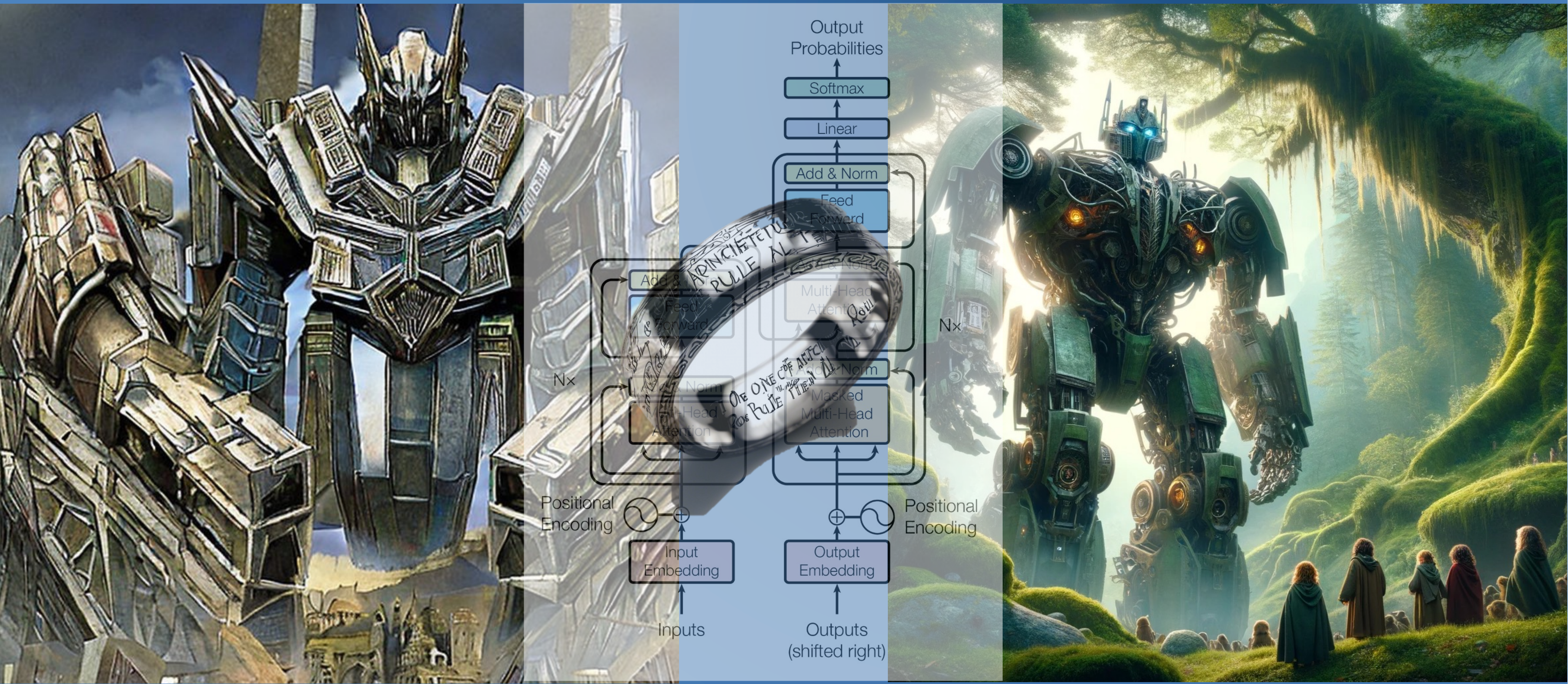
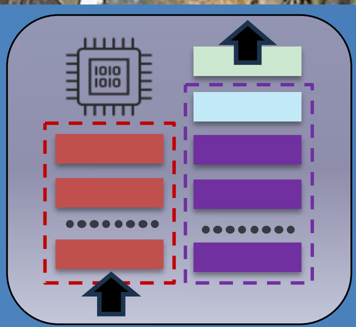


Image: Transformers: ‘ …and one ‘architecture’ to rule them all’ – Juxtaposition by Raeid Saqur (2024).



Transformers

CSC401/2511 – Natural Language Computing – Winter 2024

Gerald Penn, Sean Robertson & Raeid Saqur

Lecture 6

University of Toronto

Logistics (Feb 12, 2024)

- Office hours: Mon 12 – 13.00 (in-person, BA 2270)
- **A2 is now posted!** due Mar 8, 2024 (mid-night) – *errata recap*.
- A2 tutorials planned schedule:
  - Feb 16: A2 tutorial – 1 (ft. Arvid Frydenlund)
  - Mar 1: A2 tutorial – 2 (ft. Julia Watson)
  - Mar 8: A2 – Q/A and OH
- A3: release Mar 9, 2024
- Final exam: date to be finalized soon
- **Have a great Reading week break!** Next week: *no classes or tutorials*).

- Lecture feedback:
  - Anonymous
  - Please share any thoughts/suggestions

- **Questions?**

UNIVERSITY OF TORONTO

A2 – NMT with Transformers

Tasks	Section	Class	criterion	Max mark	Sub-Total	File
1	Building Blocks	LayerNorm	:forward	2		a2_transformer_model.py
2		MultiHeadAttention	:attention	4		
3			:forward	5		
4		FeedForwardLayer	:forward	1	12	
5	Architecture	TransformerEncoderLayer	:pre_layer_norm_forward	2		
6			:post_layer_norm_forward	1		
7		TransformerDecoderLayer	:__init__	4		
8			:pre_layer_norm_forward	2		
9			:post_layer_norm_forward	2		
10		TransformerDecoder	:forward	3		
11		TransformerEncoderDecoder	:create_pad_mask	1		
12			:create_causal_mask	2		
13			:forward	3	20	
15	Decoding: Greedy, and beam-search	TransformerEncoderDecoder	:greedy_decode	5		
17			: expand_encoder_for_beam_search	3		
18			: repeat_and_reshape_for_beam_search	1		
19			: initialize_beams_for_beam_search	6		
20			: pad_and_score_sequence_for_beam_search	3		
21			:finalize_beams_for_beam_search	2	20	
22	Training and testing	TransformerRunner	:train_input_target_split	1		a2_transformer_runner.py
23			:train_step_optimizer_and_scheduler	1		
24			:train_for_epoch	5		
25			:translate	2		
26			:compute_batch_total_bleu	3	12	
27	BLEU score		BLEU score: grouper	2		a2_bleu_score.py
28			BLEU score: n_gram_precision	2		
29			BLEU score: brevity_penalty	2		
30			BLEU score: BLEU_Score(...)	2	8	
32	Analysis			8	8	analysis.pdf
			Total	80	80	

UNIVERSITY OF TORONTO

Transformers

Lecture plan (L6)

- Overview/Recap: RNNs -> Transformers
- Transformer building blocks/components
- Transformer architecture – deep dive
- Review of early popular Transformer based PLMs:
  - Encoder only (BERT, BERTology findings)
  - Encoder-Decoder: unified text-to-text format (T5)
  - Decoder only auto-regressive models (GPT)
- What’s next for LM architectures?
  - Token free architectures
  - Is Attention all we need? Attention free architectures
- Segue to the next lecture **(L7) LLMs**

UNIVERSITY OF TORONTO

Transformer networks

- Breakout paper in 2017: *Attention is all you need* [1]
- **Core idea**: replace recurrent connections with attention

Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [15]	23.75			
Deep-Att + PosUnk [32]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [31]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [8]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [26]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [32]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [31]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [8]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	$\mathbf{3.3 \cdot 10^{18}}$	
Transformer (big)	**28.4**	**41.0**	$2.3 \cdot 10^{19}$	

- Empirical results showcased using machine translation (WMT'14)

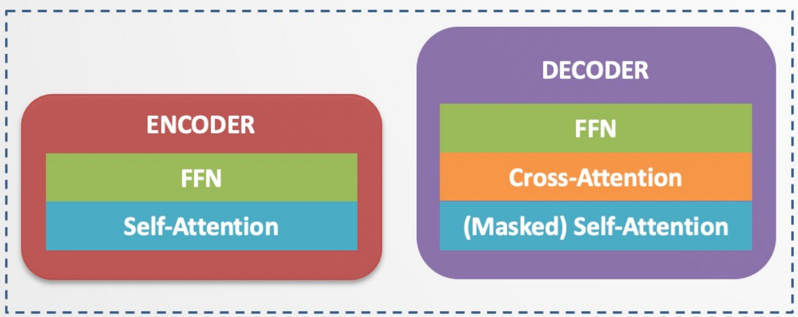
[1] Vaswani, Ashish, et al. "Attention is all you need." *NeurIPS* (2017).

Transformer networks (abstract)

*Recall Slide

Core Idea

Replace recurrence (RNN) with attention



- Encoder uses self-attention

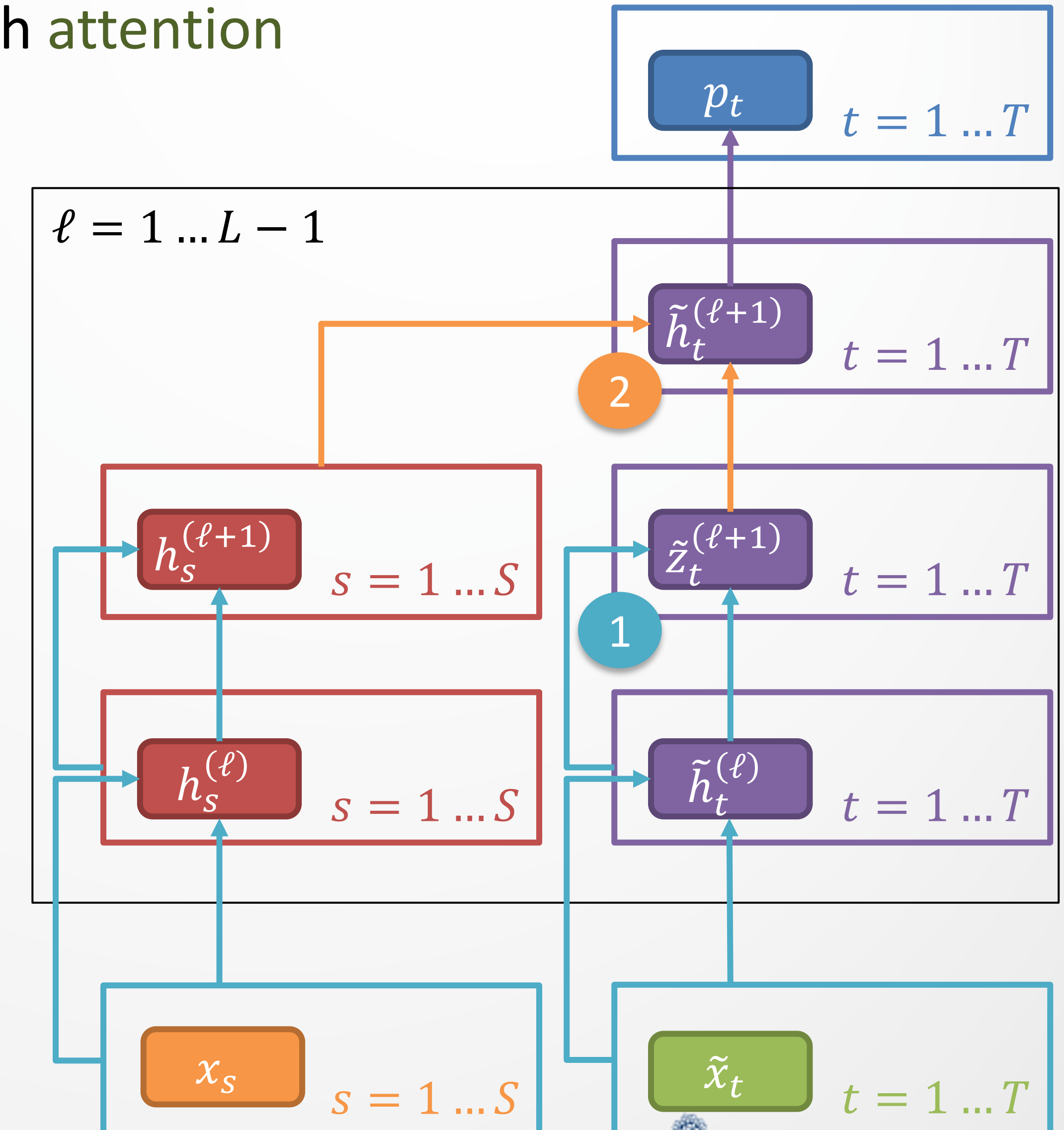
$$h_s^{(\ell+1)} \leftarrow Att_{Enc}\left(h_s^{(\ell)}, h_{1:S}^{(\ell)}\right)$$

Decoder uses 1. self-attention*

$$\tilde{z}_t^{(\ell+1)} \leftarrow Att_{Dec1}\left(\tilde{h}_t^{(\ell)}, \tilde{h}_{1:t}^{(\ell)}\right)$$

then 2. attention with encoder

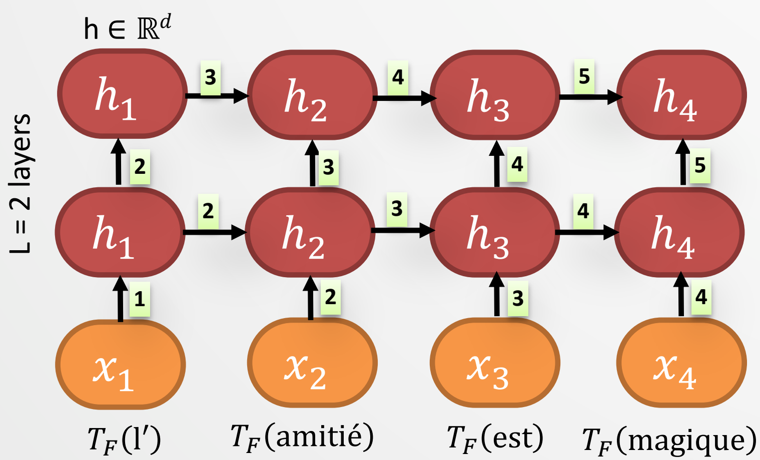
$$\tilde{h}_t^{(\ell+1)} \leftarrow Att_{Dec2}\left(\tilde{z}_t^{(\ell+1)}, h_{1:S}^{(\ell+1)}\right)$$

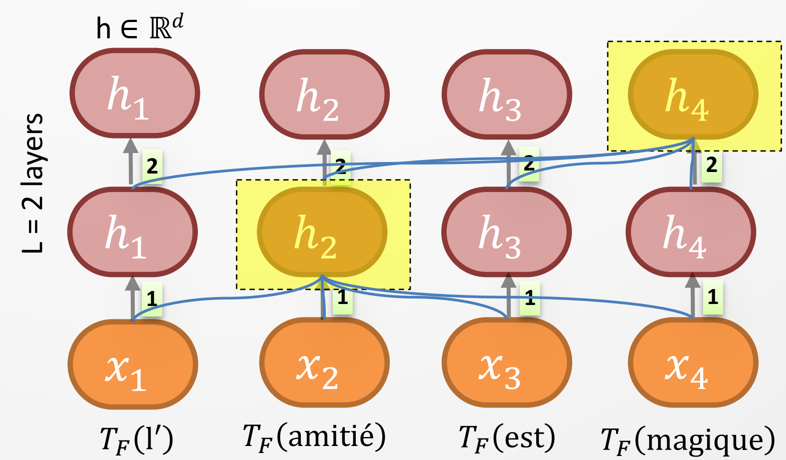


UNIVERSITY OF TORONTO

Transformer motivations

- Limitations of recurrent connections: long-term dependencies, lack of parallelizability, interaction distance (steps to distant tokens).
- Attention allows access to entire sequence
- Lots of computation can be shared, parallelized across sequence indices. Identical layers: [self, cross]-attention, feed-forward w/ tricks
  - Layer norm., residual connections, positional encodings, masking
  - See Vaswani *et al* (2017) for specific architecture



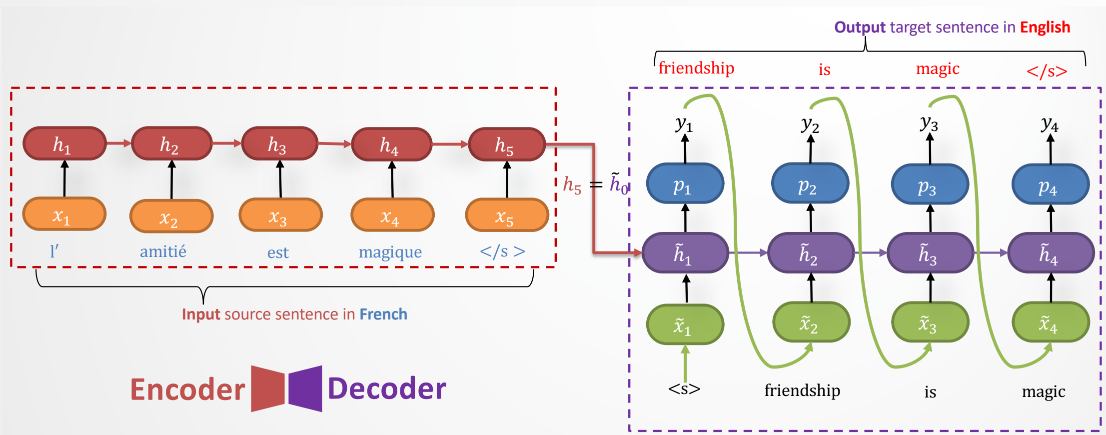


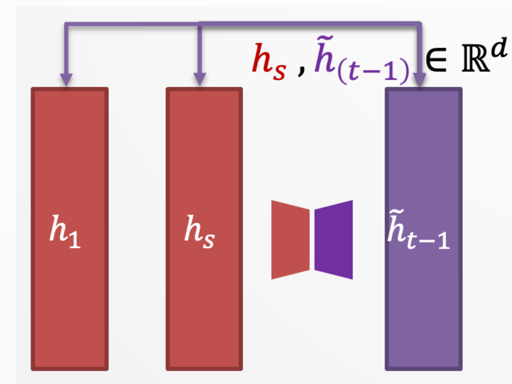
Source sentence (French): *L' amitié est magique*
Target sentence (English): *Friendship is magic*

UNIVERSITY OF TORONTO

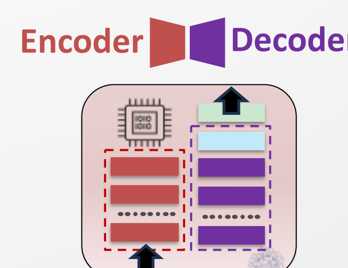
RNNs to Transformers

- **Transformers** is the underlying architecture for all state-of-the-art deep neural models – not just in NLP, but across other modalities too
- So far, we have seen encoder-decoder models using RNN (and variant) architectures using *attention* for memory bottlenecks (***seq2seq+attn***)





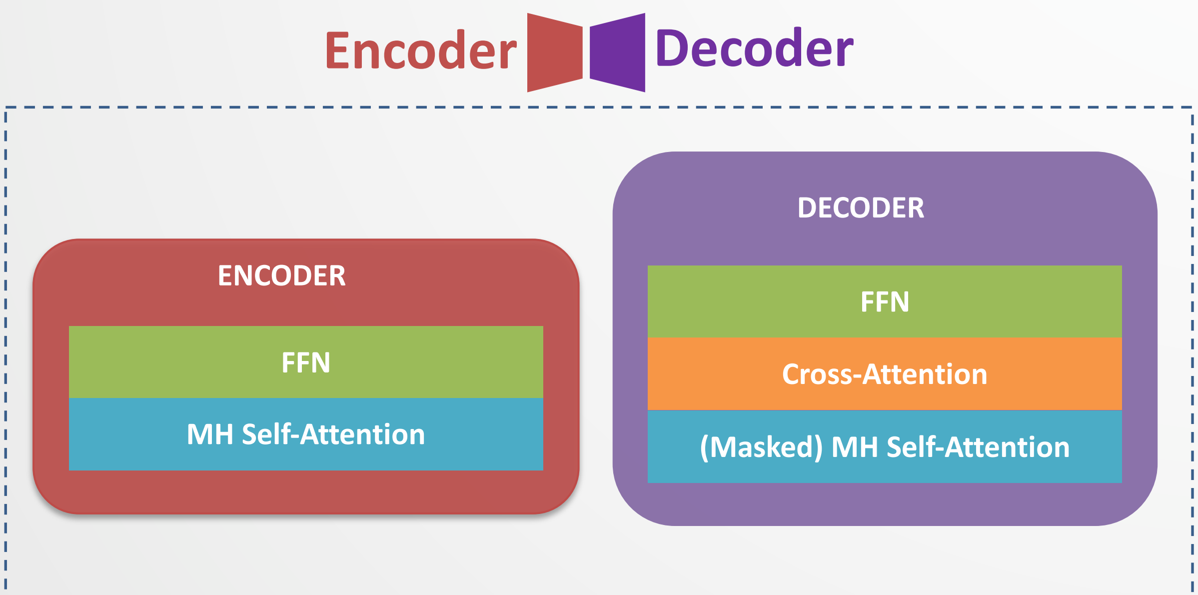
- With Transformers, we use the **same** (enc-dec) paradigm, with different **building blocks**
  - by removing recurrence with parallelizable blocks

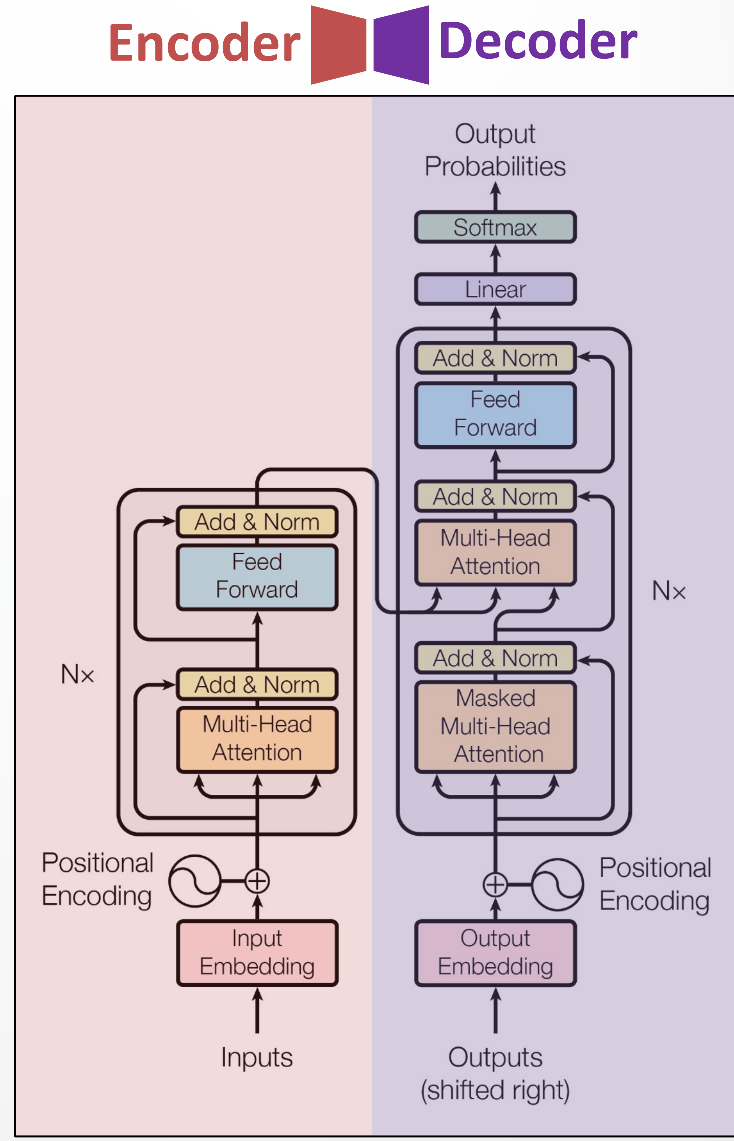


[1] Vaswani, Ashish, et al. "Attention is all you need." *NeurIPS* (2017).

Transformer Architecture

- Architecture diagram from Vaswani[1]
- Building blocks:
  1. Encoder
  2. Decoder

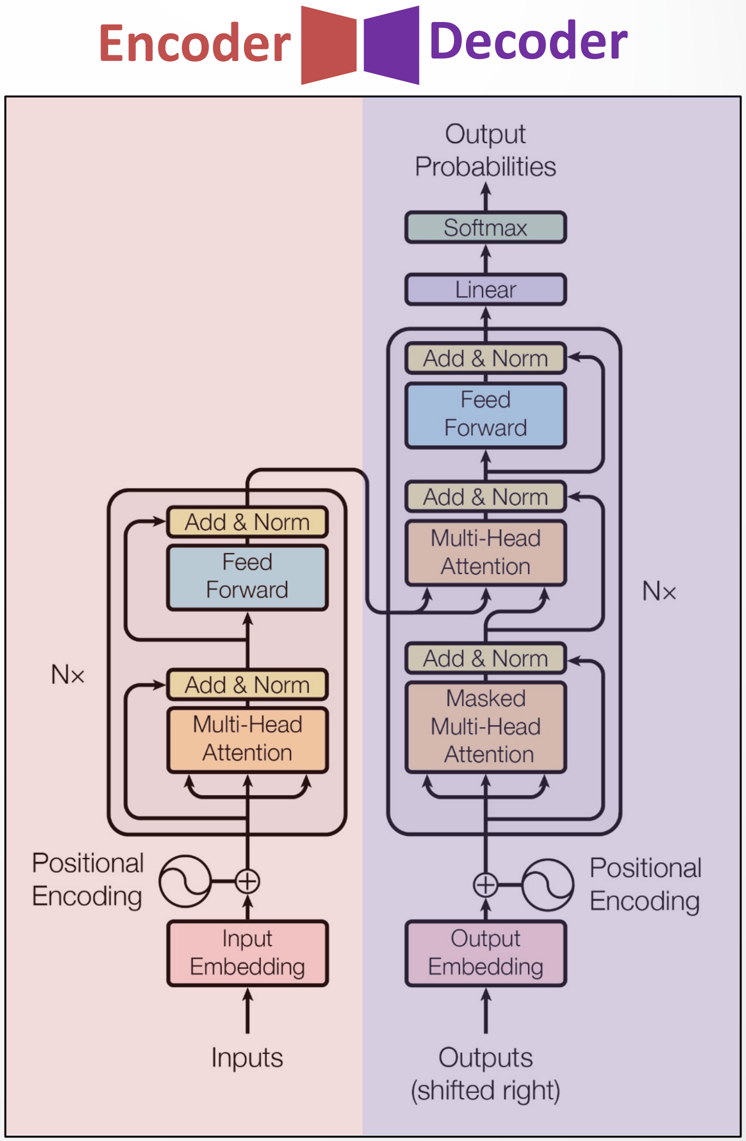




[1] Vaswani, Ashish, et al. "Attention is all you need." *NeurIPS* (2017).

Transformer Architecture

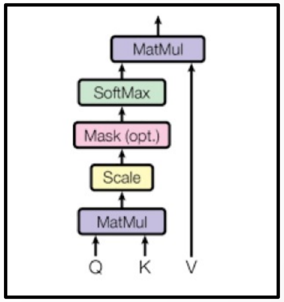
- Architecture diagram from Vaswani[1]
- Building blocks:
  1. Encoder
  2. Decoder
- Main components within building blocks:
  - Attention mechanisms:
    - single and multi-head attention
    - self, cross, and masked attention
  - Feed-forward MLPs (FFN)
  - Layer normalization (LN)
  - Positional encodings (PE)
  - Residual connections



[1] Vaswani, Ashish, et al. "Attention is all you need." *NeuIPS* (2017).

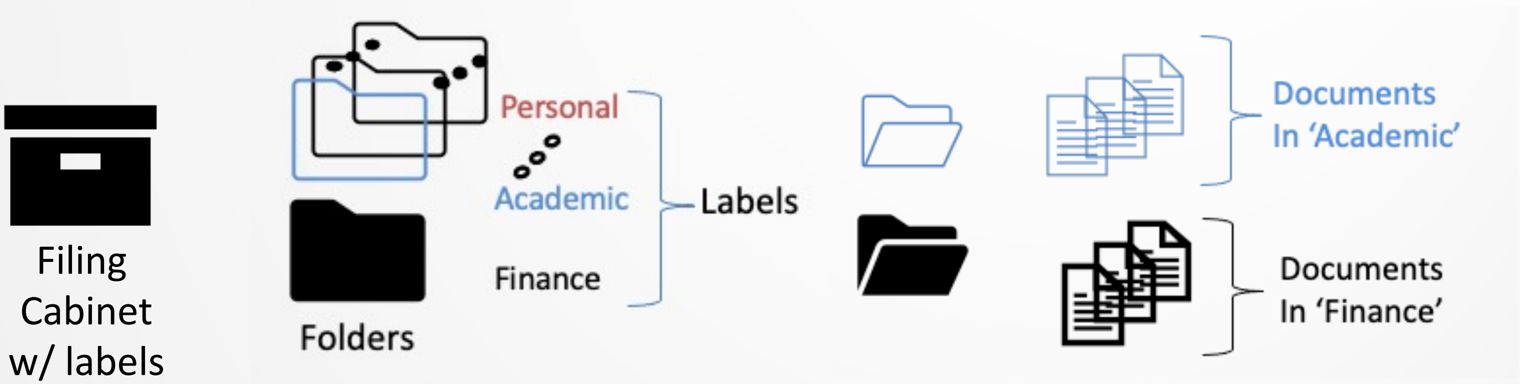
UNIVERSITY OF TORONTO

Transformer Attention(Q,K,V) : Intuition



In the classical roboquity era (2100-2250 AD), humans in designated zones/zoos are only allowed *filing cabinets* and *paper documents* to store information.

ACORN doesn't exist, and UofT students' info (financial, academic, personal) retrieval works as follows:



K Keys

V Values

Similarity Measure Dot Product Attention

Q Queries

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

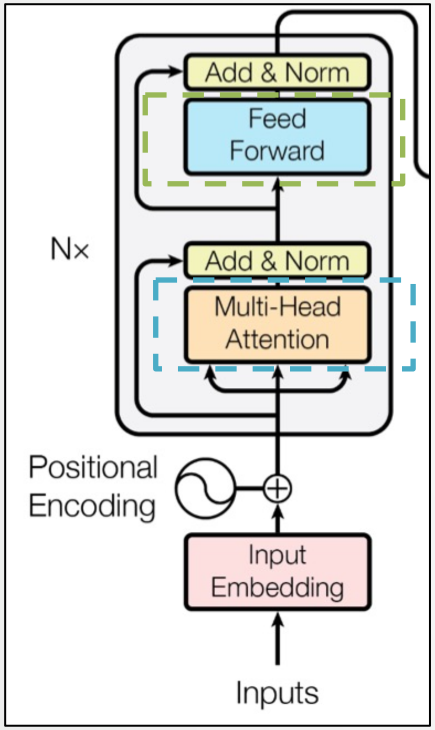
q1 What is the student's current academic standing?

q2 What is the student's current financial status?

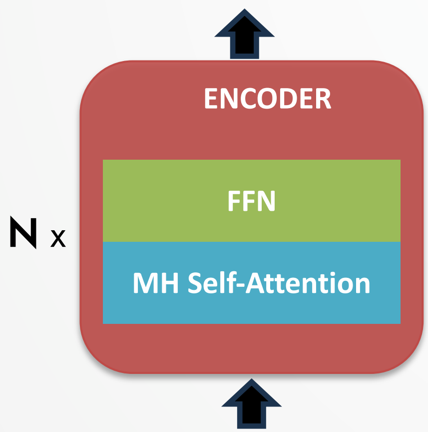
q3 What is the student's residency status in Canada?

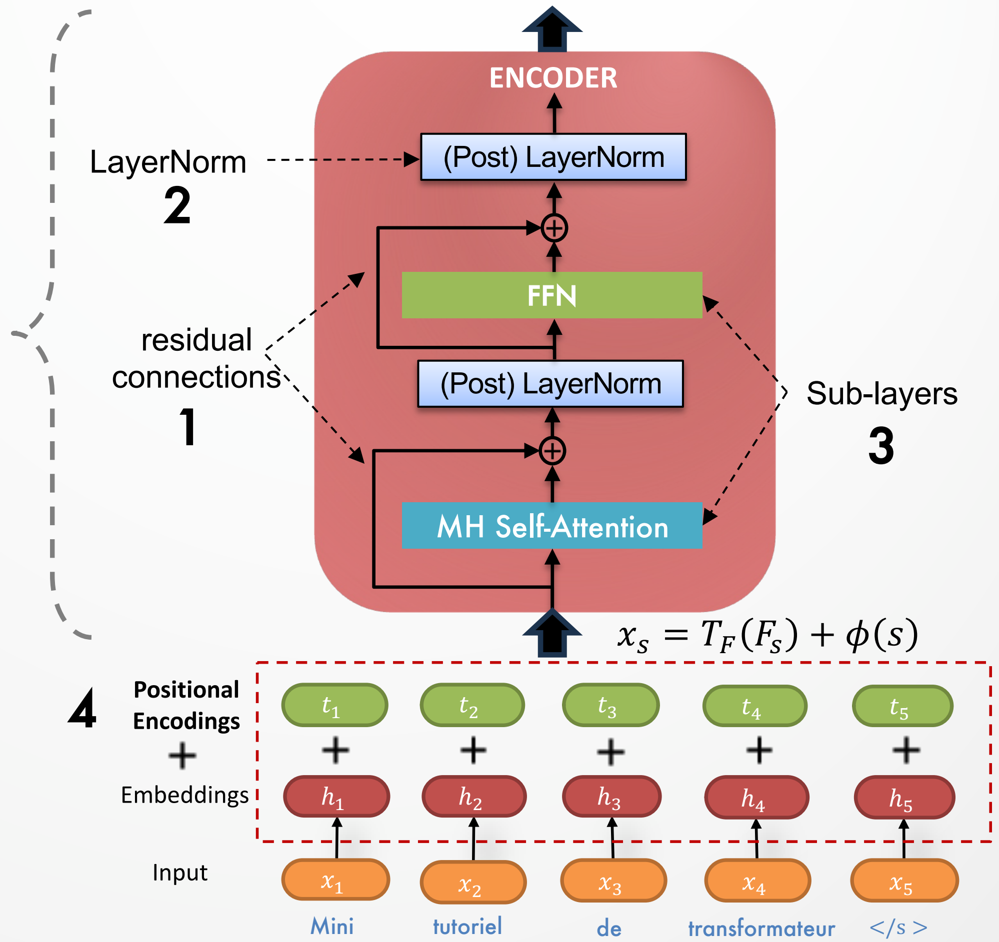
Transformer **Encoder**

F_S: Mini tutoriel de transformateur Encoder Decoder E_T: Tiny transformer tutorial



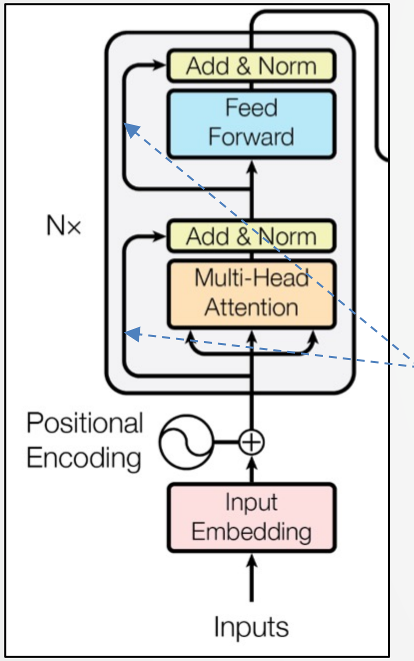
Sub-layers (only) Block

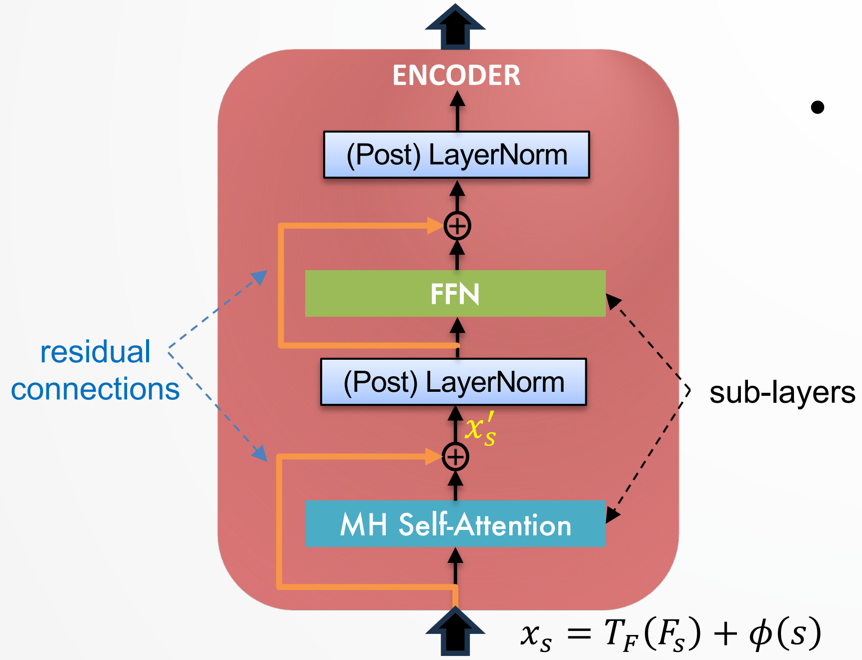




- **Plan**: discuss building blocks:
  1. Residual connections
  2. LayerNorm
  3. Attention and FFN sub-layers
  4. Positional encodings

Residual Connections





- Idea from computer vision[1]

$$x_{l+1} = \mathcal{F}(x_l) + x_l$$

+ Helps smoothen loss curvature allowing better backprop.

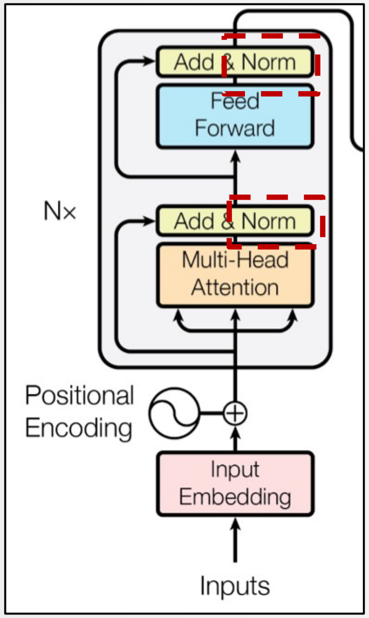
- Problem: NNs struggle to learn the identity function mapping
- Solution: Add back the input embeddings to the sub-layer's output moving up

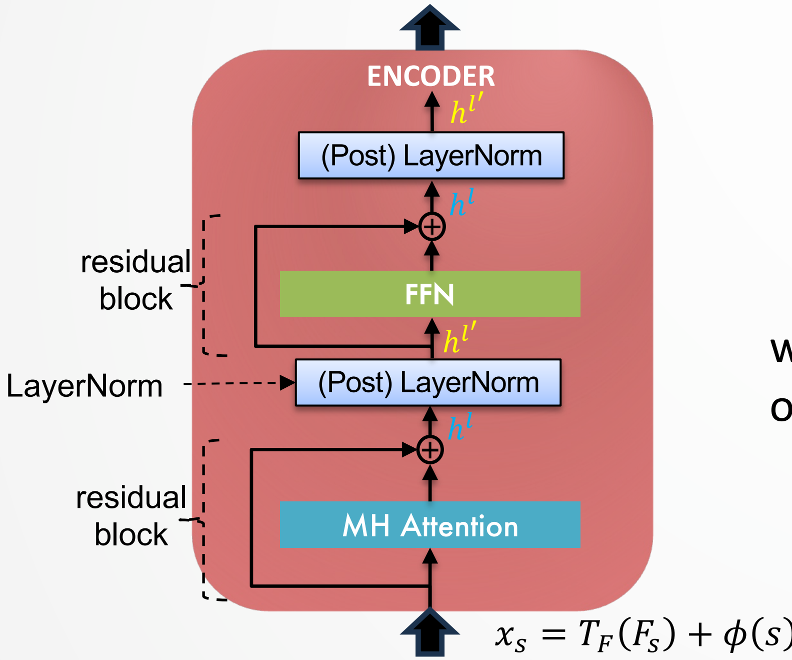
$$x_s' = Sublayer(x_s) + x_s$$

- *Analogy*: think of the information highway analogy. Helps negate forgetting past information by carrying information without distortion.

[1] He, Kaiming, et al. "Deep residual learning for image recognition." *CVPR*. 2016.

Layer Normalization: default (Post-) LN





$$h^{l'} = LayerNorm(h^l)$$
$$= \gamma\left(\frac{h^l - \mu^l}{\sigma^l}\right) + \beta$$

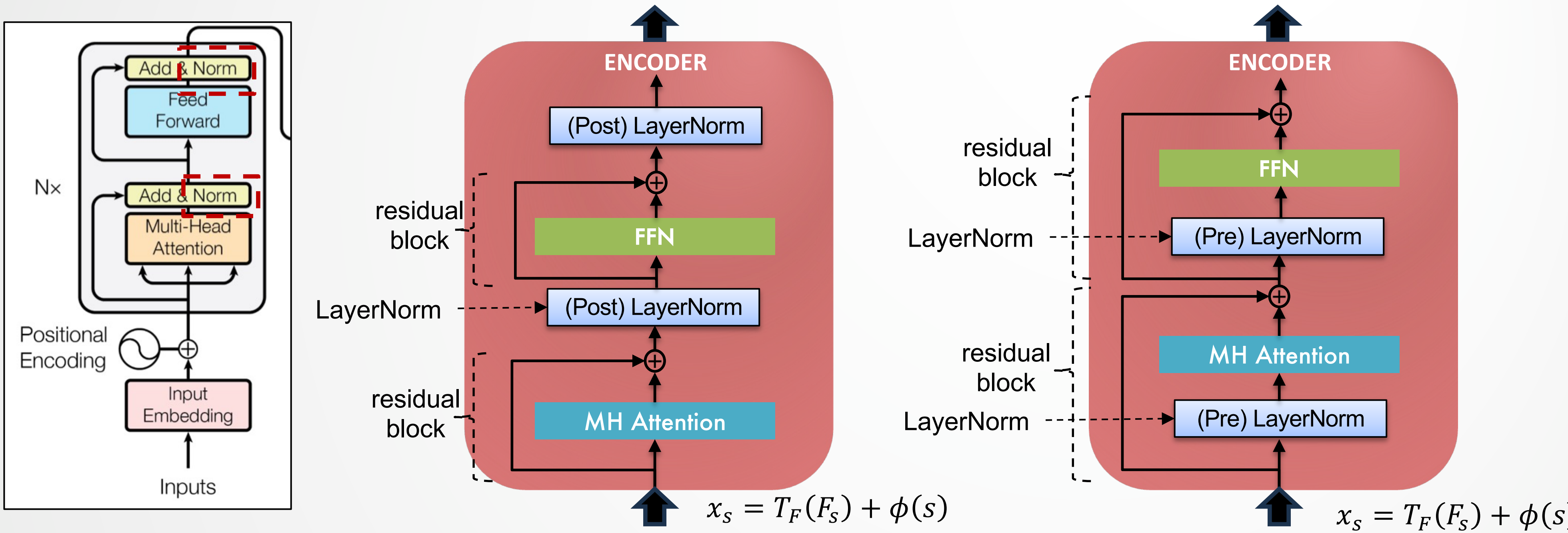
where μ, σ are mean and std. dev. of features in h^l. γ, β are scale, bias params.

$$\mu = \frac{1}{d}\sum_{k=1}^{d} h_k^l$$
$$\sigma^2 = \frac{1}{d}\sum_{k=1}^{d} \left(h_k^l - \mu\right)^2$$

- Layer Normalization[1]:
  - **Normalize** input layer's distribution to 0 mean and 1 standard deviation.
  - Removes uninformative variation in layer's features

[1] Ba, Jimmy Lei, Jamie Ryan Kiros, and Geoffrey E. Hinton. "Layer normalization. 2016" [link]
[2] Xiong, Ruibin, et al. "On layer normalization in the transformer architecture." *ICML*. PMLR, 2020. [link]

Layer Normalization Variant: Pre-LN

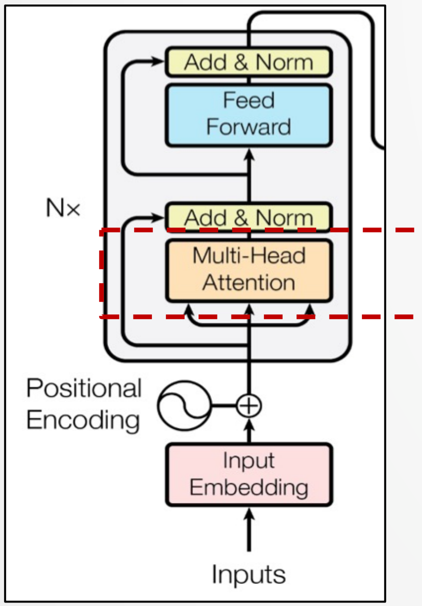


- Layer Normalization[1]: two popular variants
  - *Post layer normalization* (Post-LN): original Transformer model: requires learning rate warm-up due to initial instability of large output gradients.
  - *Pre layer normalization* (**Pre-LN**): puts layer-norm within the residual block. Allows removing warm-up stage. More stable training initialization.

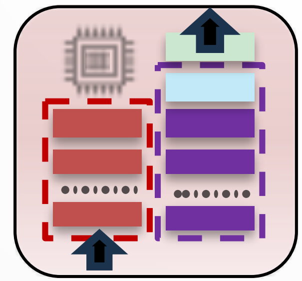
[1] Ba, Jimmy Lei, Jamie Ryan Kiros, and Geoffrey E. Hinton. "Layer normalization. 2016" [link]
[2] Xiong, Ruibin, et al. "On layer normalization in the transformer architecture." *ICML*. PMLR, 2020. [link]

Transformer Encoder - Self Attention

- Our running example:



E_T: Tiny transformer tutorial



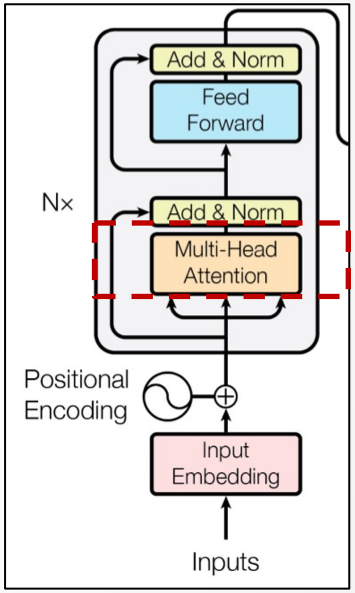
	x_1	x_2	x_3	x_4	x_5
F_S:	Mini	tutoriel	de	transformateur	</s >

- Three weight matrices W^Q, W^K, W^V
- Let's look at 'self-attention' with input F_S

[1] Vaswani, Ashish, et al. "Attention is all you need." *NeuIPS* (2017).

Transformer Encoder - Self Attention

- Recall the attention steps we discussed in last lecture
- Steps:
  1. Calculate the query, key, and value for each token
     - Attention of each query (q_i) against all the keys ($k_{1:j}$)
  2. Calculate the **attention score** between query and keys
  3. **Normalize** the attention scores by applying softmax
  4. Calculate values by taking a **weighted sum**



$$q_i = W^Q x_i$$
$$k_i = W^K x_i$$
$$v_i = W^V x_i$$

$$a_{i,j} = score(q_i, k_j)$$
$$a_{i,j} = q_i . k_j$$
$$a_{i,j} = \frac{q_i . k_j}{\sqrt{d_k}}$$

$$\alpha_{i,j} = softmax(a_{i,1:K})$$
$$\alpha_{i,j} = \frac{exp^{(a_{i,j})}}{\sum_{k=1}^{K} exp^{(a_{i,k})}}$$

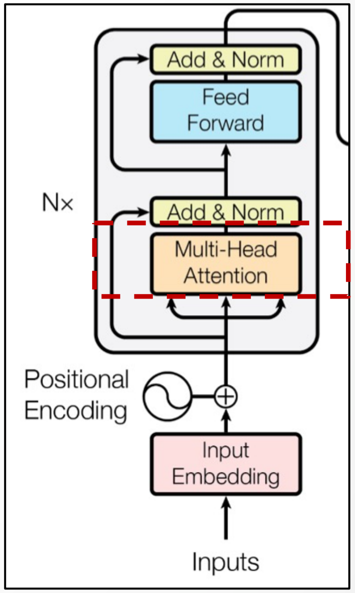
$$c_i = \sum_j \alpha_{i,j} v_j$$

[1] Vaswani, Ashish, et al. "Attention is all you need." *NeurIPS* (2017).

UNIVERSITY OF TORONTO

Transformer Encoder - Self Attention

- Recall the attention steps we discussed in last lecture
- Steps:
  1. Calculate the query, key, and value for each token
     - Attention of each query (q_i) against all the keys ($k_{1:j}$)
  2. Calculate the **attention score** between query and keys
  3. **Normalize** the attention scores by applying softmax
  4. Calculate values by taking a **weighted sum**



Vectorized notation:

$$Q = XW^Q$$
$$K = XW^K$$
$$V = XW^V$$

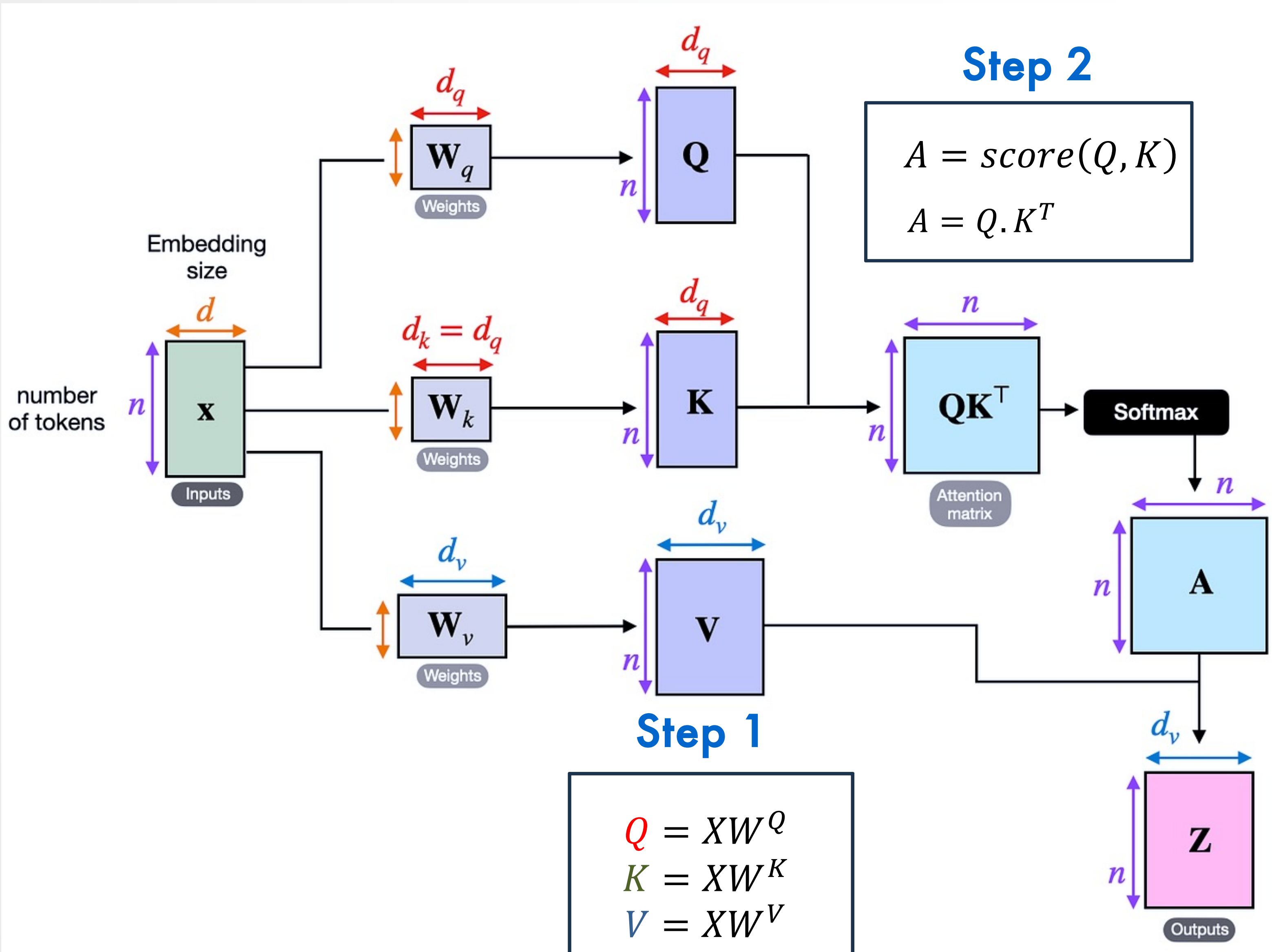
$$A = score(Q, K)$$
$$A = Q.K^T$$
$$A = \frac{Q.K^T}{\sqrt{d_k}}$$

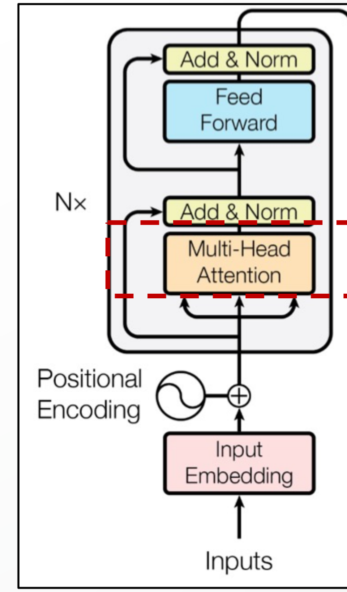
$$A = softmax(A)$$
$$A = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)$$

$$Z = A.V$$

[1] Vaswani, Ashish, et al. "Attention is all you need." *NeuIPS* (2017).

Transformer Encoder - Self Attention





Step 1

$$Q = XW^Q$$
$$K = XW^K$$
$$V = XW^V$$

Step 2

$$A = score(Q, K)$$
$$A = Q.K^T$$

Step 3

$$A = softmax(A)$$

Step 4

$$Z = A.V$$

[1] Vaswani, Ashish, et al. "Attention is all you need." *NeuIPS* (2017).

Multi-head Self Attention (MHA)

- As alluded to in L5, multi-head attention (MHA) allows to jointly attend to information from different representation subspaces at different positions

$$MHA(Q,K,V) = Concat(head_1, \ldots, head_h)W^O$$
$$where\ head_i = Attention\left(QW_i^Q, KW_i^K, VW_i^V\right)$$

And projections are parameter matrices:

$$W_i^Q \in \mathbb{R}^{d_{model} \times d_k} \qquad W^O \in \mathbb{R}^{hd_v \times d_{model}}$$
$$W_i^K \in \mathbb{R}^{d_{model} \times d_k} \qquad d_k = d_v = \frac{d_{model}}{h}$$
$$W_i^V \in \mathbb{R}^{d_{model} \times d_v}$$

[1] Vaswani, Ashish, et al. "Attention is all you need." *NeurIPS* (2017).

Feed-forward (FFN) layers

- Attention only **re-weighs** the **value vectors**
- We need to apply *non-linearities* (activations) to enable (deep) learning
- The feed-forward layer(s) (**FFN**) provide non-linear activation to attention layer outputs
- Specifically, each output x undergoes two (layer) linear transformations with a ReLU activation in between:

$$Z = A.V$$

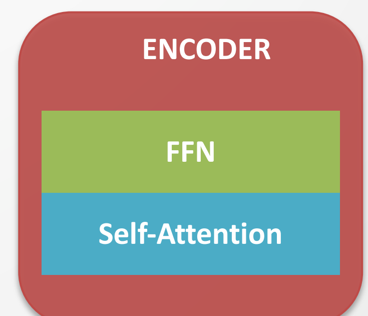
$$FFN(x) = \max(0, xW_1 + b_1)\, W_2 + b_2$$

$$z_i = \sum_j \alpha_{i,j} v_j$$

- FFN sub-layer is applied to each token pos. separately and identically
- Given x is a sequence of tokens $(x_1, \dots, x_S)$, point-wise computation of FFN sub-layer on any token x_i is:

$$FFN(x_i) = ReLU(\, x_i W_1 + b_1)\, W_2 + b_2$$

where W_1, W_2, b_1 *and* b_2 are parameters



[1] Vaswani, Ashish, et al. "Attention is all you need." *NeurIPS* (2017).

Position (in)dependence

*Recall slide: L5_NMT

- Attention mechanism is agnostic to sequence order
  - For permutation vector v s.t. $sorted(v) = (1,2, \dots, V)$

$$Att(a, b_v) = Att(a, b_{1:V})$$

- **Caveat:** but the *word order* **matters** in language translation
- **Solution**: encode position in input:

$$x_s = T_F(F_s) + \phi(s)$$

Transformer - Positional Encoding

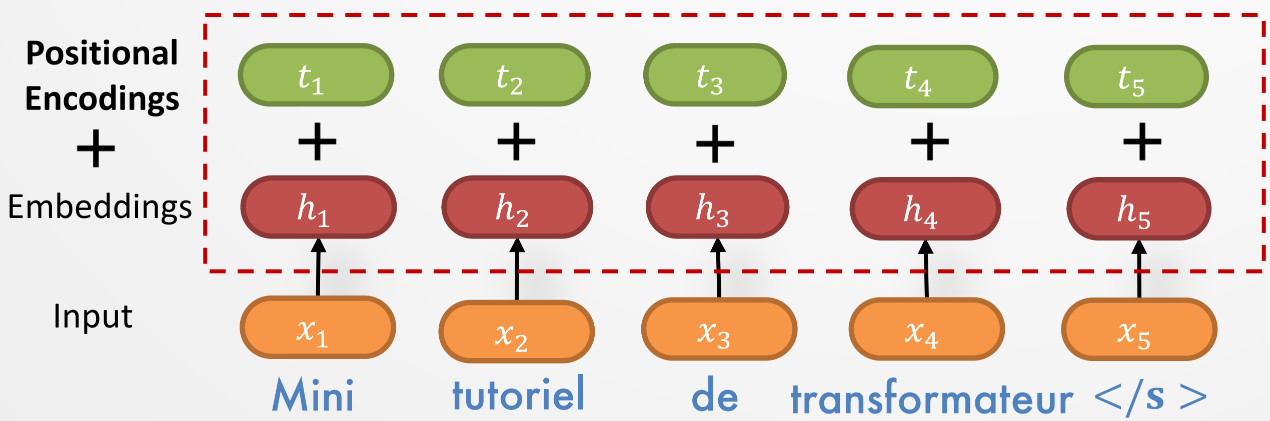
*Recall slide: L5_NMT

Idea

Add positional information of an input token in the sequence into the input embedding vectors.

$$PE_{(\underbrace{pos}_{\text{token position}},\, \mathbf{2}\underset{\uparrow \text{dim index}}{i})} = \mathbf{sin}\left(\frac{pos}{10000^{\left(\frac{2i}{d_{model}}\right)}}\right);\; PE_{(pos,\, \mathbf{2}i+\mathbf{1})} = \mathbf{cos}\left(\frac{pos}{10000^{\left(\frac{2i}{d_{model}}\right)}}\right)$$

- The *positional encodings* (PE) have the same dimension d_{model} as the embeddings (for summation)
- Many choices of PEs possible: learned or fixed.



UNIVERSITY OF TORONTO

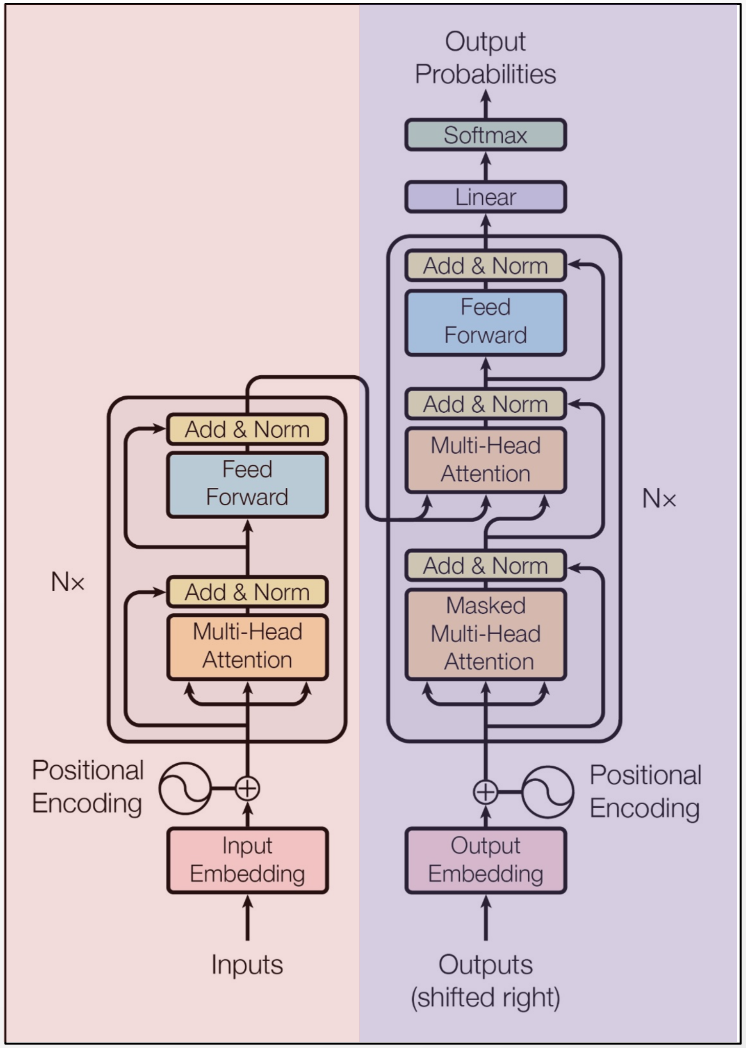
Recap: Transformer Architecture

- Architecture diagram from Vaswani[1]
- Building blocks:

1. Encoder
2. **Decoder**

- Main components within building blocks:
  - Attention mechanisms:
    - single and multi-head attention
    - self, **cross, and masked attention**
  - Feed-forward MLPs (FFN)
  - Layer normalization (LN)
  - Positional encodings (PE)
  - Residual connections

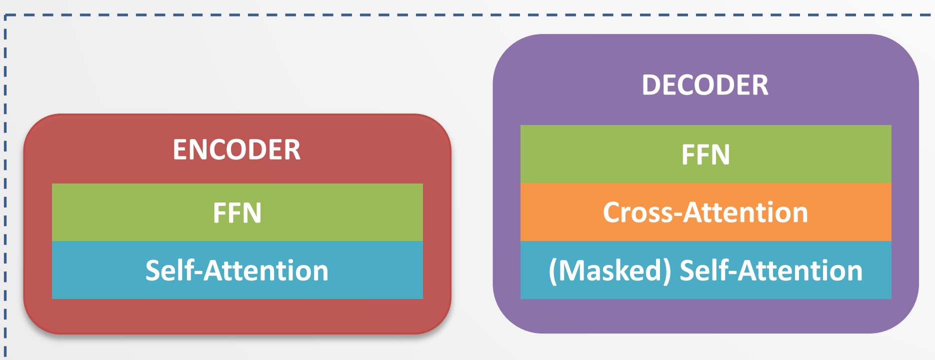
Encoder Decoder



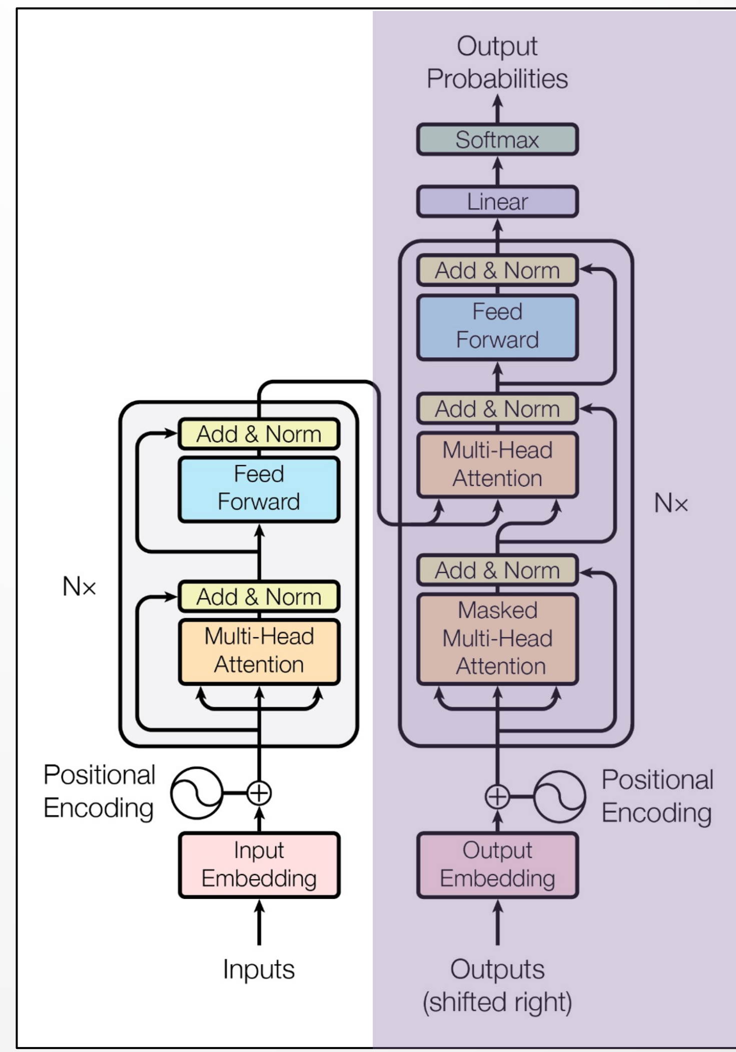
UNIVERSITY OF TORONTO

Next block: Transformer Decoder

- Layer normalization, residual connections, FFNs are identical to the encoder block
- Thus, we focus on remaining:
  - Masked/Causal self-attention sub-layer
  - Cross-attention



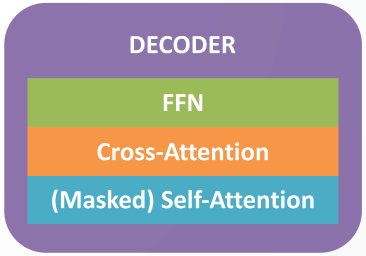
Encoder Decoder



[1] Vaswani, Ashish, et al. "Attention is all you need." *NeuIPS* (2017).

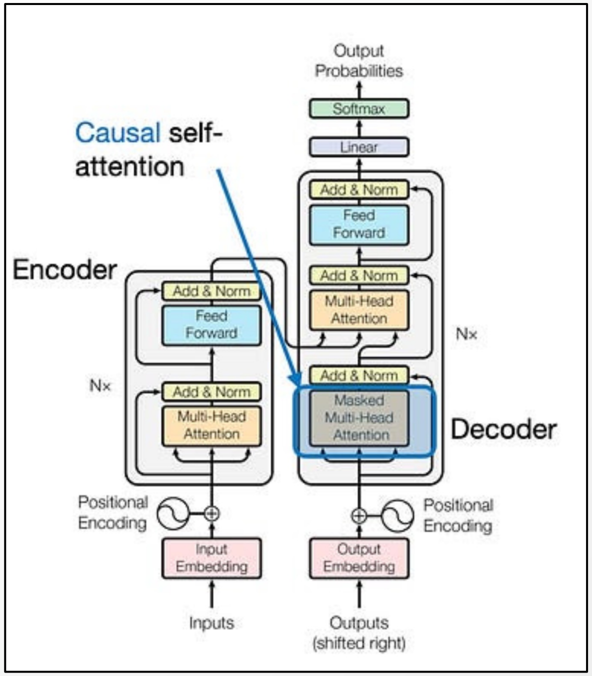
Decoder – Masked Self-Attention

- Masked (Multi-head) self-attention:
  - Enforce auto-regressive language modeling objective. The decoder cannot peek and pay attention to the (unknown) future words
  - **Solution**: use a *look-head mask M,* by setting attention scores of future tokens to –inf.



$$\begin{array}{c|cccc} & a^K & b^K & c^K & D^K \\ \hline a^Q & 0 & -\infty & -\infty & -\infty \\ b^Q & 0 & 0 & -\infty & -\infty \\ c^Q & 0 & 0 & 0 & -\infty \\ D^Q & 0 & 0 & 0 & 0 \end{array}$$

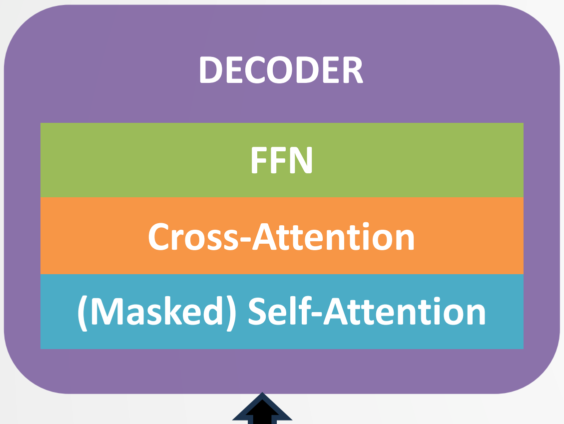
$$a_{ij} = \begin{cases} q_i^T . k_j, j < i \\ -\infty, j \geq i \end{cases}$$



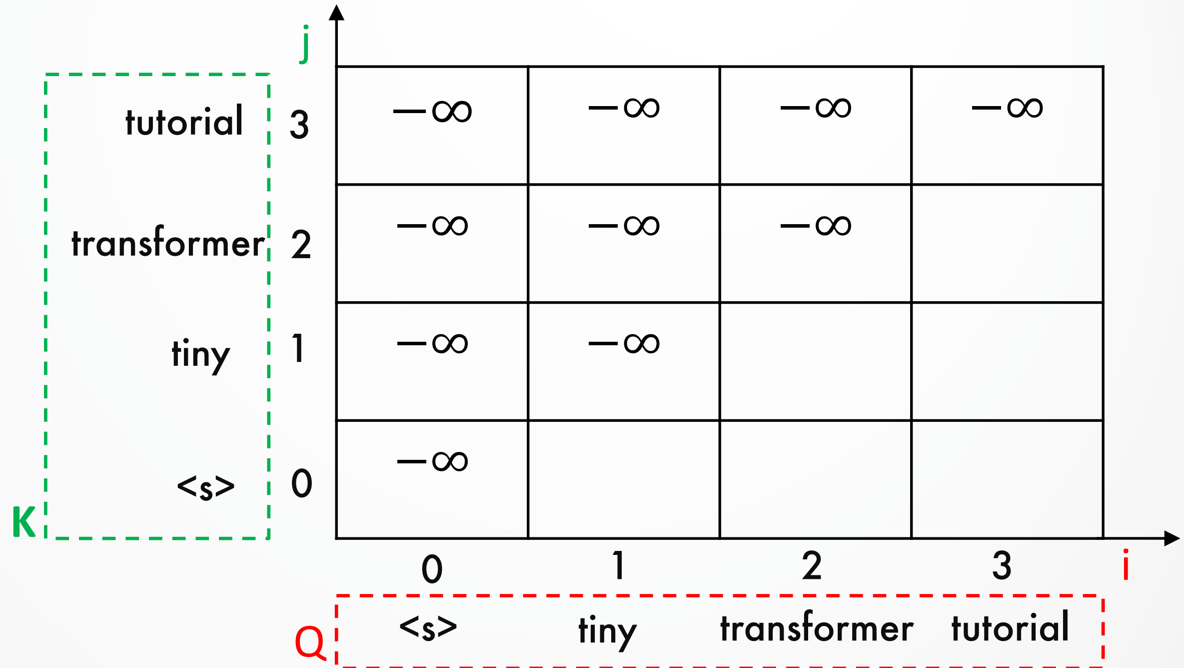
[1] Vaswani, Ashish, et al. "Attention is all you need." *NeuIPS* (2017).

UNIVERSITY OF TORONTO

Decoder – Masked Self-Attention



E_T: <s> tiny transformer tutorial



$$
\begin{array}{c c}
 & \begin{array}{cccc} a^K & b^K & c^K & D^K \end{array} \\
\begin{array}{c} a^Q \\ b^Q \\ c^Q \\ D^Q \end{array} &
\begin{bmatrix} 0 & -\infty & -\infty & -\infty \\ 0 & 0 & -\infty & -\infty \\ 0 & 0 & 0 & -\infty \\ 0 & 0 & 0 & 0 \end{bmatrix}
\end{array}
$$

$$a_{ij} = \begin{cases} q_i^T . k_j, j < i \\ -\infty, j \geq i \end{cases}$$

Recall

$$a_{i,j} = score(q_i, k_j)$$

or,

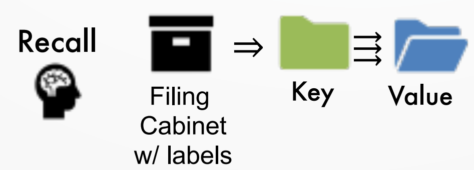
$$A = score(Q, K)$$

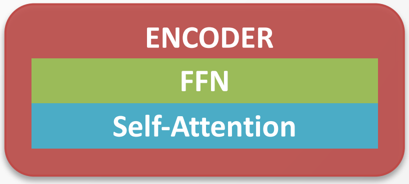
[1] Vaswani, Ashish, et al. "Attention is all you need." *NeuIPS* (2017).

Encoder-Decoder (Cross) Attention

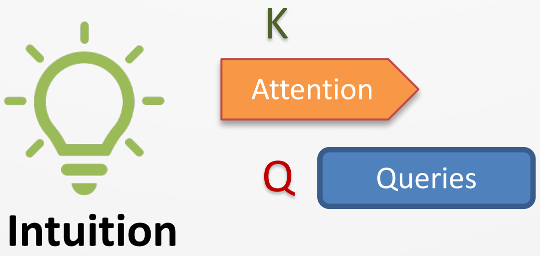
- In self-attention: Q, K and V has same source (tokens)
- **Cross attention** is encoder <> decoder attention between encoder and decoder's output vectors (like we are used to from L5-NMT
- Using our running e.g. and notations:
  - Let $h_1, \ldots, h_S$ be encoder output vectors, where $h_i \in \mathbb{R}^{d_k}$
  - Let $\tilde{h}_1, \ldots, \tilde{h}_T$ be decoder output vectors, where $\tilde{h}_i \in \mathbb{R}^{d_q}$

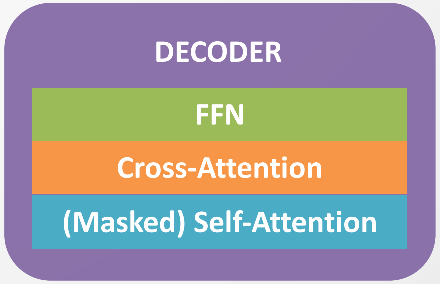
- Then, keys and values: K and V comes from encoder (or, *memory*):
  - $k_i = K h_i \,,\; v_i = V h_i$



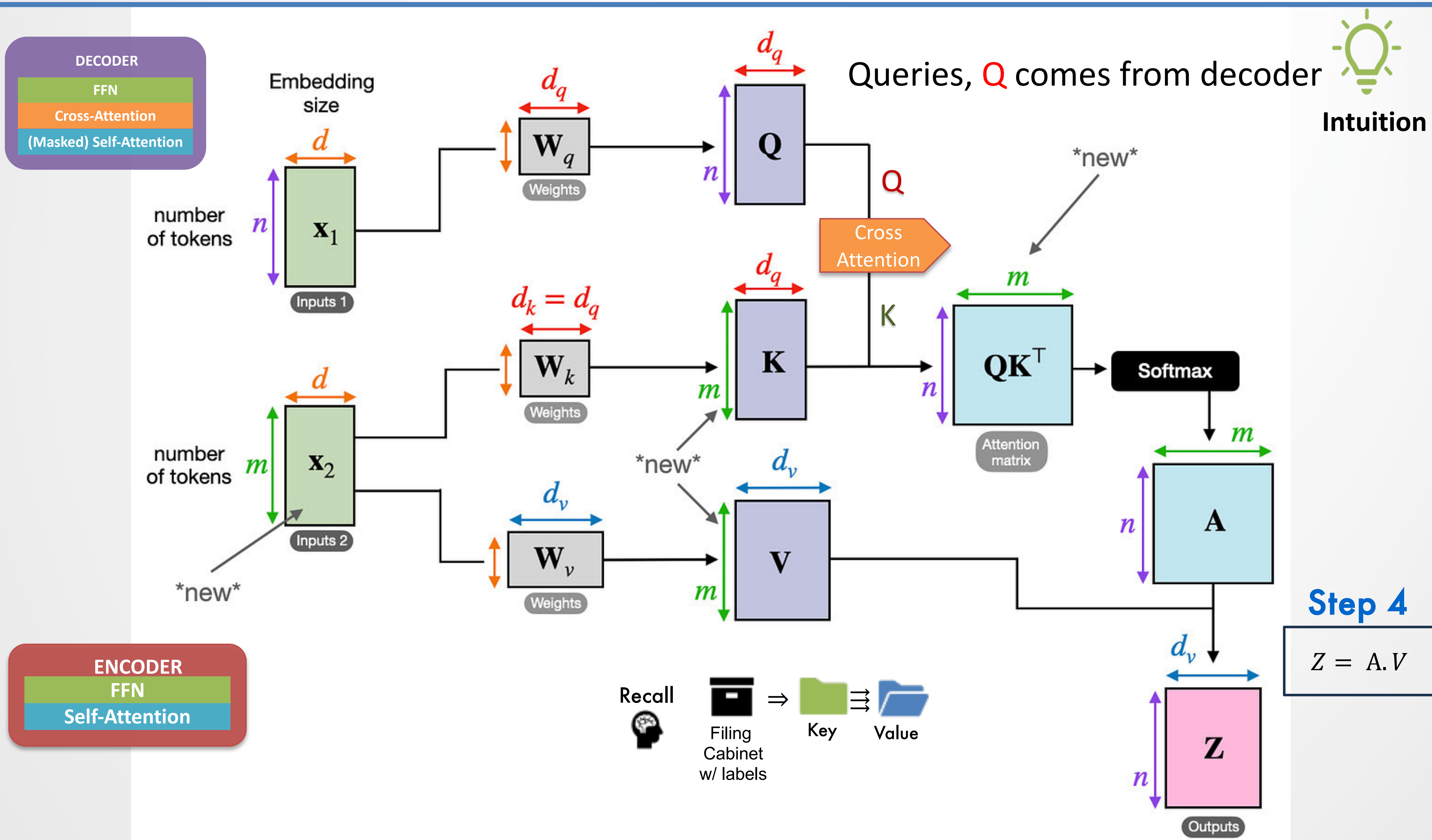


- Queries, Q comes from decoder:
  - $q_i = Q \tilde{h}_i$

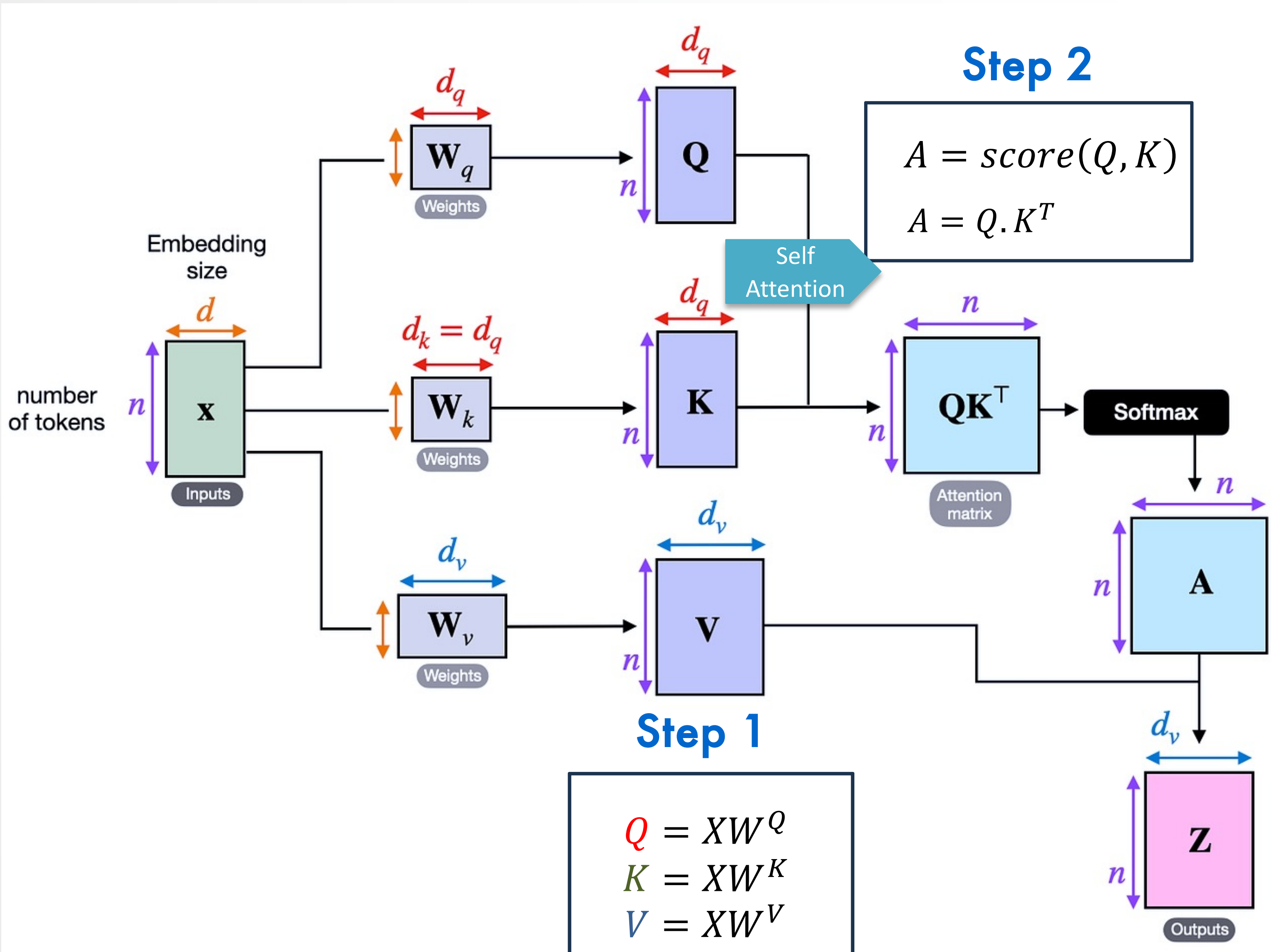




Encoder-Decoder (Cross) Attention



(Compare) Encoder - Self Attention

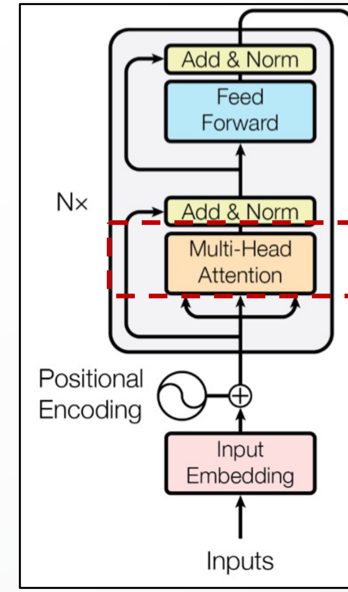


Step 2

$$A = score(Q, K)$$
$$A = Q.K^T$$

Step 1

$$Q = XW^Q$$
$$K = XW^K$$
$$V = XW^V$$



Step 3

$$A = softmax(A)$$

Step 4

$$Z = A.V$$

[1] Vaswani, Ashish, et al. "Attention is all you need." *NeuIPS* (2017).

UNIVERSITY OF TORONTO

Recap

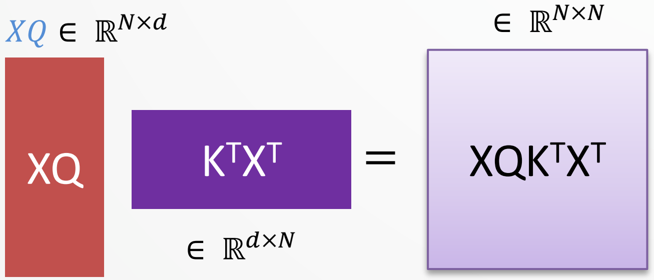
- We have now covered all the primitives you need for building a transformer!
- These are too abstract, but not to worry …
- Assignment 2 was designed for you to implement all these concepts into a working MT model of your own!

Tasks	Section	Class	criterion	Max mark	Sub-Total	File
1	Building Blocks	LayerNorm	:forward	2		a2_transformer_model.py
2		MultiHeadAttention	:attention	4		
3			:forward	5		
4		FeedForwardLayer	:forward	1	12	
5	Architecture	TransformerEncoderLayer	:pre_layer_norm_forward	2		
6			:post_layer_norm_forward	1		
7		TransformerDecoderLayer	:__init__	4		
8			:pre_layer_norm_forward	2		
9			:post_layer_norm_forward	2		
10		TransformerDecoder	:forward	3		
11		TransformerEncoderDecoder	:create_pad_mask	1		
12			:create_causal_mask	2		
13			:forward	3	20	
15			:greedy_decode	5		

UNIVERSITY OF TORONTO

Transformers - Drawbacks

- Attention's **quadratic** computation **cost**
  - Function of sequence length N, and token dimension d
  - Computing all token pairs mean the function grows quadratically with N, $O(N^2d)$ unlike RNNs: $O(Nd)$



- Can you see why this could be the biggest hurdle for increasing a transformer LM's **context length** (i.e., the size of input it can process)?

Transformers - Drawbacks

- Context (input) size limitation:
  - Dimension d in modern LLMs are ~>3K
  - If one sentence length is ~10-30 word tokens, then computation scales with 10^2-30^2 times d
  - Thus, modern LLMs set a bound on N (usually 512 tokens)
  - But, we want **N** to be much larger
  - E.g., processing a document (N > 10K) at one go (instead of chunking by N for every call)

- Active research area: improving the quadratic cost of attention, like self-attention with linear complexity[1]
  - + Roformer, flash attention, sliding-window etc.

[1] Wang, Sinong, et al. "Linformer: Self-attention with linear complexity." (2020). link.

UNIVERSITY OF TORONTO

Transformers - Drawbacks

- Other drawbacks, improvement areas:
  - **Positional encoding representations**:
    - Do we need absolute indices to represent position?

$$f_{t:t\in\{q,k,v\}}(\boldsymbol{x}_i, i) := \boldsymbol{W}_{t:t\in\{q,k,v\}}(\boldsymbol{x}_i + \boldsymbol{p}_i)$$

  - Slew of works and variants has been proposed to the vanilla (sinusoidal, absolute) position encoding we saw

- General trend:
  - Move towards **relative position encoding**
  - E.g. Relative linear position attention [Shaw et al. 2018]

$$f_q(\boldsymbol{x}_m) := \boldsymbol{W}_q \boldsymbol{x}_m$$
$$f_k(\boldsymbol{x}_n, n) := \boldsymbol{W}_k(\boldsymbol{x}_n + \tilde{\boldsymbol{p}}_r^k)$$
$$f_v(\boldsymbol{x}_n, n) := \boldsymbol{W}_v(\boldsymbol{x}_n + \tilde{\boldsymbol{p}}_r^v)$$

where $\tilde{\boldsymbol{p}}_r^k, \tilde{\boldsymbol{p}}_r^v \in \mathbb{R}^d$ are trainable relative position embeddings

Relative distance between pos. m and n

$$r = clip(m - n, r_{min}, r_{max})$$

[Aside] Rotary Position Embeddings

- RoPE – now adopted default in modern LLMs
  - Encodes absolution position with a rotation matrix
  - RoPE + Transformer = **RoFormer**

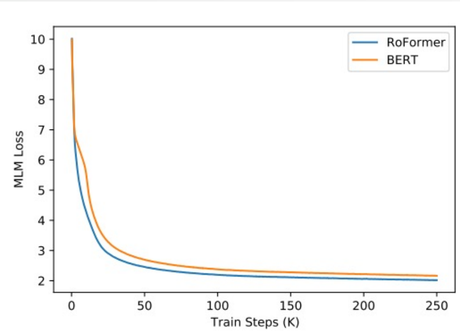
$$f_{\{q,k\}}(\boldsymbol{x}_m, m) = \boldsymbol{R}^d_{\Theta,m} \boldsymbol{W}_{\{q,k\}} \boldsymbol{x}_m$$

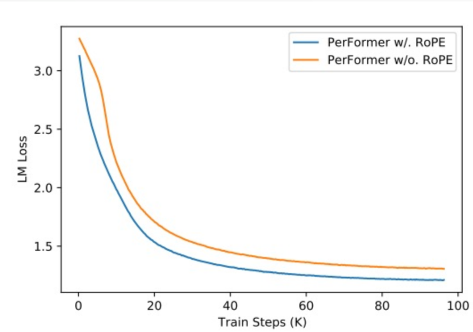
$$f_{\{q,k\}}(\boldsymbol{x}_m, m) = \begin{pmatrix} \cos m\theta & -\sin m\theta \\ \sin m\theta & \cos m\theta \end{pmatrix} \begin{pmatrix} W^{(11)}_{\{q,k\}} & W^{(12)}_{\{q,k\}} \\ W^{(21)}_{\{q,k\}} & W^{(22)}_{\{q,k\}} \end{pmatrix} \begin{pmatrix} x^{(1)}_m \\ x^{(2)}_m \end{pmatrix}$$

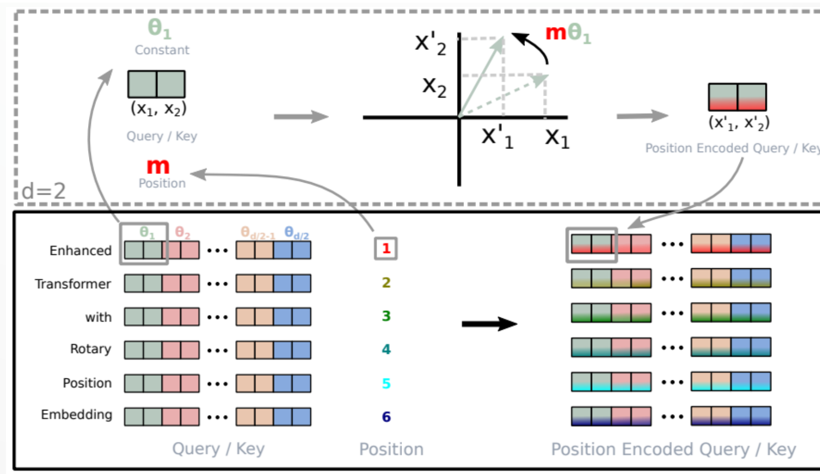
Model	BLEU
Transformer-baseVaswani et al. [2017]	27.3
RoFormer	**27.5**

Table 2: Comparing RoFormer and BERT by fine tuning on downstream GLEU tasks.

Model	MRPC	SST-2	QNLI	STS-B	QQP	MNLI(m/mm)
BERTDevlin et al. [2019]	88.9	93.5	90.5	85.8	71.2	84.6/83.4
RoFormer	**89.5**	90.7	88.0	**87.0**	**86.4**	80.2/79.8





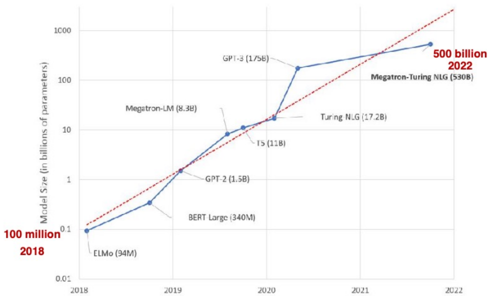


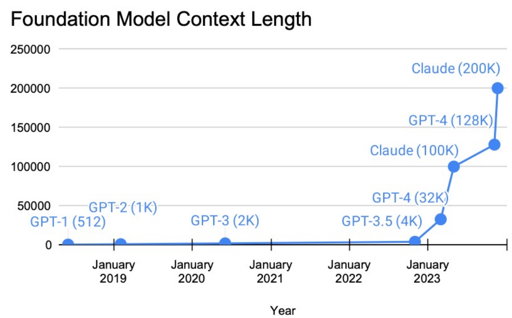
[1] **RoPE**: Su, Jianlin, et al. "Roformer: Enhanced transformer with rotary position embedding." (2021). [arxiv]

UNIVERSITY OF TORONTO

[Aside] GEMM-Based Architecture

Foundation Models Scaling in Model Size and Context

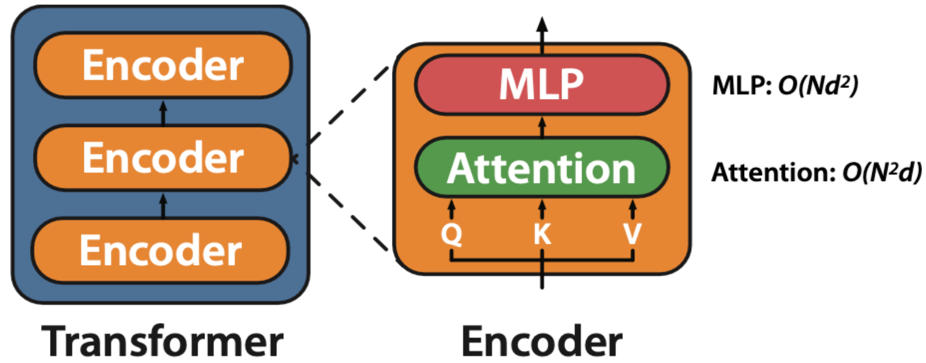




Scaling in Model Size:
100M to 500B in Four Years

Scaling in Sequence Length:
512 to 200K in Five Years

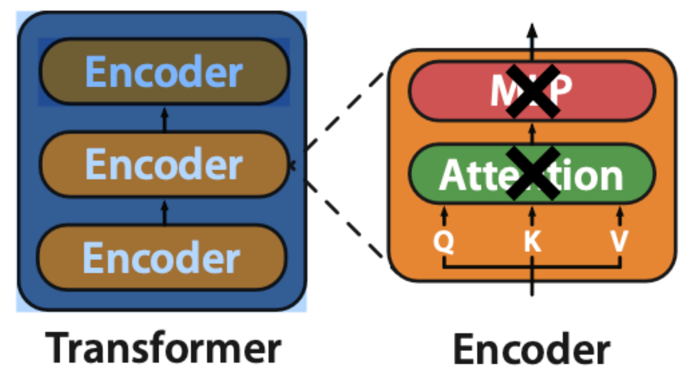
Our Current Models Scale Quadratically



Quadratic scaling in sequence length N, model dimension d

Can we find a single primitive that scales sub-quadratically to replace both?

Can we Scale Sub-Quadratically?



18

Slide credits with thanks to: Daniel Fu <danfu@cs.standford.edu>

UNIVERSITY OF TORONTO

[Aside] GEMM-Based Architecture

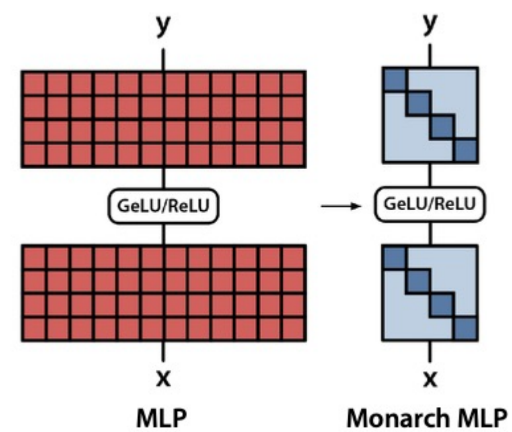
Yes… The Key is Monarch Matrices!

Artistic Rendition of
“Monarch Mixer” (DALL-E 3)

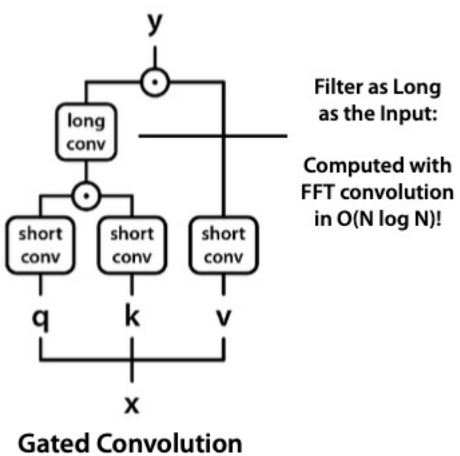
Monarch Mixer (M2): new sub-quadratic, hardware-efficient architecture

- Scales **sub-quadratically** in context length, model dimension
- Matches BERT, ViT quality with up to 50% fewer FLOPs, **faster wall-clock**
- M2-BERT for long-document retrieval

Sparsify the MLP

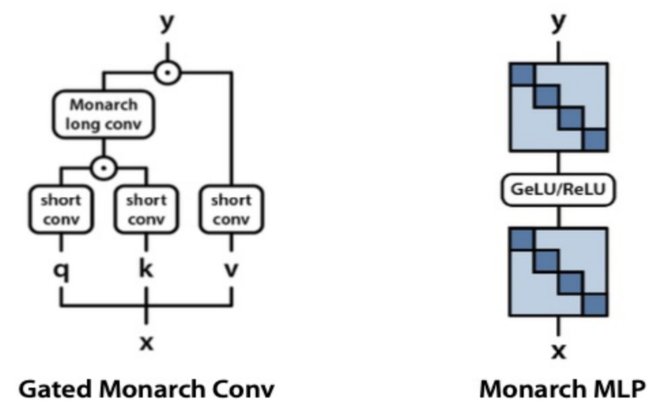


Gated Convolutions: A Match for Attention?



Model	Pile PPL (10B Tokens)
Transformer	11.9
Gated Conv	11.8

Monarch Mixer: Sub-Quadratic Mixing for Both



Slide credits with thanks to: Daniel Fu <danfu@cs.standford.edu>

TRANSFORMER BASED LANGUAGE MODELS

Architectural Variants

- The Transformers architecture underpins most SoTA LLMs of today.

- There are 3 main variations of the encoder-decoder blocks we studies for Transformers in L6

  - Encoders – e.g. models: **BERT** and BERT-variants
  - Decoders – e.g. the **GPT** series
  - Encoder-Decoder – e.g. vanilla transformer, **T5,** Flan-T5[1] (instruction tuned T5) etc.

[1] Flan-T5 - Chung, Hyung Won, et al. "Scaling instruction-finetuned language models.". link.

UNIVERSITY OF TORONTO

BERT: Bidirectional Encoder Representations from Transformers

	Rank	Name	Model	URL	Score	CoLA	SST-2	MRPC	STS-B	QQP
	1	T5 Team - Google	T5		89.7	70.8	97.1	91.9/89.2	92.5/92.1	74.6/90.4
	2	ALBERT-Team Google Language	ALBERT (Ensemble)		89.4	69.1	97.1	93.4/91.2	92.5/92.0	74.2/90.5
+	3	王玮	ALICE v2 large ensemble (Alibaba DAMO NLP)		89.0	69.2	97.1	93.6/91.5	92.7/92.3	74.4/90.7
	4	Microsoft D365 AI & UMD	FreeLB-RoBERTa (ensemble)		88.8	68.0	96.8	93.1/90.8	92.4/92.2	74.8/90.3
	5	Facebook AI	RoBERTa		88.5	67.8	96.7	92.3/89.8	92.2/91.9	74.3/90.2
	6	XLNet Team	XLNet-Large (ensemble)		88.4	67.8	96.8	93.0/90.7	91.6/91.1	74.2/90.3
+	7	Microsoft D365 AI & MSR AI	MT-DNN-ensemble		87.6	68.4	96.5	92.7/90.3	91.1/90.7	73.7/89.9
	8	GLUE Human Baselines	GLUE Human Baselines		87.1	66.4	97.8	86.3/80.8	92.7/92.6	59.5/80.4
	9	Stanford Hazy Research	Snorkel MeTaL		83.2	63.8	96.2	91.5/88.5	90.1/89.7	73.1/89.9
	10	XLM Systems	XLM (English only)		83.1	62.9	95.6	90.7/87.1	88.8/88.2	73.2/89.8

BERT

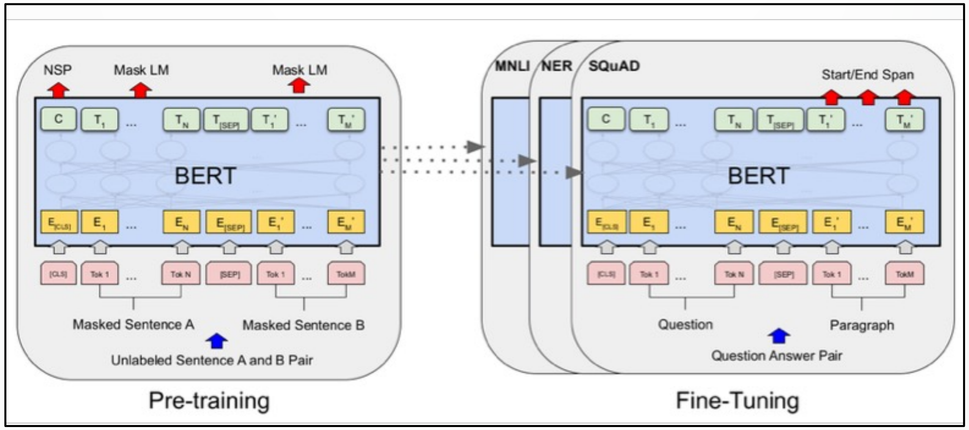
Humans

- The age of humans is over?

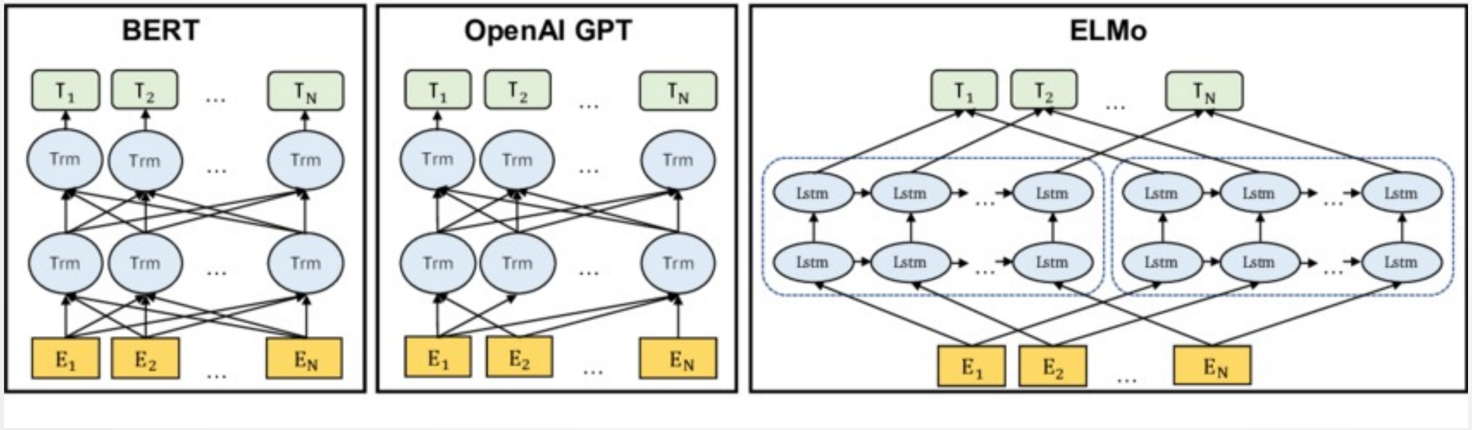
BERT: Bidirectional Encoder Representations from Transformers

Think of the **encoder** part of the transformer architecture

- Landmark, pivotal neural LM that has become an ubiquitous baseline in NLP.
- BERT is conceptually simple (multi-layer, bidirectional transformer), empirically powerful.



- Unlike predecessors (ELMo) or contemporaneous LMs (GPT), BERT is deeply **bi**directional and independent of task-specific features with unified architecture across different tasks.

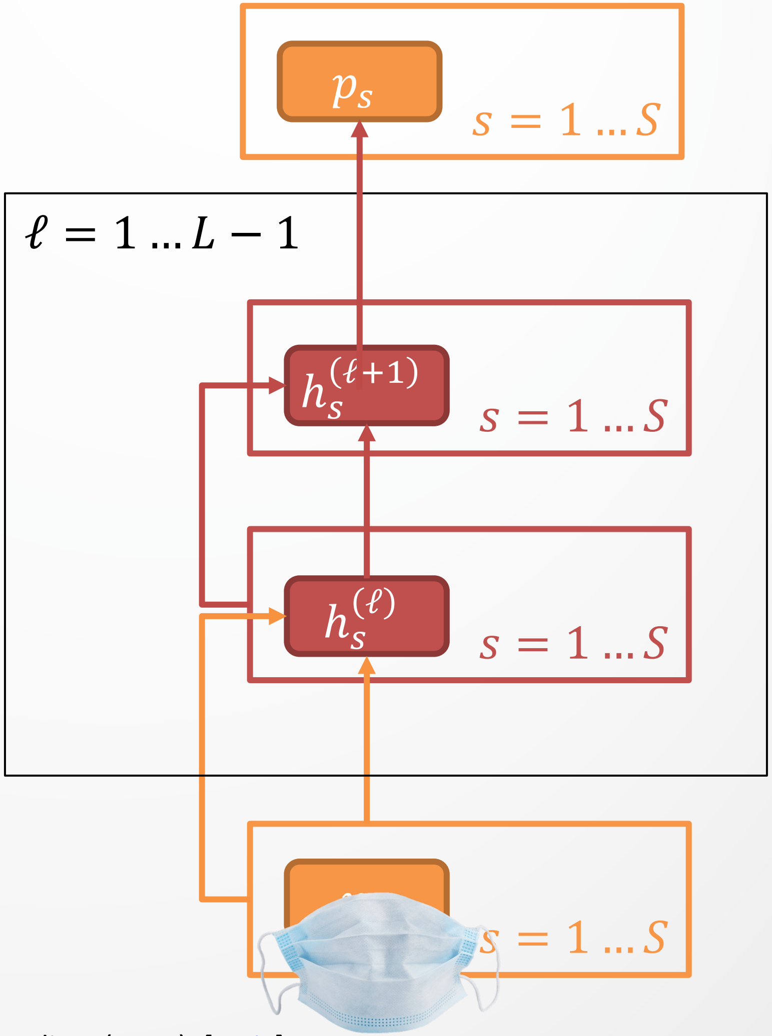


Devlin *et al.* BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. (2019). [arxiv]

Code and models: https://github.com/google-research/bert [Colab] Google AI

BERT: Bidirectional Encoder Representations from Transformers

- First, **pre-trained** on (large) unlabeled data on two unsupervised tasks/objectives:
  - Masked LM (**MLM**), and
  - Next Sentence Prediction (**NSP**)
- Then, **fine-tuned** using labeled data from downstream tasks
- Training entails feeding the final hidden vectors to an output linear layer with softmax over the possibilities (e.g. the vocabulary as in a standard LM)



Devlin *et al.* BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. (2019). [arxiv]

Code and models: https://github.com/google-research/bert [Colab] Google AI

UNIVERSITY OF TORONTO

BERT: Bidirectional Encoder Representations from Transformers

Pre-training objectives

- Masked LM (**MLM**): predict randomly masked words:

Input: The man went to the [MASK]$_1$. He bought a [MASK]$_2$ of milk .
Labels: [MASK]$_1$ = store; [MASK]$_2$ = gallon

  - 80% of the target words are masked with: [MASK]. 10% are replaced with another word, and 10% are kept as-is, to bias '*towards the observation*'.
  - *Variants*: masking granularity can be varied (word-piece, word, span) with respective quirks. E.g., masking named entities improves structured knowledge representation.

- Next sentence prediction (**NSP**): does sentence B follow A?

Sentence A = The man went to the store.	**Sentence A** = The man went to the store.
Sentence B = He bought a gallon of milk.	**Sentence B** = Penguins are flightless.
Label = IsNextSentence	**Label** = NotNextSentence

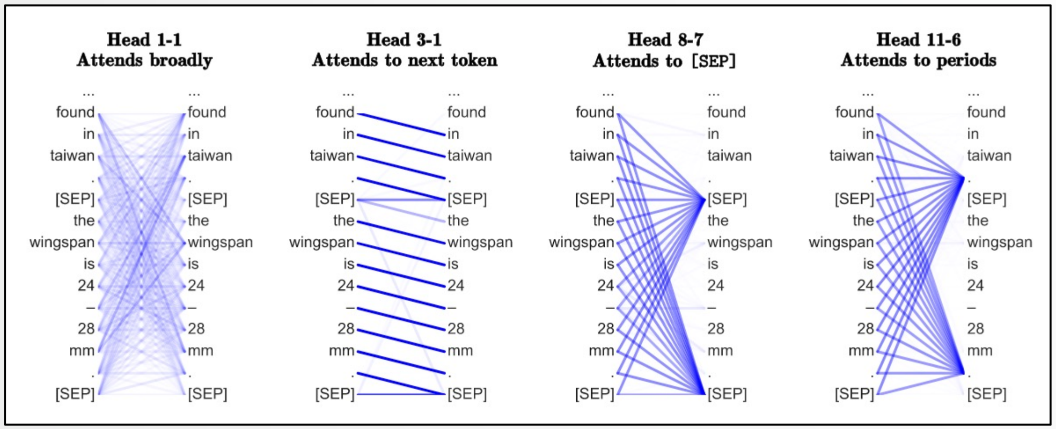
  - 50% of the time true, 50% of the time it's a random sentence.
  - Later research finds removing the NSP task does not hurt, or slightly improves performance. [2]

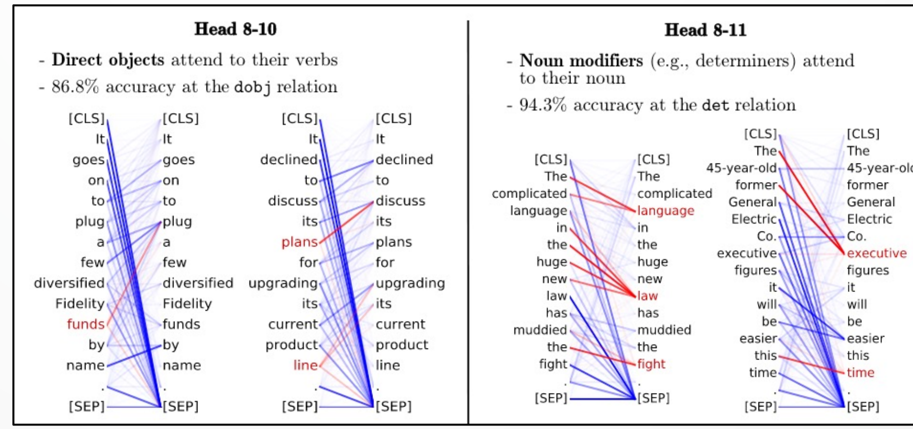
[1] Aroca-Ouellette S, Rudzicz F (2020) On Losses for Modern Language Models, EMNLP.
[2] Rogers, Anna et al. "A primer in BERTology: What we know about how BERT works." TACL(2020). link

BERT: Bidirectional Encoder Representations from Transformers

Findings from ablative studies [1,2,3]

- **Heads**: Analysis of the multi-headed attention mechanism in BERT shows attention heads exhibiting attentions on various linguistic (e.g. syntax, coreference) patterns. [1]





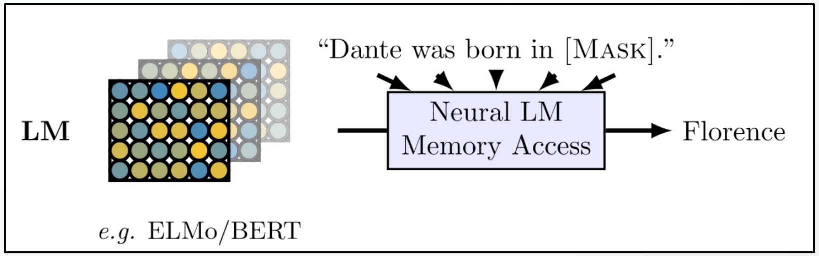
- **Layers**: linear word order and surface features captured most by lower layers. Syntactic information most prominent in middle layers. Semantic and task specific features are best captured in higher/final layers.
- Research on proposed improvements and modifications to BERT, both architectural choices (e.g. # of layers, heads) and training methods is voluminous and ongoing. Due to overall trend towards larger model sizes, systematic ablations have become prohibitively expensive.

1. Clark et al. "What does bert look at? an analysis of bert's attention." (2019). link
2. Tenney et al. "BERT rediscovers the classical NLP pipeline." (2019). link
3. Rogers, Anna et al. "A primer in BERTology: What we know about how bert works." TACL(2020). link

BERT: Bidirectional Encoder Representations from Transformers

Findings from ablative studies

- **Limitations:** BERT's possession of impressive syntactic, semantic, and world knowledge has caveats.
- **World Knowledge:**
  - BERT struggles with pragmatic inference, and role-based event knowledge.
  - It can 'guess' object affordances and properties *but* cannot reason about relationships between them. Example: it 'knows' people can walk into houses, houses are big, but cannot infer that houses are bigger than people.
- **Semantic Knowledge:**
  - Struggles with representations of numbers.
  - Surprisingly brittle to *named entity* replacements: e.g. 85% drop in performance in coreference task with names replaced.



- **Syntactic Knowledge:**
  - Does not 'understand' negations and is insensitive to malformed input.
  - Findings suggest that either its syntactic knowledge is incomplete, or not dependent on it for solving its tasks.

BERT: Bidirectional Encoder Representations from Transformers

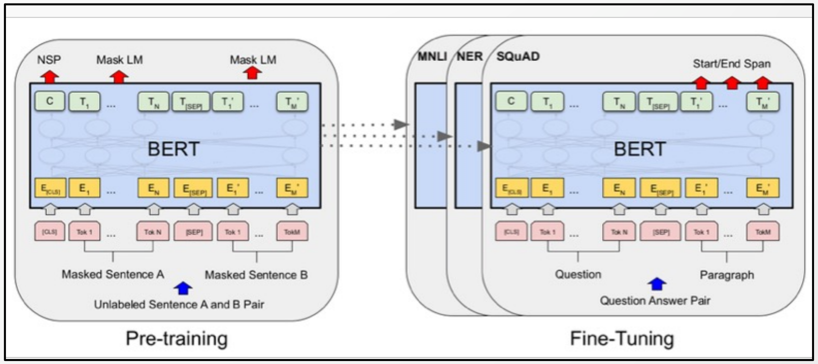
Why BERT matters/mattered?

System	MNLI-(m/mm)	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	**Average**
	392k	363k	108k	67k	8.5k	5.7k	3.5k	2.5k	-
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	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**92.7**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**82.1**

System	Dev	Test
ESIM+GloVe	51.9	52.7
ESIM+ELMo	59.1	59.2
OpenAI GPT	-	78.0
$\text{BERT}_{\text{BASE}}$	81.6	-
$\text{BERT}_{\text{LARGE}}$	**86.6**	**86.3**
Human (expert)†	-	85.0
Human (5 annotations)†	-	88.0

System	Dev EM	Dev F1	Test EM	Test F1
Top Leaderboard Systems (Dec 10th, 2018)				
Human	-	-	82.3	91.2
#1 Ensemble - nlnet	-	-	86.0	91.7
#2 Ensemble - QANet	-	-	84.5	90.5
Published				
BiDAF+ELMo (Single)	-	85.6	-	85.8
R.M. Reader (Ensemble)	81.2	87.9	82.3	88.5
Ours				
$\text{BERT}_{\text{BASE}}$ (Single)	80.8	88.5	-	-
$\text{BERT}_{\text{LARGE}}$ (Single)	84.1	90.9	-	-
$\text{BERT}_{\text{LARGE}}$ (Ensemble)	85.8	91.8	-	-
$\text{BERT}_{\text{LARGE}}$ (Sgl.+TriviaQA)	**84.2**	**91.1**	**85.1**	**91.8**
$\text{BERT}_{\text{LARGE}}$ (Ens.+TriviaQA)	**86.2**	**92.2**	**87.4**	**93.2**

- **Performance and versatility:**
  - Impressive performance – not only beating (then) SOTA peers, but humans too!
  - Popularized the pipeline of **'pretraining' -> 'fine-tuning'** NLMs
  - Fine-tuning (pre-trained) BERT on downstream tasks led to new SOTA results on a broad range of NLP tasks



UNIVERSITY OF TORONTO

BERT: Bidirectional Encoder Representations from Transformers

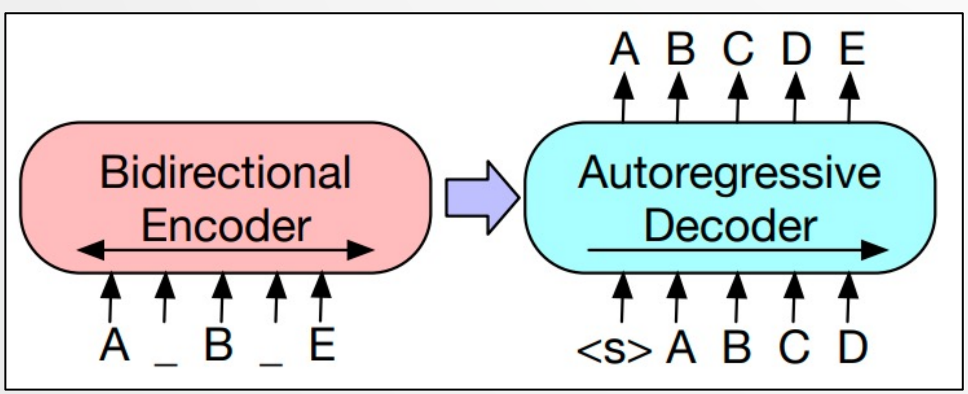
Use BERT for everything?

- BERT has an encoder-only architecture
- While the preceding results are great, but pretrained encoders aren't great for everything
- For example, tasks involving generating sequences auto-regressively (one token at a time) - like our machine translation task!
- Solution: use a (pretrained) decoder in conjunction

UNIVERSITY OF TORONTO

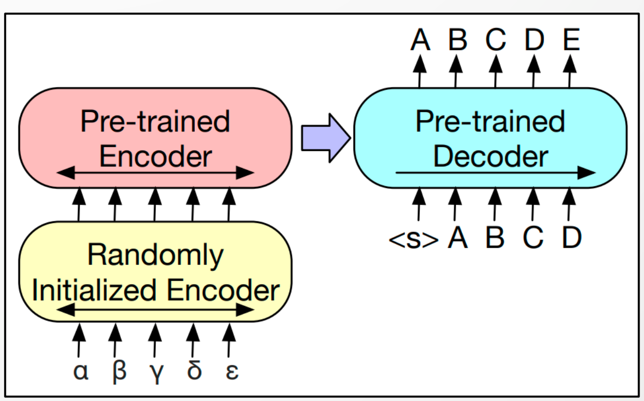
Aside – BERT → BART → NMT

- Explosion of variants to BERT
- Pretrained BERT language model used to re-score/fine-tune downstream NLP tasks
- BART (Lewis *et al*, 2020) adds the decoder back to BERT, keeping the BERT objective
- Add some source language layers on top to train for NMT



BART→

BART for NMT→



Lewis, Mike, et al. "Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension." (2019). link.

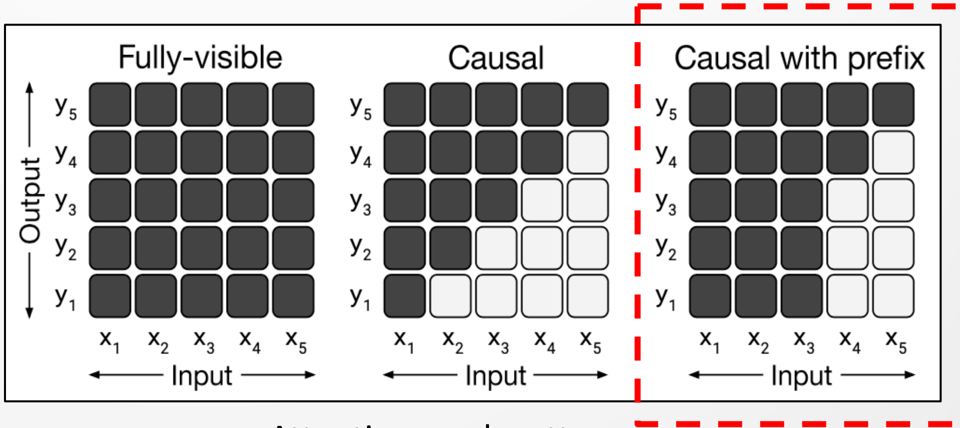
T5: Text-to-Text Transfer Transformer

A *refined* Transformer updated with better methodologies

- T5 is an unified framework that casts all NLP problems into a '*text-to-text*' format.
- Architecturally (almost) identical to the original Transformer (Vaswani et al., 2017).
- Draws from a systematic study comparing pre-training objectives, architectures, unlabeled data sets, transfer approaches, and other factors on dozens of language understanding tasks.
- Introduces and uses a new curated **dataset**: "*Colossal Clean Crawled Corpus*" (**C4**) for training.

Distinguishing features:

- Consistent, task-invariant MLE training objective.
- Self-attention "mask" with prefix.
- Unsupervised "denoising" training objectives: ***span corruption*** (conceptually same to MLM, mask 'spans' instead of words).



Attention mask patterns

1. Raffel et al. "Exploring the limits of transfer learning with a unified text-to-text transformer." (2020). link

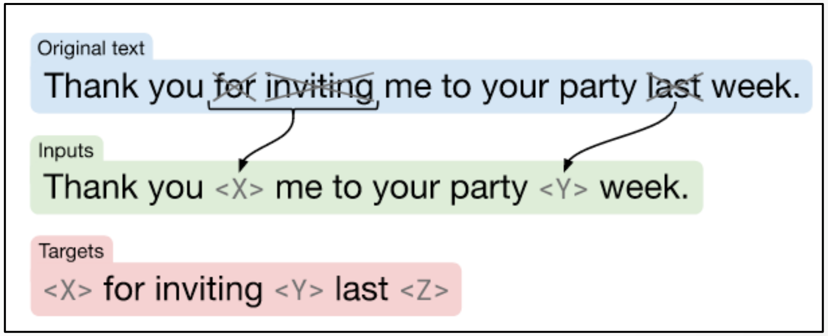
T5: Text-to-Text Transfer Transformer

Example Task: English to German (En-De) translation:

Input sentence: "*That is good.*"
Target: "*Das ist gut.*"

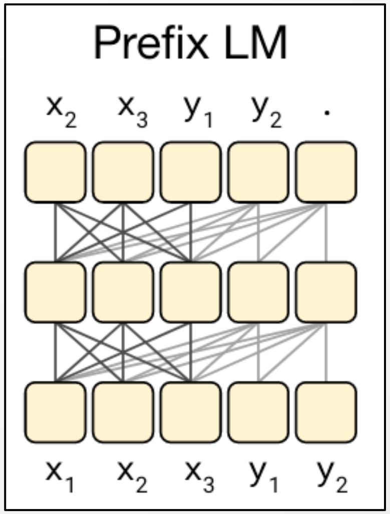
- **Training:** task specification is imbued by prepending task prefix to the input sequence. Model trained on next sequence prediction over the concatenated input sequence:

  "*translate En-De: That is good. Das ist gut.*"



Input/Output format for training denoising objective

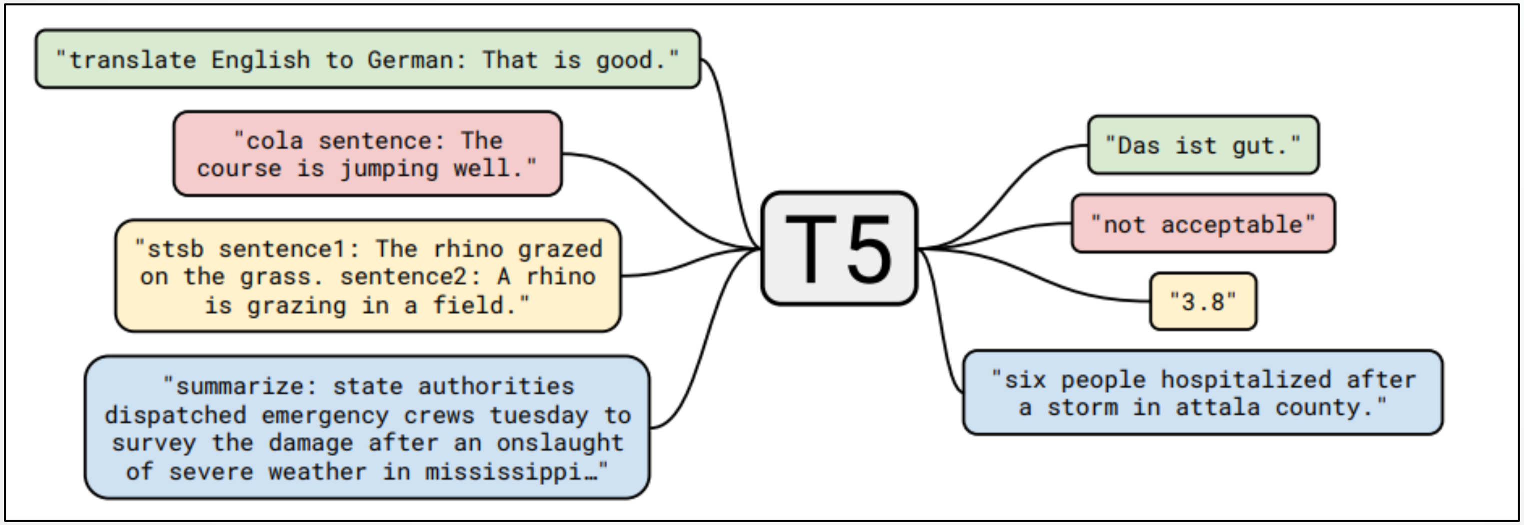
- For prediction, the model is fed **prefix**:
  - "*translate En-De: That is good. target:*"
- For **classification** tasks, the model predicts a single word corresponding to the target label.
- E.g. MNLI task of entailment prediction:
  - "*mnli premise: I hate pigeons. hypothesis: I am hostile to pigeons. entailment.* "
- Model predicts label: {"entailment", "neutral", "contradiction"}.



T5: Text-to-Text Transfer Transformer

Why T5 matters?

- Unifying diverse NLP problems as one (‘*text-to-text*’) format is a **really cool idea**.
- This allows us to use the **same** model, loss function, hyperparameters etc. across a diverse set of tasks



- Remarkable transfer learning capabilities: T5 can be finetuned to answer a wide range of (open-domain dataset NQ, WQ, TQA) questions, retrieving knowledge from its parameters

UNIVERSITY OF TORONTO

On that note – A2 BART Analysis

- **Code Demo** HuggingFace [t5_small|BART-base] NMT trained on Hansard (Fr-En)
- https://huggingface.co/docs/transformers/model_doc/bart
- https://huggingface.co/raeidsaqur/bart-base
- https://huggingface.co/raeidsaqur/mt_fr2en_hansard_t5-small

- Optional: bonus pointers

Lewis, Mike, et al. "Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension." (2019). link.

GPT SERIES

GPT: Generative Pretrained Transformers

Open AI GPT-series of models – uses multi-layer decoder only blocks

- Open AI GPT papers: GPT (2018)[1], GPT-2 (2019)[2], GPT-3 (2020)[3]

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

Language Models are Unsupervised Multitask Learners

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

Language Models are Few-Shot Learners

OpenAI

- Architecturally (almost) identical – each scales (params, data) on predecessor

- Pretraining objective is classic '**language modeling**', to maximize the likelihood:

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

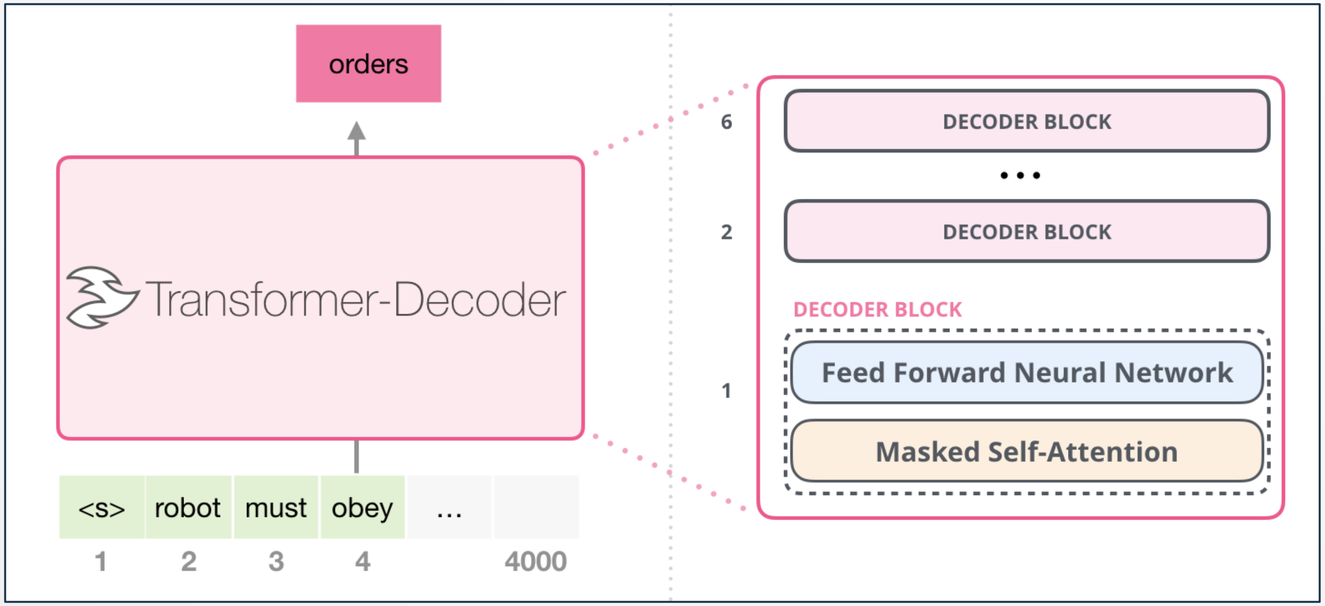
- Specifically, given an unsupervised corpus of tokens $\boldsymbol{\mu} = \{\mu_1, \ldots, \mu_n\}$, where k is context window, P is modelled using a neural network with parameters θ.

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

1. Radford, Alec, et al. "Improving language understanding by generative pre-training." (2018)
2. Radford, Alec, et al. "Language models are unsupervised multitask learners." OpenAI Blog 1.8 (2019)
3. Brown, Tom B., et al. "Language models are few-shot learners." arXiv preprint arXiv:2005.14165 (2020)

GPT: Generative Pretrained Transformers

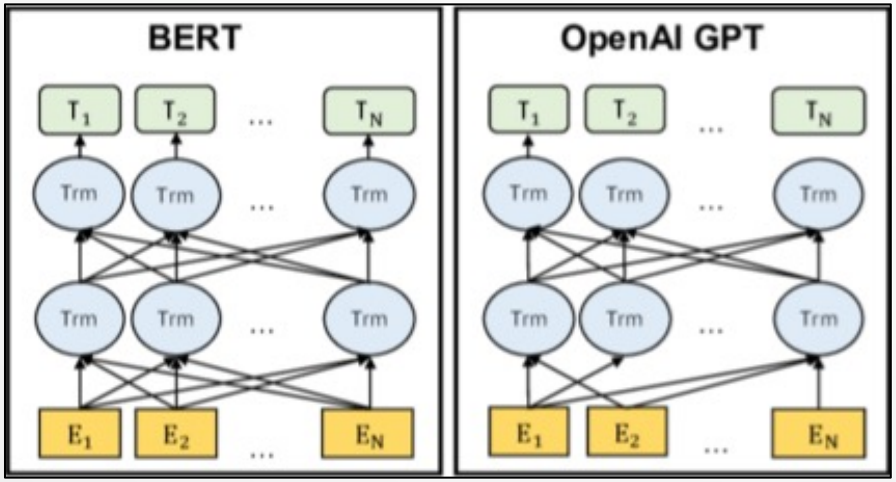
- Distinguishing features:
  - Uses multi-layer transformer **decoder only blocks**
  - Auto-regressive generative model, does not see the future (no bi-directional awareness)
  - Like traditional LMs, outputs one token at a time



1. Radford, Alec, et al. "Improving language understanding by generative pre-training." (2018)
2. Radford, Alec, et al. "Language models are unsupervised multitask learners." OpenAI Blog 1.8 (2019)
3. Brown, Tom B., et al. "Language models are few-shot learners." arXiv preprint arXiv:2005.14165 (2020)

Key architectural differences

- GPT vs. BERT-variants:
  - GPT uses ‘transformer’ blocks as *decoders*, and BERT as *encoders*.
  - Underlying (block level) ideology is same
  - GPT (later Transformer XL, XLNet) is an **autoregressive** model, BERT is not
    - At the cost of auto-regression, BERT has bi-directional context awareness.
  - GPT, like traditional LMs, outputs (predicts) one token at a time.
- Compare with T5, BART that uses encoder-decoder



[1] Radford, Alec, et al. "Improving language understanding by generative pre-training." (2018).

UNIVERSITY OF TORONTO

GPT-3 LLMs take off …

- Increasingly convincing results permeating into the public sphere and zeitgeist
- In-context learning:

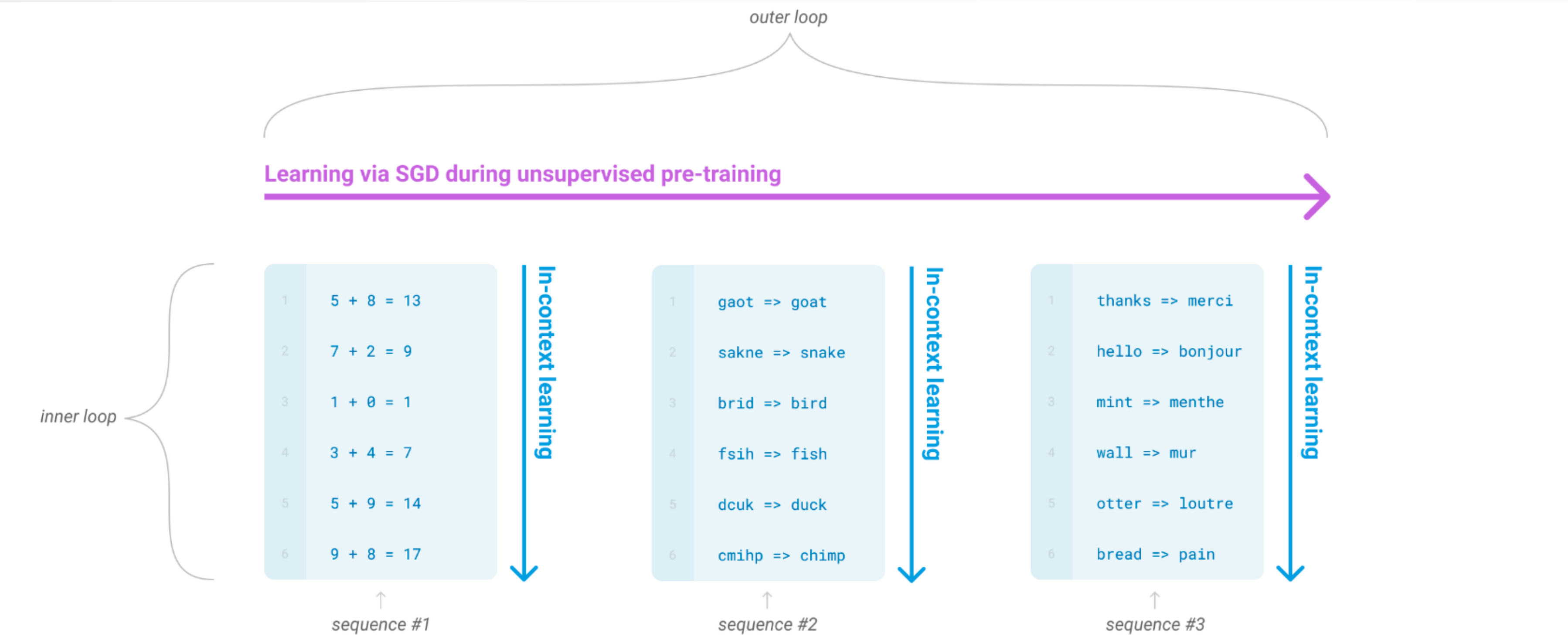


Figure 1.1: Language model meta-learning. During unsupervised pre-training, a language model develops a broad set of skills and pattern recognition abilities. It then uses these abilities at inference time to rapidly adapt to or recognize the desired task. We use the term "in-context learning" to describe the inner loop of this process, which occurs within

1. Brown, Tom B., et al. "Language models are few-shot learners." arXiv preprint arXiv:2005.14165 (2020)

GPT-3: In context learning + … prompting

- The notion of ‘**prompting**’ begins to emerge …

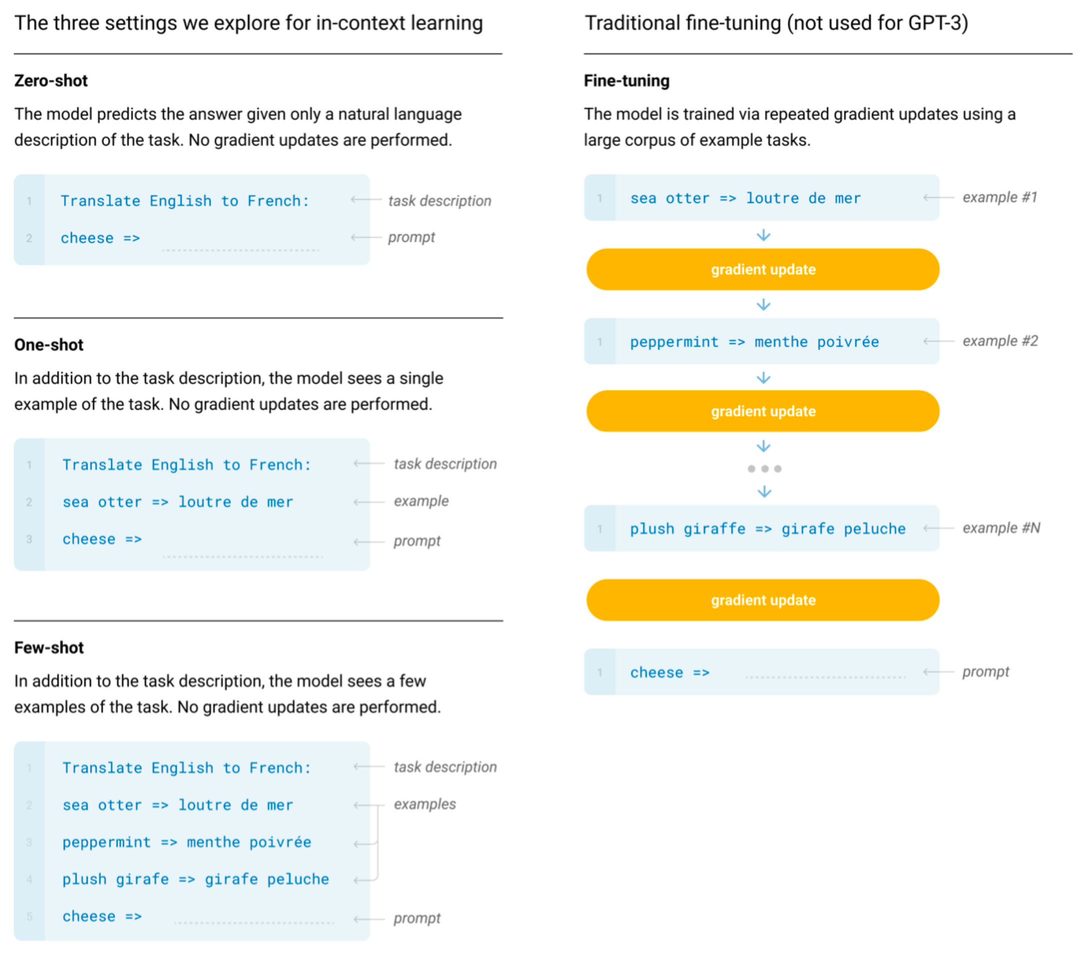


Figure 2.1: Zero-shot, one-shot and few-shot, contrasted with traditional fine-tuning. The panels above show

1. Brown, Tom B., et al. "Language models are few-shot learners." arXiv preprint arXiv:2005.14165 (2020)

UNIVERSITY OF TORONTO

GPT-3: In context learning + … prompting

- The notion of '**prompting**' begins to emerge …

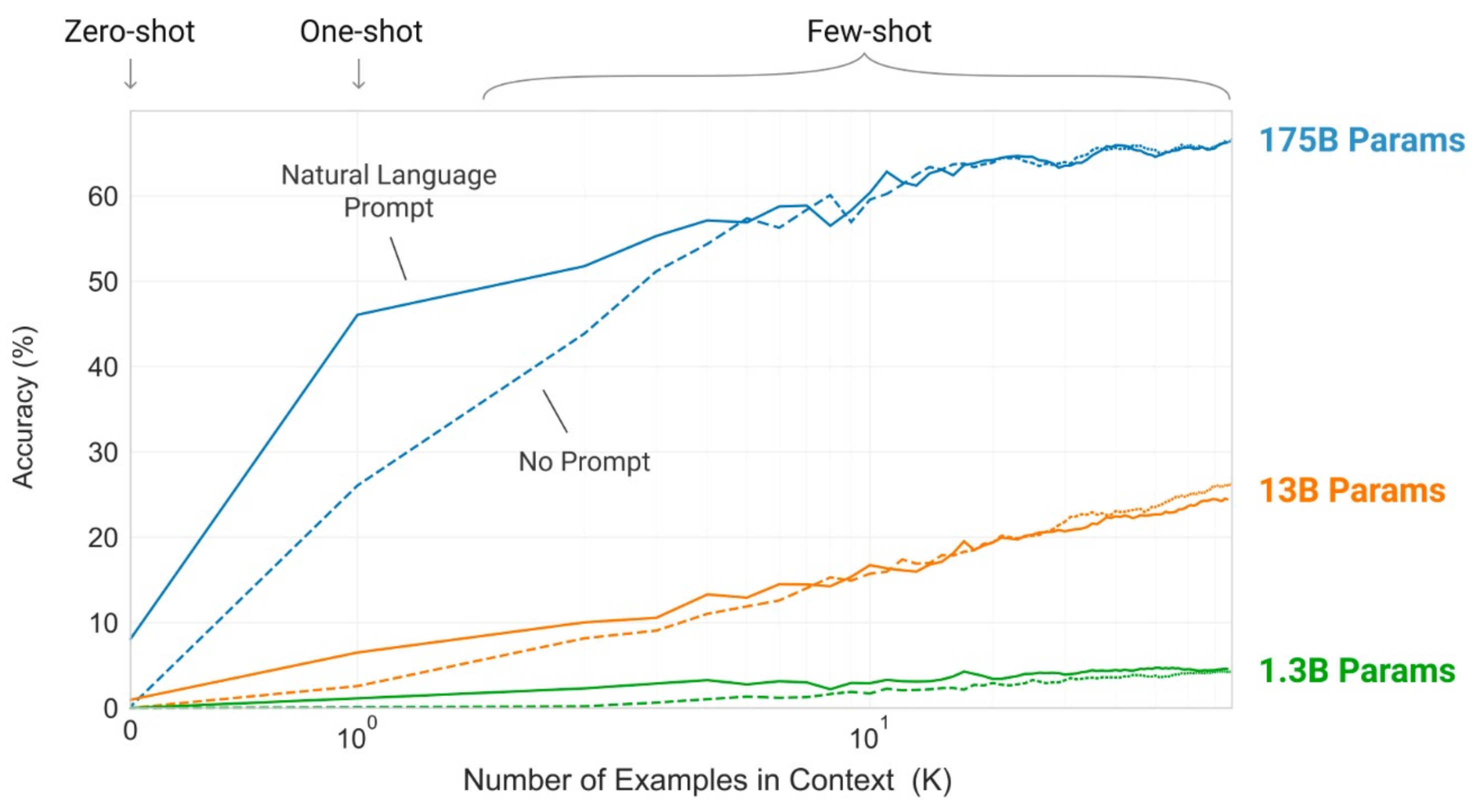


Figure 1.2: Larger models make increasingly efficient use of in-context information. We show in-context learning

1. Brown, Tom B., et al. "Language models are few-shot learners." arXiv preprint (2020)

GPT-3: Models

- Initialization, pre-normalization (inherited from GPT2).
- 8 model sizes trained to study effect of model size

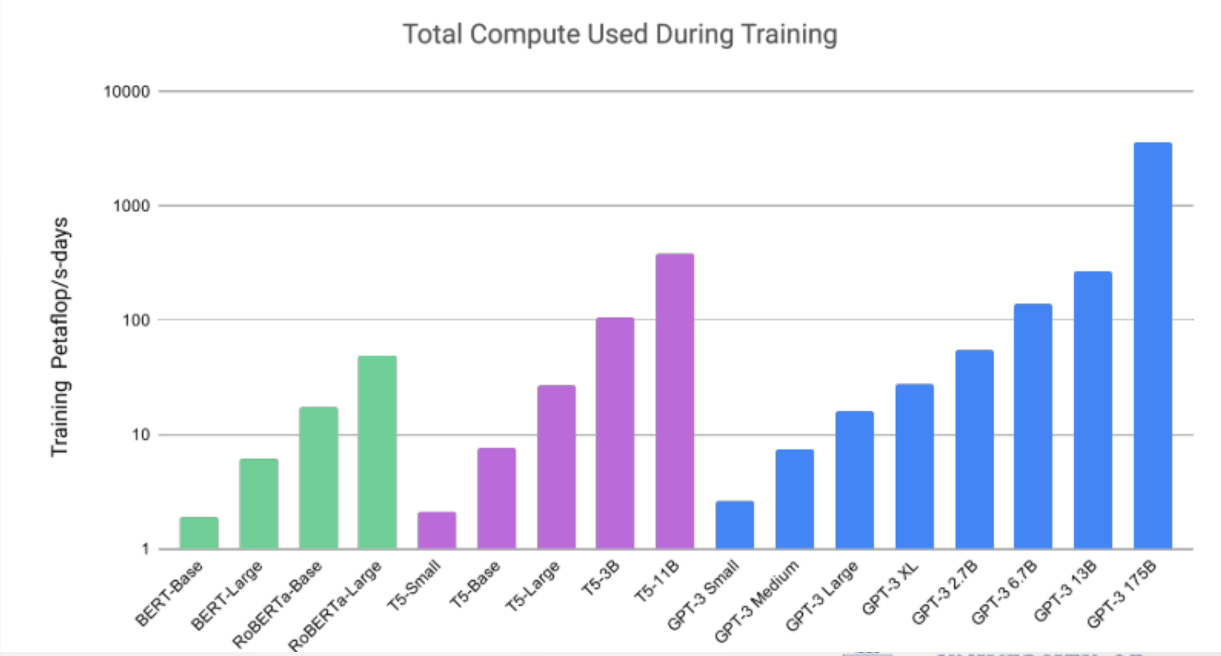
Model Name	n_{params}	n_{layers}	d_{model}	n_{heads}	d_{head}	Batch Size	Learning Rate
GPT-3 Small	125M	12	768	12	64	0.5M	6.0×10^{-4}
GPT-3 Medium	350M	24	1024	16	64	0.5M	3.0×10^{-4}
GPT-3 Large	760M	24	1536	16	96	0.5M	2.5×10^{-4}
GPT-3 XL	1.3B	24	2048	24	128	1M	2.0×10^{-4}
GPT-3 2.7B	2.7B	32	2560	32	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.

GPT-3: Training Datasets

- Datasets
  - Common Crawl dataset: nearly a trillion words, with quality curation.
  - Added: WebText, Books1, 2 and English Wikipedia
- Train time

Dataset	Quantity (tokens)	Weight in training mix
Common Crawl (filtered)	410 billion	60%
WebText2	19 billion	22%
Books1	12 billion	8%
Books2	55 billion	8%
Wikipedia	3 billion	3%



UNIVERSITY OF TORONTO

GPT-3: Results

- TL;DR: unprecedently impressive results across task domains
  - Performance (e.g. world knowledge) **increases with size**
- Datasets grouped to 9 categories of downstream tasks
  - Examples: language modeling, QA, translation, Winograd, common-sense reasoning, reading comprehension, NLI etc.
  - Read the paper for details

Setting	PTB
SOTA (Zero-Shot)	35.8[a]
GPT-3 Zero-Shot	**20.5**

Table 3.1: Zero-shot results on PTB language modeling dataset. Many other common language modeling datasets are omitted because they are derived from Wikipedia or other sources which are included in GPT-3's training data. [a][RWC+19]

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]

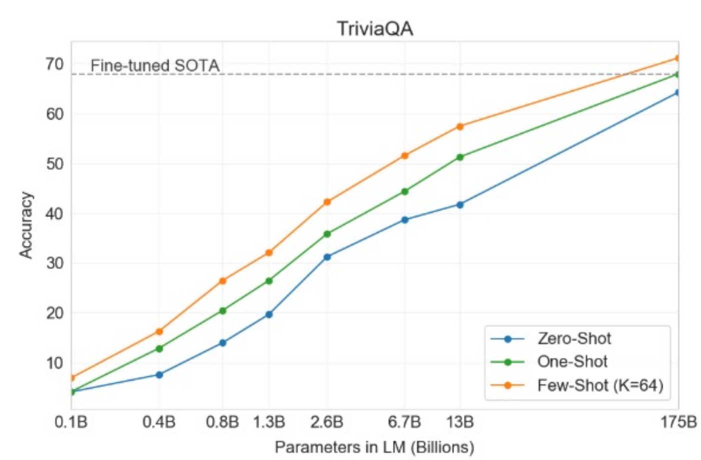


Figure 3.3: On TriviaQA GPT3's performance grows smoothly with model size, suggesting that language models continue to absorb knowledge as their capacity increases. One-shot and few-shot performance make significant gains over zero-shot behavior, matching and exceeding the performance of the SOTA fine-tuned open-domain model, RAG [LPP+20]

UNIVERSITY OF TORONTO

DIFFERENT DIRECTIONS

Token free models

- Unlike the ubiquitous pre-trained LMs that operate on sequences of tokens corresponding to word or sub-word units, *token free models*:

  - Operate on raw text (bytes or characters) **directly.**
  - Removes necessity for (error-prone, complex) text preprocessing pipelines.

  - Con: raw sequences significantly **longer than token sequences**, increases computational complexity. (Reminder: *'attention' costs are quadratic to the length of input sequence*)

1. Clark et al. "**CANINE**: Pre-training an efficient tokenization-free encoder for language representation." (2021). link
2. Xue et al. "**ByT5**: Towards a token-free future with pre-trained byte-to-byte models." (2022). link

Token free models

- Pitfalls of explicit (word, sub-word) tokenization:
  - Need for large language dependent (fixed) vocabulary mapping matrices.
  - Applies hand-engineered, costly, language-specific string tokenization/segmentation algorithms (e.g. BPE, word-piece, sentence-piece) requiring linguistic expertise.
  - Heuristic string-splitting, however nuanced, cannot capture full breadth of linguistic phenomena, (e.g. morphologically distant agglutinative, non-concatenative languages). Other examples include languages without white-space (Thai, Chinese), or that uses punctuation as letters (Hawaiian, Twi). *Fine-tuning* tokenization needs to match *pretraining* tokenization methods.
  - Brittle to noise, corruption of input (typos, adversarial manipulations). Corrupted tokens lose vocabulary coverage.

1. Clark et al. "**CANINE**: Pre-training an efficient tokenization-free encoder for language representation." (2021). **link**
2. Xue et al. "**ByT5**: Towards a token-free future with pre-trained byte-to-byte models." (2022). **link**

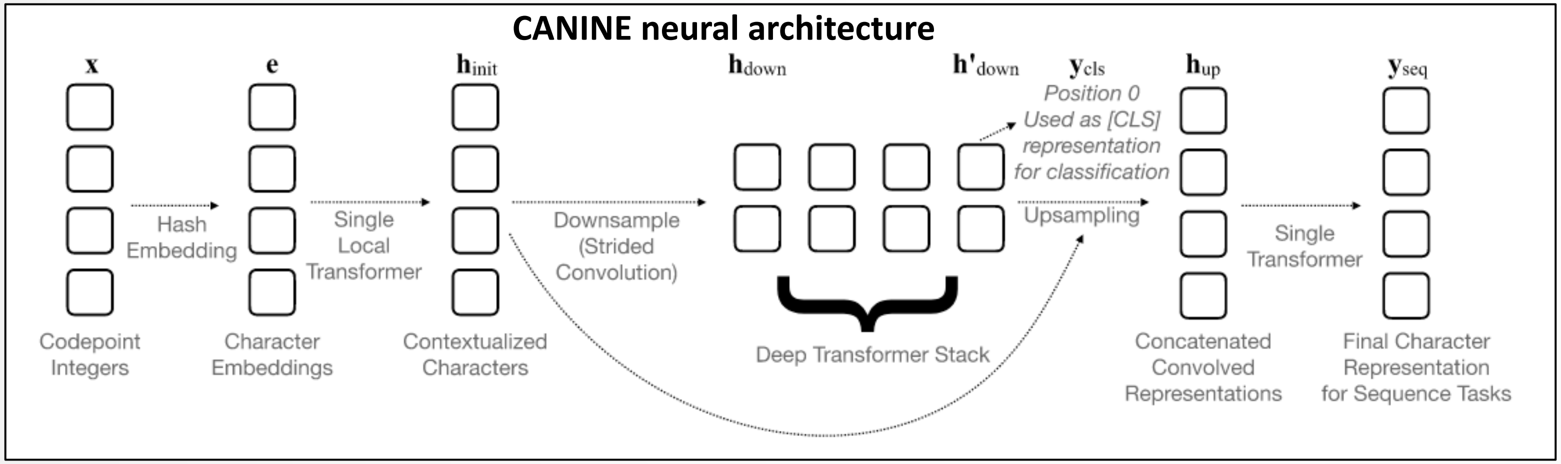
[Aside] Token free models - CANINE

CANINE: Character Architecture with No tokenization In Neural Encoders.

- CANINE is a large language encoder with a deep transformer stack at its core.
- Inputs to the model are sequences of Unicode characters. 143,698 Unicode codepoints assigned to characters covers 154 scripts and over 900 languages!
- To avoid slowdown from increasing sequence length, it uses **stride convolutions to down-sample** input sequences to a shorter length, before the deep transformer stack to encode context.
- Three primary components:
  - Vocab free embedding technique;
  - Character-level model (CLM) with efficiency measures (up/down sampling of sequences); and
  - Perform **unsupervised** masked LM (MLM) pretraining on the CLM using variants:
    - Autoregressive character prediction
    - Subword prediction

Clark et al. "**CANINE**: Pre-training an efficient tokenization-free encoder for language representation." (2022).

[Aside] Token free models - CANINE



- The overall functional composition form uses [UP|DOWN]-sampling, and primary encoder:

  $Y_{seq} \leftarrow \textsc{Up}(\textsc{Encode}(\textsc{Down}\ (e))$ where $e \in \mathbb{R}^{n \times d}$ is an input characters sequence, and
  $Y_{seq} \in \mathbb{R}^{n \times d}$ is output of sequence predictions

- **Down-sampling**: $h_{init} \leftarrow \textsc{LocalTransformer}(e)$; $h_{down} \leftarrow \textsc{StridedConv}(h_{init}, r)$

  where $h_{down} \in \mathbb{R}^{m \times d}$ and $m = \frac{n}{r}$ is the number of downsampled positions

- **Up-sampling**: prediction require model's output layer sequence length to match input's length

  $h_{up} \leftarrow \textsc{Conv}(h_{init} \oplus h'_{down}, w)$; $y_{seq} \leftarrow \textsc{Transformer}\ (h_{up})$

  where $\oplus$ is vector concatenation, CONV projects $\mathbb{R}^{n \times 2d}$ back to $\mathbb{R}^{n \times d}$ across a window of w characters. Applying a final transformer layer yields a final sequence representation: $Y_{seq} \in \mathbb{R}^{n \times d}$

UNIVERSITY OF TORONTO