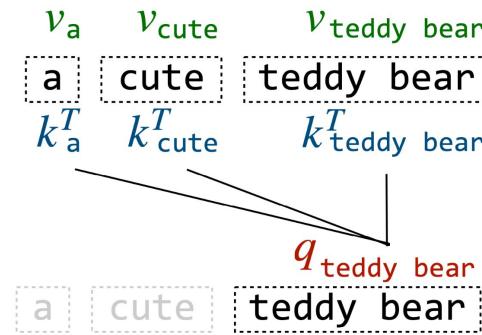
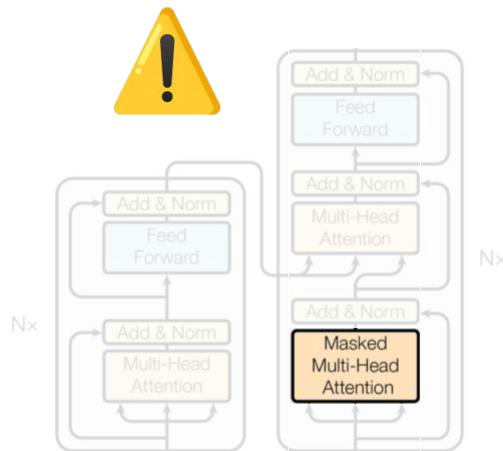


Bidirectional



Why "BERT"?

BERT = Bidirectional Encoder Representations from Transformers



Not bidirectional

Why "BERT"?

Deep contextualized word representations

Matthew E. Peters[†], Mark Neumann[†], Mohit Iyyer[†], Matt Gardner[†],
`{matthewp, markn, mohiti, mattg}@allenai.org`

Christopher Clark*, Kenton Lee*, Luke Zettlemoyer^{†*}
`{csquared, kentonl, lsz}@cs.washington.edu`

[†]Allen Institute for Artificial Intelligence

*Paul G. Allen School of Computer Science & Engineering, University of Washington

Abstract

We introduce a new type of *deep contextualized* word representation that models both (1) complex characteristics of word use (e.g., syn-

guage model (LM) objective on a large text corpus. For this reason, we call them ELMo (Embeddings from Language Models) representations. Unlike previous approaches for learning contextu-

ELMo paper
[submitted in Feb 2018]

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

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

Abstract

We introduce a new language representation model called **BERT**, which stands for **Bidirectional Encoder Representations from Transformers**. Unlike recent language representation models (Peters et al. 2018a; Rad-

There are two existing strategies for applying pre-trained language representations to downstream tasks: *feature-based* and *fine-tuning*. The feature-based approach, such as ELMo (Peters et al., 2018a), uses task-specific architectures that include the pre-trained representations as addi-

BERT paper
[submitted in Oct 2018]

Why "BERT"?



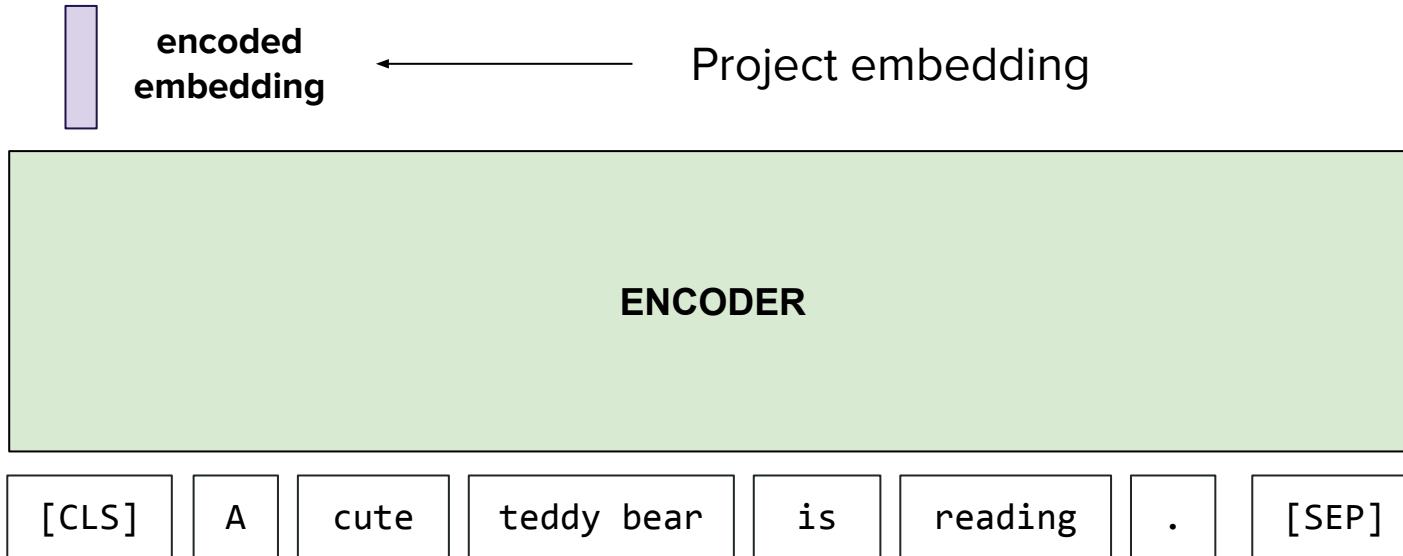
Elmo



Bert

BERT and encoder-only models

Idea. Keep only **encoders** + leverage **encoded embeddings** to make predictions.



BERT and encoder-only models

Strategy.

- **Step 1:** Pretraining with proxy tasks (MLM and NSP)
- **Step 2:** Finetuning for given end task

BERT and encoder-only models

Strategy.

- **Step 1:** Pretraining with proxy tasks (MLM and NSP)
- **Step 2:** Finetuning for given end task

Discussion.

Pros	Cons
<ul style="list-style-type: none">● Finetuning does not need a lot of data● Good performance	<ul style="list-style-type: none">● Not suited for a range of tasks (e.g. text generation)● Finetuning is a required step

BERT and encoder-only models

Strategy.

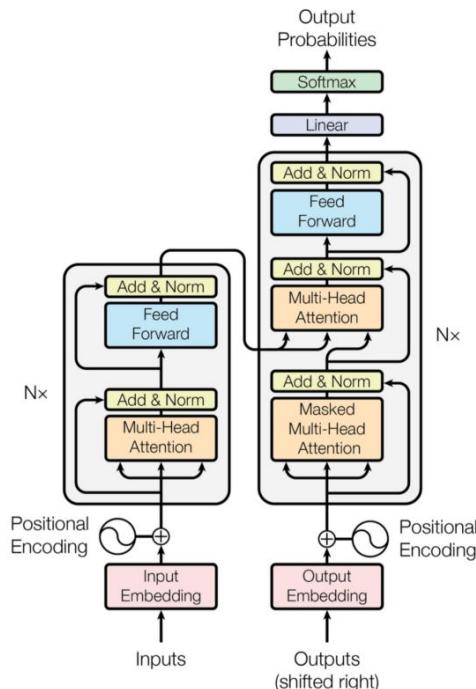
- **Step 1:** Pretraining with proxy tasks (MLM and NSP)
- **Step 2:** Finetuning for given end task

Discussion.

Pros	Cons
<ul style="list-style-type: none">● Finetuning does not need a lot of data● Good performance	<ul style="list-style-type: none">● Not suited for a range of tasks (e.g. text generation)● Finetuning is a required step

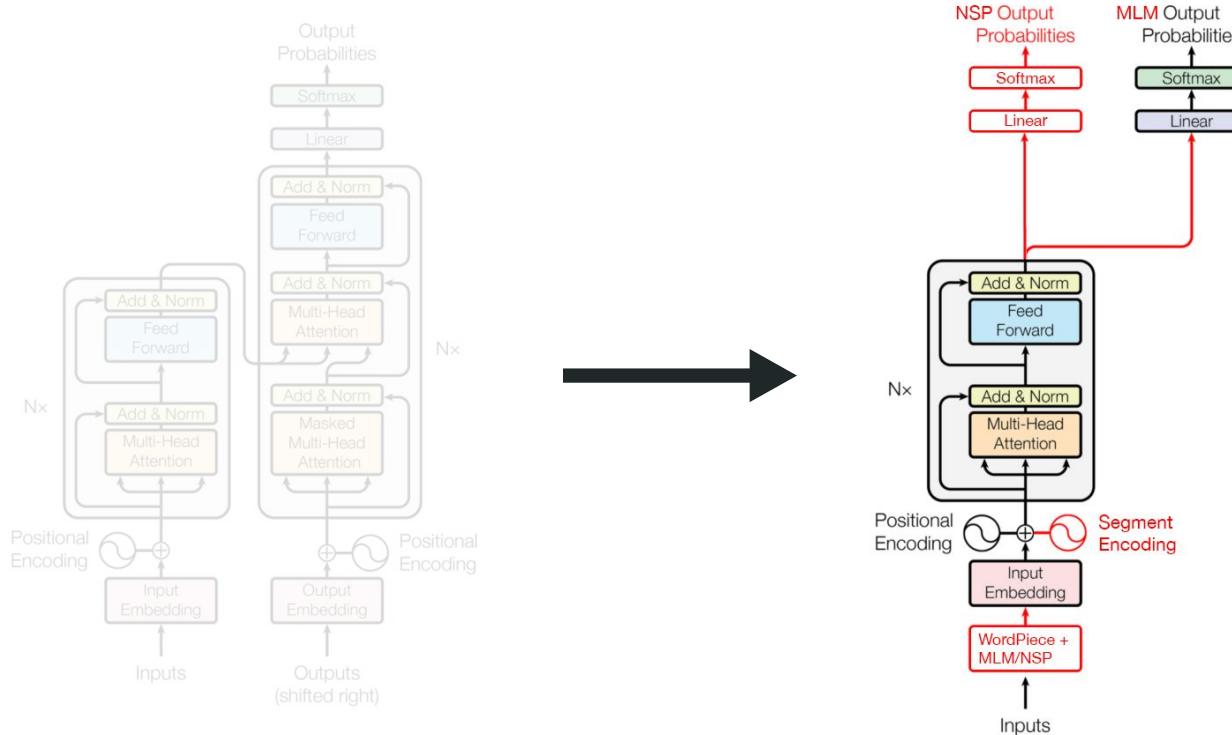
Variants. RoBERTa, DistilBERT, ALBERT

BERT architecture



Original transformer (2017)

BERT architecture

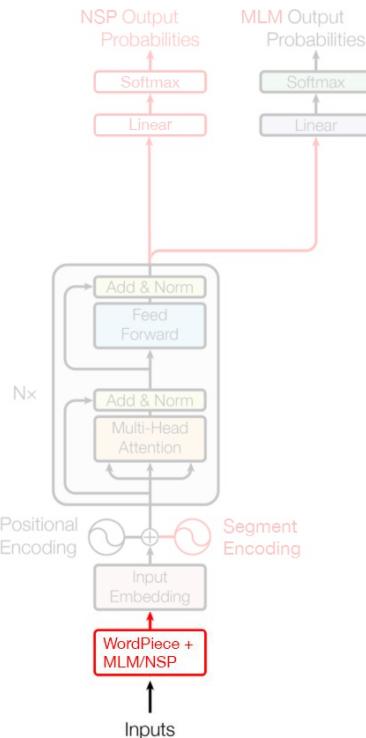


Original transformer (2017)

BERT (2018)

Figure adapted from “*Attention Is All You Need*”, Vaswani et al., 2017.

Input processing



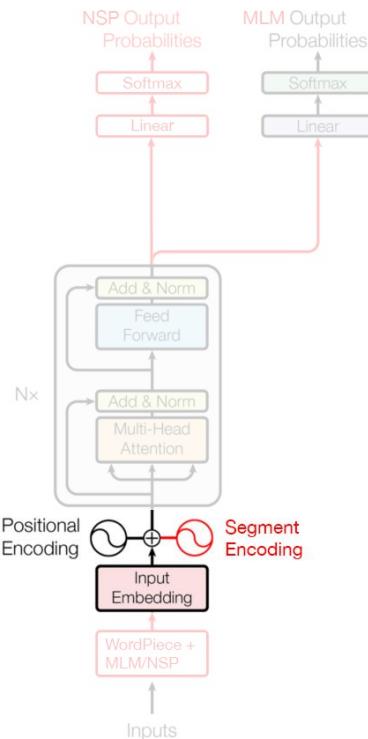
WordPiece algorithm

- Tokenizer trained on a training set beforehand
- Vocabulary size: ~30,000
- Great at detecting common particles

NSP / MLM task processing

- Add [CLS] token at the beginning of the input
- Separate consecutive segments with the [SEP] token and put another one at the end
- Use [MASK] to mask inputs

Input embedding



Input embeddings

- Gigantic lookup table
- Learns an embedding for each word of the vocabulary

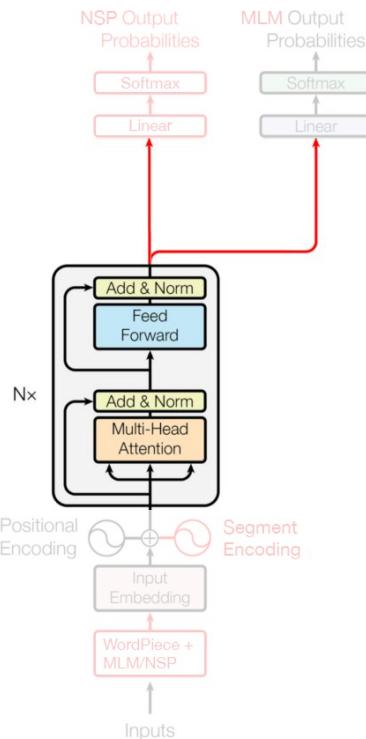
Positional encoding

- Helps the network associate tokens with a position
- Encoding either learnt or fixed with cosines and sines

(new!) Segment encoding

- Shared embedding for a segment

Encoder-only model



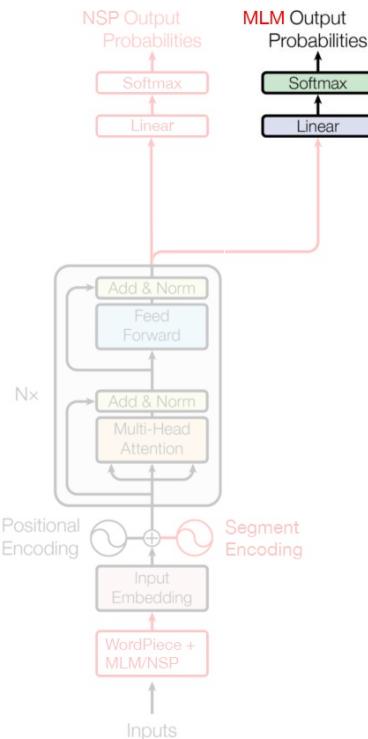
Model

Encoder part of the original transformers paper

Goal

- Represent input data with features (hopefully) needed for NLP tasks
- Leverage the Transformer's self-attention mechanism
- Use learned embedding towards classification-oriented tasks

Proxy task: Masked Language Modeling



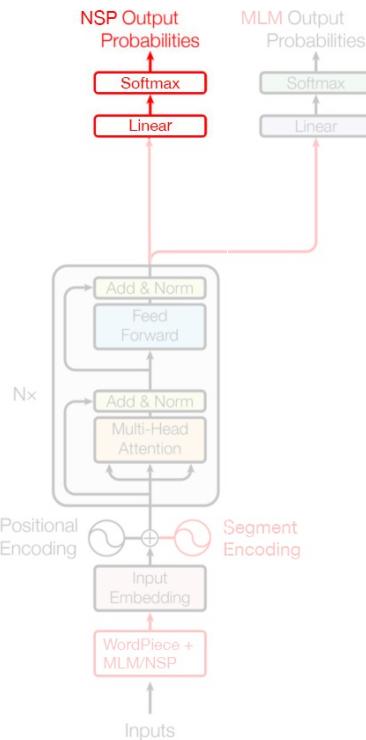
Idea. 15% of input tokens are set up for prediction where:

- 80% are masked
- 10% are changed to a random word
- 10% are unchanged

Benefits

- Network learns language modeling based on contextual information
- Regularization reflects probabilistic nature of language

Proxy task: Next Sentence Prediction



Idea. Pick two sentences from the corpus, where:

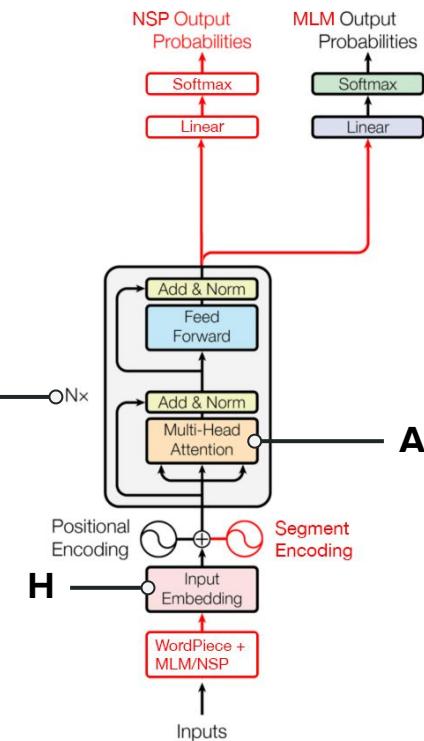
- 50% of the time, they follow each other
- 50% of the time, they **do not** follow each other

Task. Predict if they actually follow each other.

Benefits

- Network implicitly learns to detect useful contextual information
- Easy classification task that does not require any labels

Hyperparameters



Model.

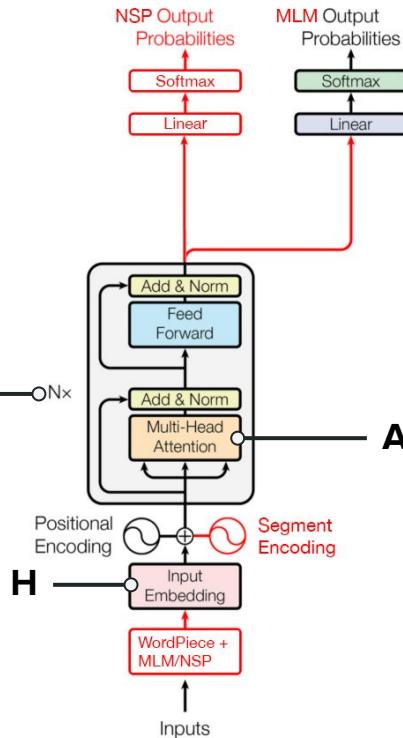
- **L** Layers.
- **H** Hidden layer size aka dimension of embeddings.
- **A** Attention heads operating in parallel.

Data.

- Language-specific / multilingual. Languages on which the model has been trained on.
- Cased / uncased. Whether inputs are converted to lowercase or not.

Some numbers

ICML



	L	H	A	Parameters
BERT-Tiny	2	128	2	4M
BERT-Mini	4	256	4	11M
BERT-Small	4	512	8	30M
BERT-Medium	8	512	8	42M
BERT-Base	12	768	12	110M
BERT-Large	24	1024	16	340M

Overview of BERT finetuning

Goal. Leverage embeddings learned by BERT for a “sister” task

Overview of BERT finetuning

Goal. Leverage embeddings learned by BERT for a “sister” task

Tricks

- Use weights from **already massively pretrained** model
- **Freezing** early layers: sometimes better trade-off complexity/performance
- Great results possible with **minimal labeled data** (depending on complexity/proximity to pretrained data distribution + objectives)

Overview of BERT finetuning

Goal. Leverage embeddings learned by BERT for a “sister” task

Tricks

- Use weights from **already massively pretrained** model
- **Freezing** early layers: sometimes better trade-off complexity/performance
- Great results possible with **minimal labeled data** (depending on complexity/proximity to pretrained data distribution + objectives)

Use cases

- Sequence classification: e.g. sentiment extraction
- Token classification: e.g. question answering

Finetuning: sentiment extraction

This teddy bear is SO CUTE!

Finetuning: sentiment extraction

this teddy bear is so cute!

Finetuning: sentiment extraction

this teddy bear is so cute !

Finetuning: sentiment extraction

Add [CLS] token as a placeholder for sentiment.

[CLS] this teddy bear is so cute !

Finetuning: sentiment extraction

[CLS] this teddy bear is so cute ! [SEP] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD]

Finetuning: sentiment extraction



embedding



Finetuning: sentiment extraction



position embedding



embedding

[CLS]

0

this

1

teddy

2

bear

3

is

4

so

5

cute

6

!

7

[SEP]

8

[PAD]

9

[PAD]

10

[PAD]

11

[PAD]

12

[PAD]

13

[PAD]

14

Finetuning: sentiment extraction



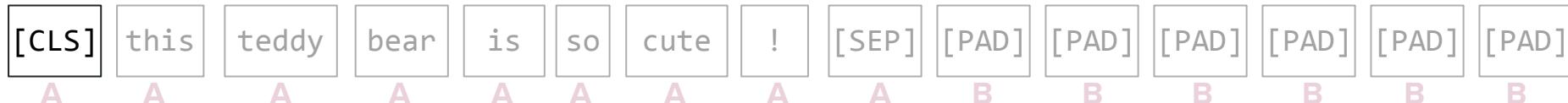
segment embedding



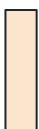
position embedding



embedding



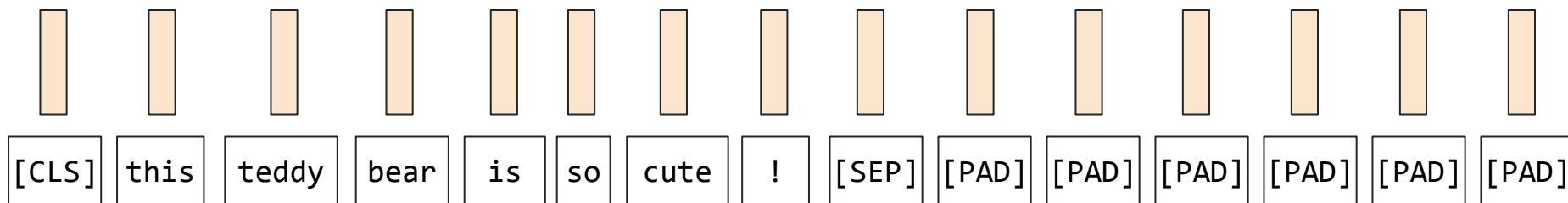
Finetuning: sentiment extraction



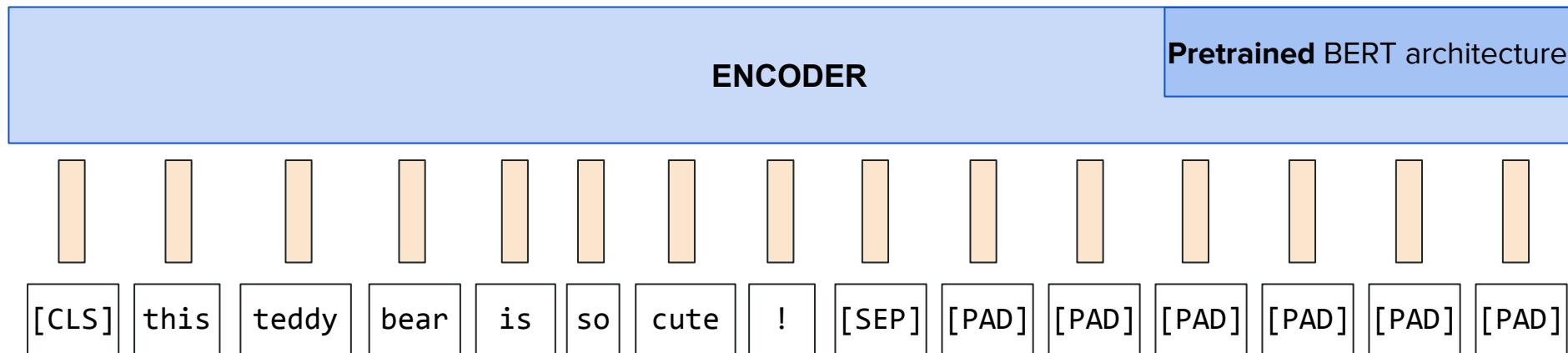
position- and segment-aware embedding



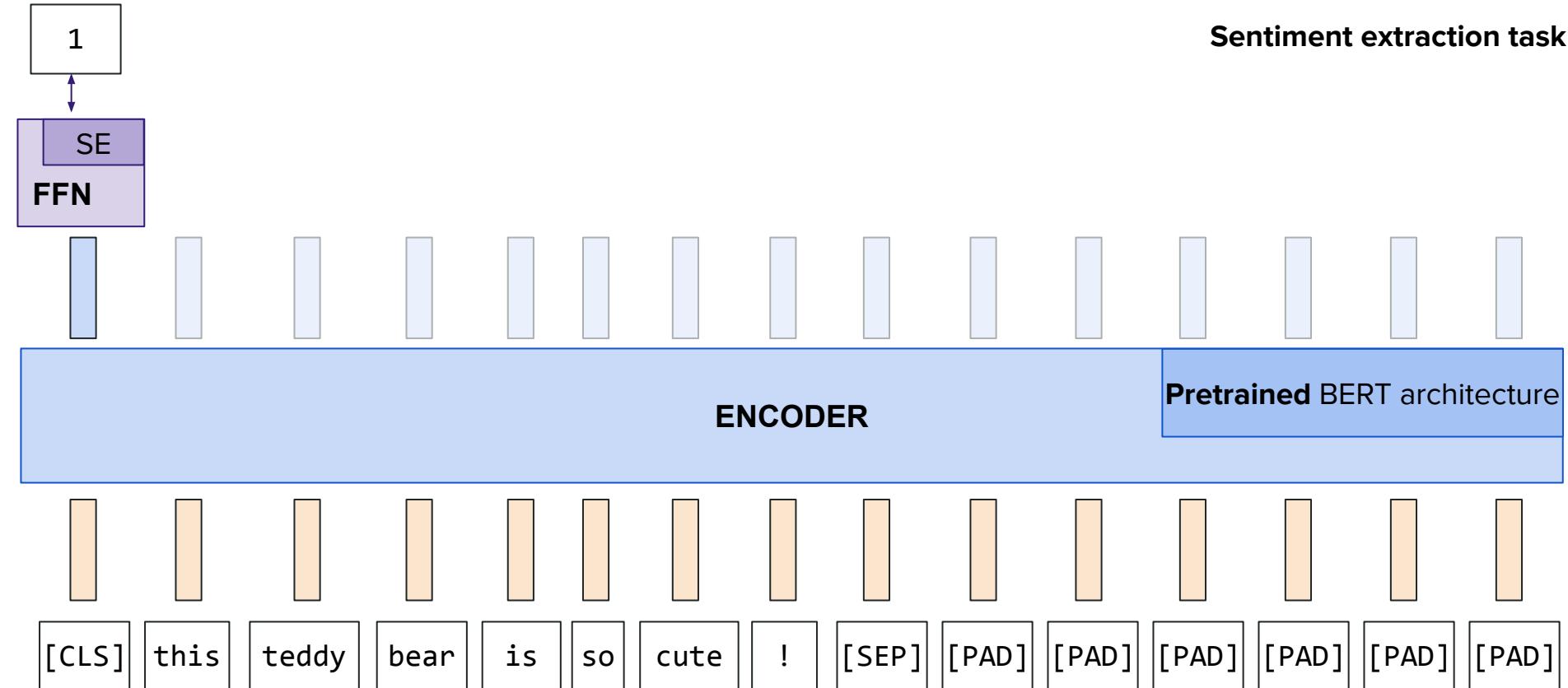
Finetuning: sentiment extraction



Finetuning: sentiment extraction



Finetuning: sentiment extraction



Takeaways and shortcomings

Benefits.

- State of the art results
- ~True contextual representation of words
- Adaptable to many classification tasks

Takeaways and shortcomings

Benefits.

- State of the art results
- ~True contextual representation of words
- Adaptable to many classification tasks

Applications. Widely used in the industry for anything related to encoding

Takeaways and shortcomings

Benefits.

- State of the art results
- ~True contextual representation of words
- Adaptable to many classification tasks

Applications. Widely used in the industry for anything related to encoding

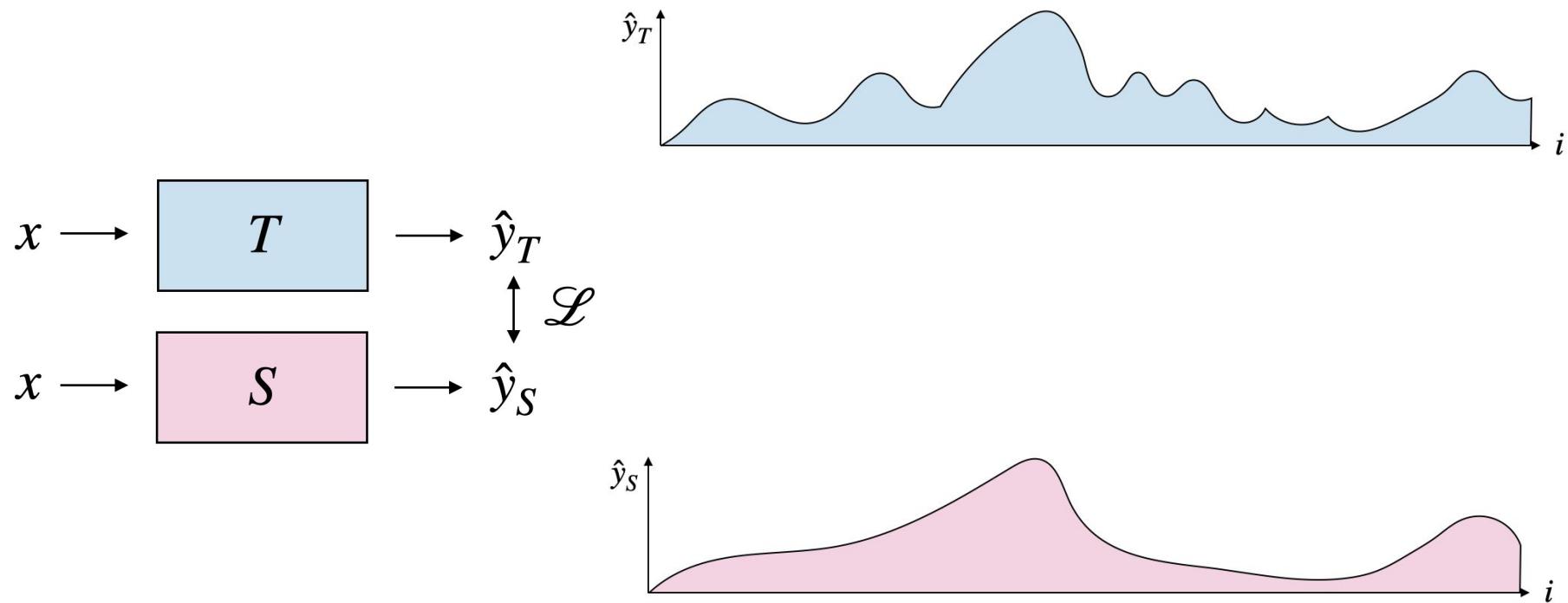
Limitations.

- Context window size is limited
- Computationally expensive: hard sell for low-latency/cost-sensitive applications
- Training paradigm is complex: MLM/NSP + finetuning

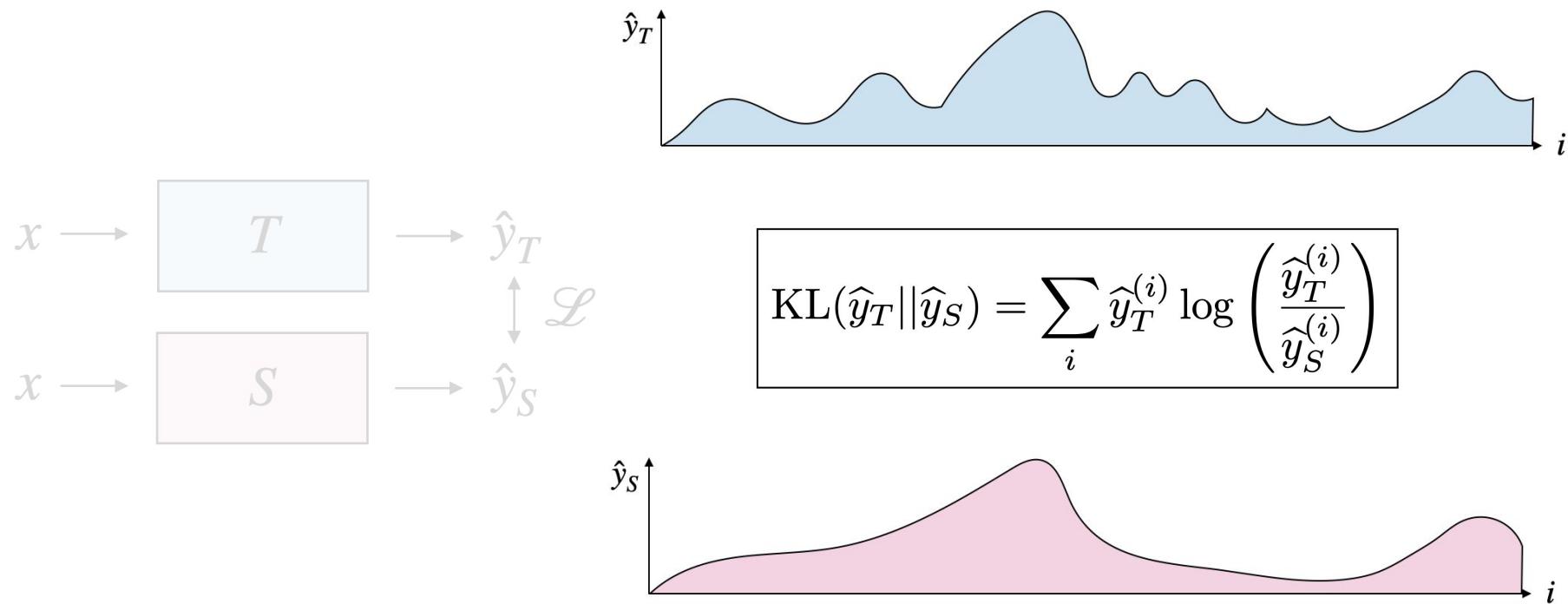
Trick: distillation

"The **soft targets** contain almost all the **knowledge**."

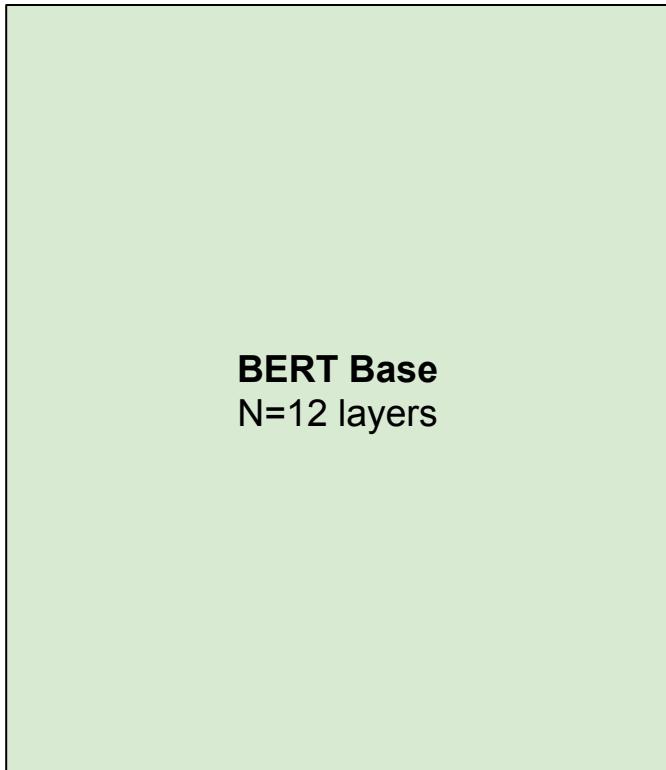
Trick: distillation



Trick: distillation



Variant for efficiency: DistilBERT



~1.6x faster



~97% performance

Variant for performance: RoBERTa

Goal. Thorough analysis of directions of optimization

Variant for performance: RoBERTa

Goal. Thorough analysis of directions of optimization

Modeling

- Removing NSP/segment encodings: ~no effect!*
- Static → dynamic masking across epochs

*DistilBERT had also removed it

Variant for performance: RoBERTa

Goal. Thorough analysis of directions of optimization

Modeling

- Removing NSP/segment encodings: ~no effect!
- Static → dynamic masking across epochs

Data

- Richer: 16 GB → 160 GB of pretraining corpus size
- For longer: 1M steps of batch size 256 vs. 500k of batch size 8k

Variant for performance: RoBERTa

Goal. Thorough analysis of directions of optimization

Modeling

- Removing NSP/segment encodings: ~no effect!
- Static → dynamic masking across epochs

Data

- Richer: 16 GB → 160 GB of pretraining corpus size
- For longer: 1M steps of batch size 256 vs. 500k of batch size 8k

Result. +4% across benchmarks with same architecture

Thank you for your attention!
