

# CS480/680: Introduction to Machine Learning

## Lec 10: Attention

Yaoliang Yu

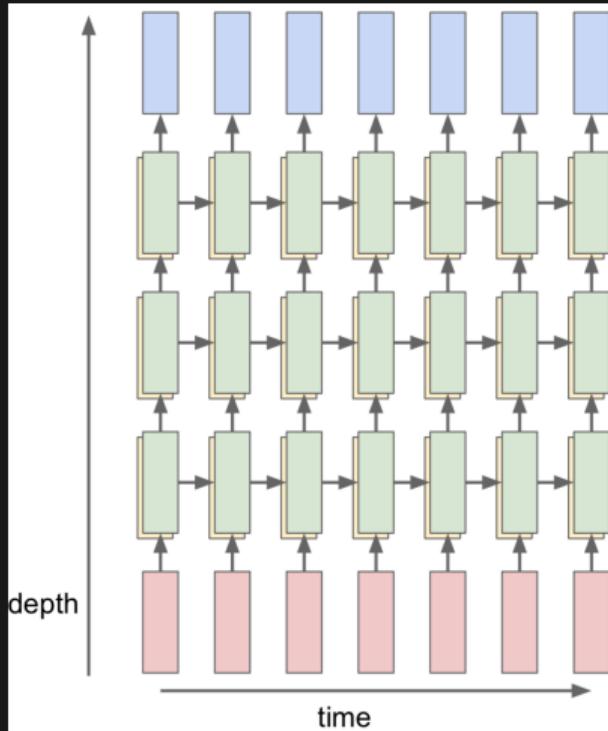


UNIVERSITY OF  
**WATERLOO**

FACULTY OF MATHEMATICS  
**DAVID R. CHERITON SCHOOL  
OF COMPUTER SCIENCE**

June 10, 2024

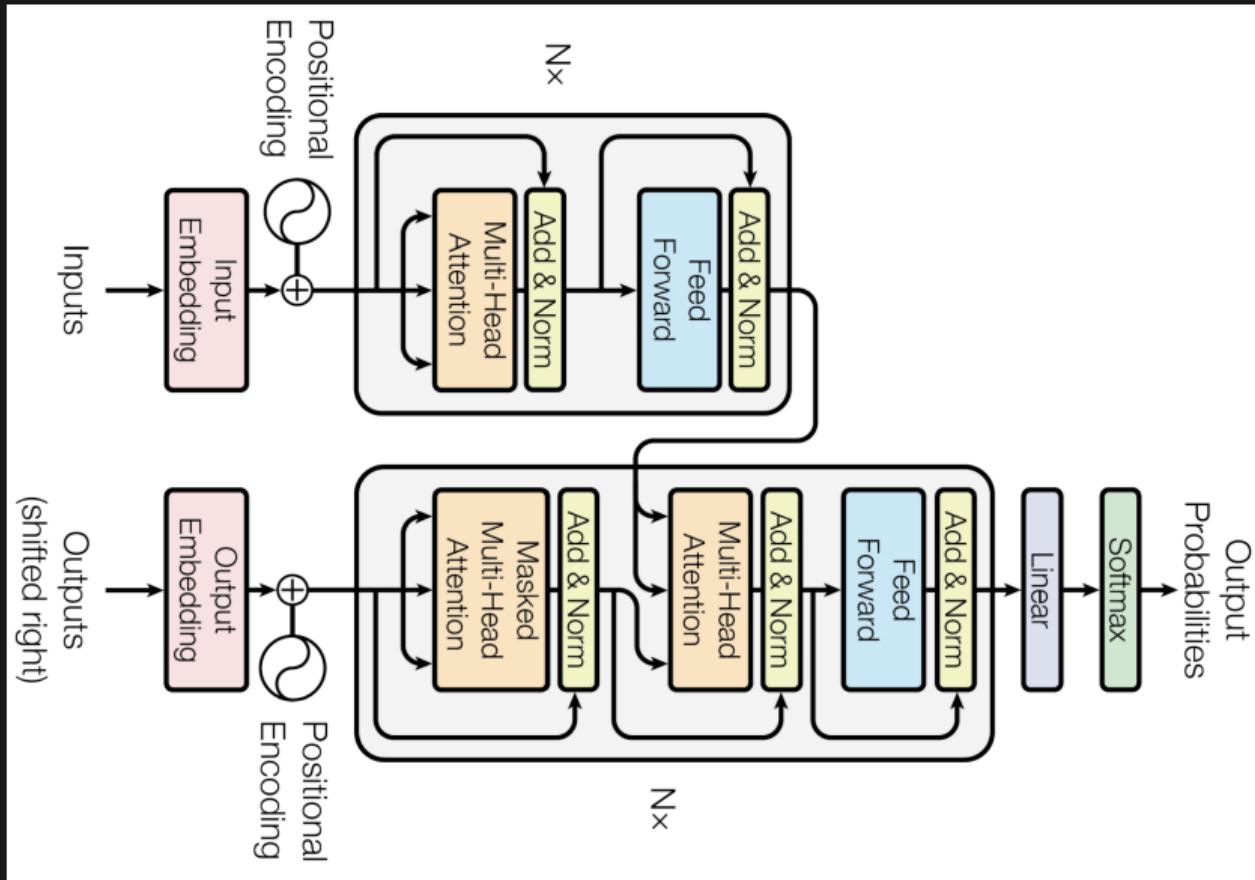
# Sequential Nature of RNN



- Sequential in both time and depth: slow and expensive!

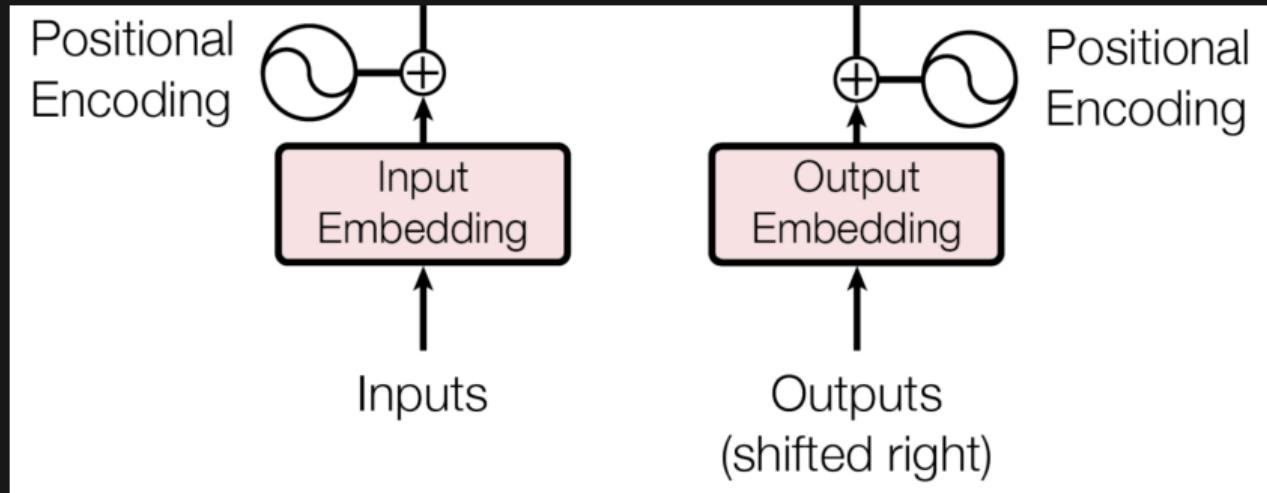






A. Vaswani et al. "Attention is All you Need". In: *Advances in Neural Information Processing Systems 30*. 2017, pp. 5998–6008.

# Input and Output



- Input sequence  $X = (\mathbf{x}_1, \dots, \mathbf{x}_m)^\top$ ,  $\mathbf{x}_t \in \mathbb{R}^p$  one-hot
- Output sequence  $Y = (\mathbf{y}_1, \dots, \mathbf{y}_l)^\top$ ,  $\mathbf{y}_t \in \mathbb{R}^p$  one-hot
- Embedding:  $XW^e$  and  $YW^e$ , where  $W^e \in \mathbb{R}^{p \times d}$ ,  $d = 512$
- (Additive) positional encoding:  $W^p \in \mathbb{R}^{\text{len} \times d}$ 
  - $w_{t,2i}^p = \sin(t/10000^{2i/d})$ ,  $w_{t,2i+1}^p = \cos(t/10000^{2i/d})$ ,  $i = 0, \dots, \frac{d}{2} - 1$

# Self-Attention

---

$$V \leftarrow VW^v$$

$$V \leftarrow \text{softmax}(VV^\top/\lambda) \cdot V, \quad e.g., \quad \lambda = \sqrt{d}$$

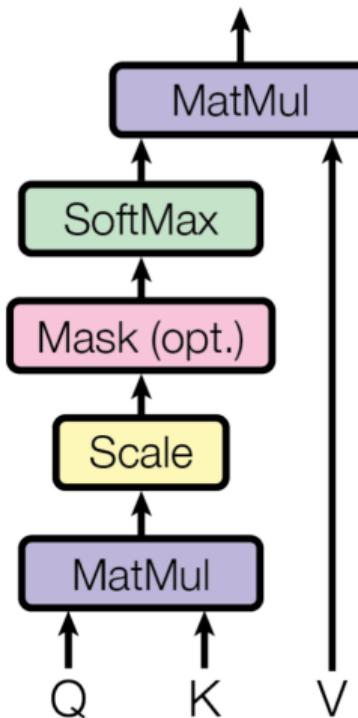
- Highly parallelizable matrix product replaces recurrence
- Each output is a **convex** combination of **all** inputs
- Dot product  $\langle \mathbf{v}_{i:}, \mathbf{v}_{j:} \rangle$  measures similarity: more similar, more contribution
- Softmax is **dense**: every output attends to every input

# More Generally

$$[Q, K, V] \leftarrow [QW^q, KW^k, VW^v]$$
$$V \leftarrow \text{softmax}(QK^\top/\lambda) \cdot V$$

- $Q$ : query, linear transformed by  $W^q$
- $K$ : key, linear transformed by  $W^k$
- $V$ : value, linear transformed by  $W^v$
- Similarity by dot product between query  $Q$  and key  $K$
- Output is convex combination of input
- Self-attention becomes a special case:  
 $[Q, K, V] \leftarrow [VW^v, VW^v, VW^v]$

## Scaled Dot-Product Attention



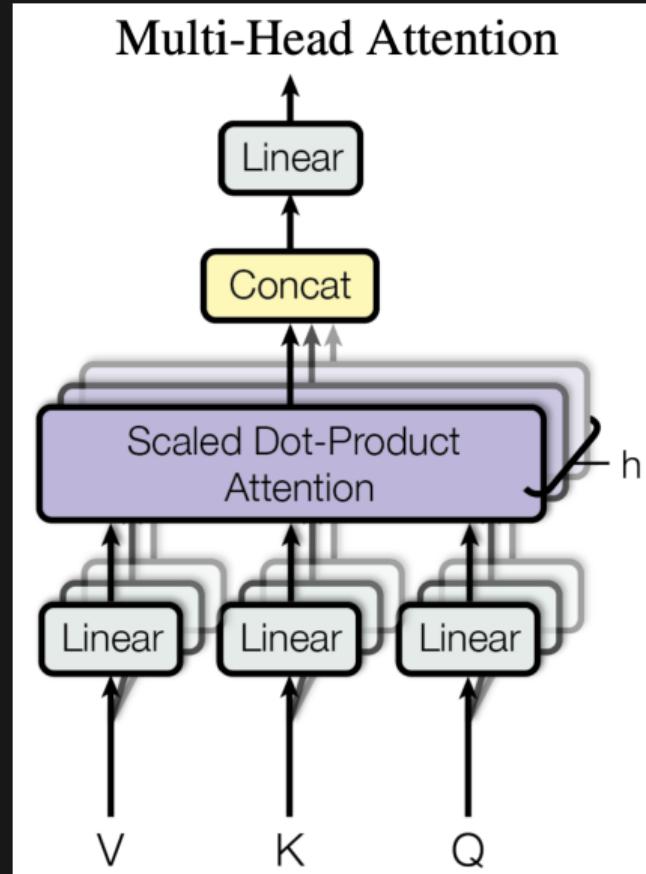
# Multi-head

- $h = 8$  or  $h = 12$
- keep  $d = 512$

---

```
1 for  $i = 1, \dots, h$  do
    // linear transformation
     $[Q, K, V] \leftarrow [QW_i^q, KW_i^k, VW_i^v]$ 
    // convex combination
     $V_i \leftarrow \text{softmax}(QK^\top / \lambda)V$ 
    // concatenation & linear trans
4  $V \leftarrow [V_1, \dots, V_h]W$ 
```

---

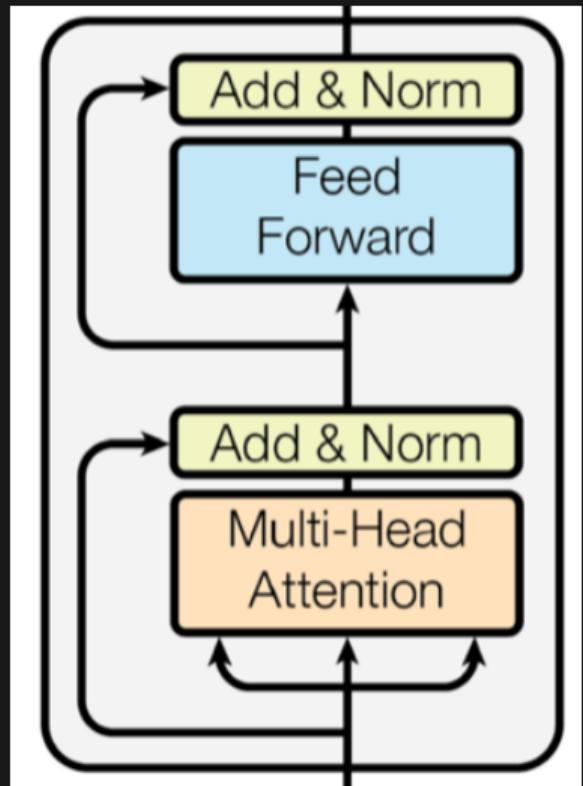


# Position-wise Feed-forward

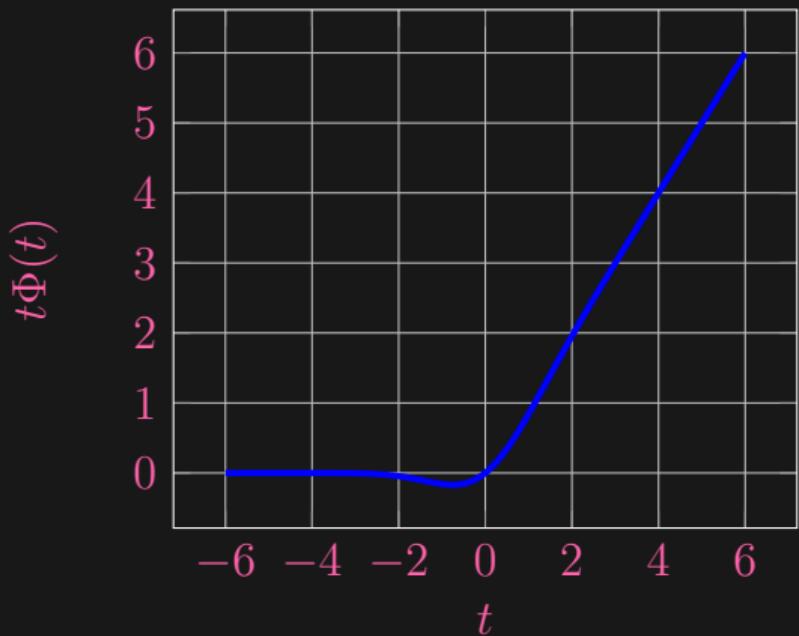
- Let  $\mathbf{V} \in \mathbb{R}^{\text{len} \times d}$  be the values (at some layer), arranged row-wise:

$$\text{FFN}(\mathbf{V}) = \sigma(\mathbf{V}\mathbf{W}_1)\mathbf{W}_2$$

- two-layer network
- $\sigma(t) = t\Phi(t)$ : Gaussian Error Linear Unit (GELU)
- Weights  $\mathbf{W}_1 \in \mathbb{R}^{d \times 4d}$  and  $\mathbf{W}_2 \in \mathbb{R}^{4d \times d}$  are shared among different positions but change from layer to layer
- Residual connections and layer-wise normalization



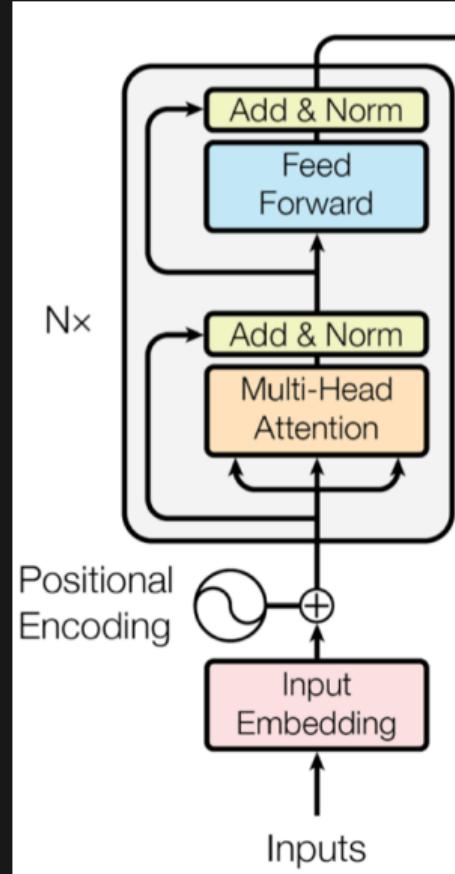
$t\Phi(t)$



$[t\Phi(t)]'$



# Transformer Encoder



# Understanding Attention

- 1x1 conv: consider  $d$  filters  $W^v = [\mathbf{w}_1, \dots, \mathbf{w}_d]$  so that

$$u_{ik} \leftarrow \langle \mathbf{v}_{i:}, \mathbf{w}_k \rangle$$

- Global weighted average pooling:

$$\mathbf{v}_{j:} \leftarrow \sum_i p_{ji} \cdot \mathbf{u}_{i:}, \quad \text{where } p_{ji} \propto \exp(\langle \mathbf{u}_{j:}, \mathbf{u}_{i:} \rangle / \lambda)$$

- 1x1 conv: consider  $4d$  filters  $W = [\mathbf{w}_1, \dots, \mathbf{w}_{4d}]$  so that

$$u_{ik} \leftarrow \langle \mathbf{v}_{i:}, \mathbf{w}_k \rangle$$

- GELU activation:  $\mathbf{v} = \sigma(\mathbf{u})$
- 1x1 conv again with  $d$  filters

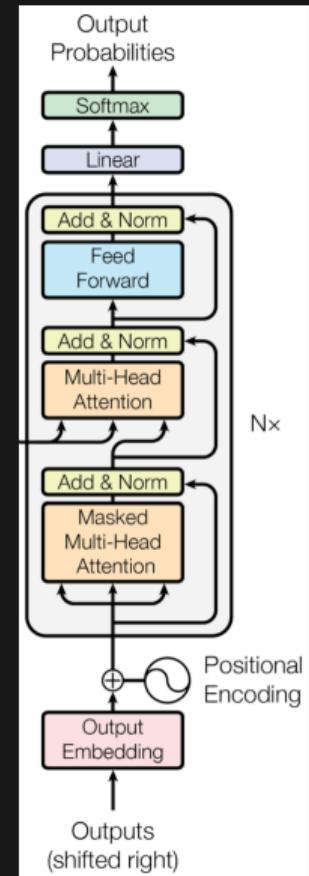


# Transformer Decoder

$$[Q, K, V] \leftarrow [VW^q, VW^k, VW^v]$$

$$V \leftarrow \text{softmax}(QK^\top/\lambda) \cdot V$$

- **Masked self-attention:** each output can only depend on the input and **previous** outputs
  - simply reset  $\langle q_{i,:}, k_{j,:} \rangle = -\infty$  if  $i \leq j$
  - or shift output to the right by 1 position and enforce above for  $i < j$
- **Context attention:**  $(K, V)$  from the encoder output while  $Q$  from decoder
  - each output attends to every position in input



# Transformer

- Last linear transformation and softmax:

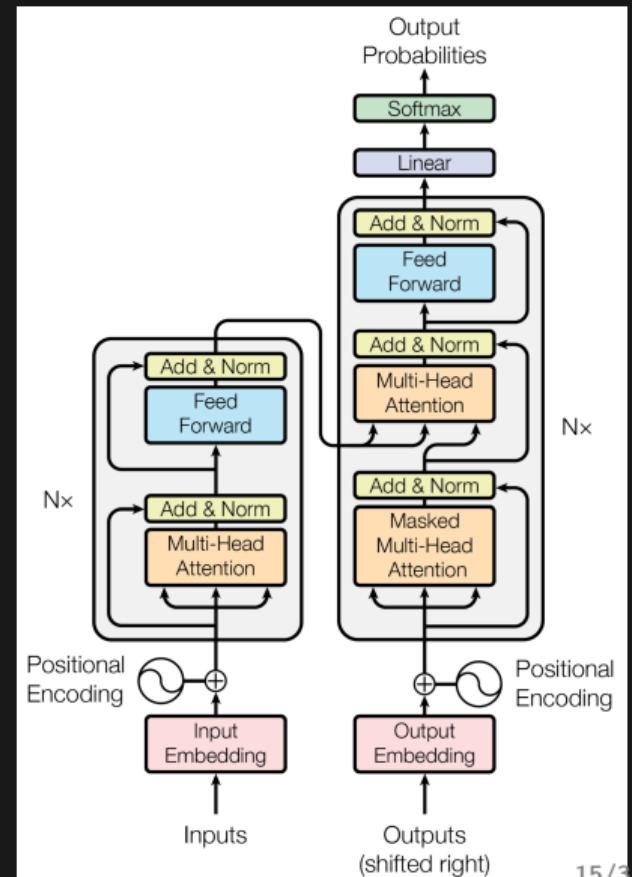
$$\hat{Y} = \text{softmax}(\mathbf{V}(\mathbf{W}^e)^\top)$$

–  $\mathbf{W}^e$  is the transpose of the input encoding

- Train by minimizing the log-loss:

$$\min_W \hat{\mathbb{E}} \left[ -\langle Y, \log \hat{Y} \rangle \right]$$

- $Y = [\mathbf{y}_1, \dots, \mathbf{y}_l]$  is output sequence, one-hot
- $\hat{Y} = [\hat{\mathbf{y}}_1, \dots, \hat{\mathbf{y}}_l]$  is the predicted probabilities
- $\hat{\mathbf{y}}_t$  depends on the input sequence  $X = [\mathbf{x}_1, \dots, \mathbf{x}_m]$ , as well as previous output tokens  $\mathbf{y}_1, \dots, \mathbf{y}_{t-1}$



# Comparison

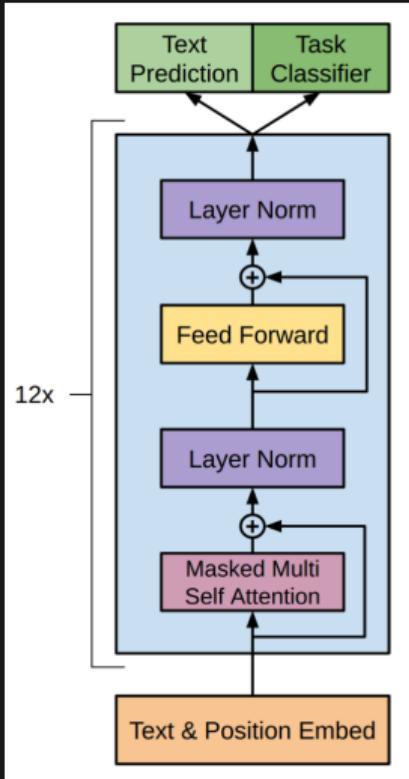
Layer type	per-layer complexity	sequential ops	max path length
Self-attention	$O(m^2d)$	$O(1)$	$O(1)$
Recurrent	$O(md^2)$	$O(m)$	$O(m)$
Convolution	$O(kmd^2)$	$O(1)$	$O(\log_k m)$
Self-attention (r)	$O(rmd)$	$O(1)$	$O(m/r)$

- $m$ : sequence length
- $d$ : dimension of internal representation
- $k$ : filter size
- $r$ : restricted attention field
- Attention also costs  $O(md^2)$  due to linear transformation  $V \leftarrow VW$

# Does It Work?

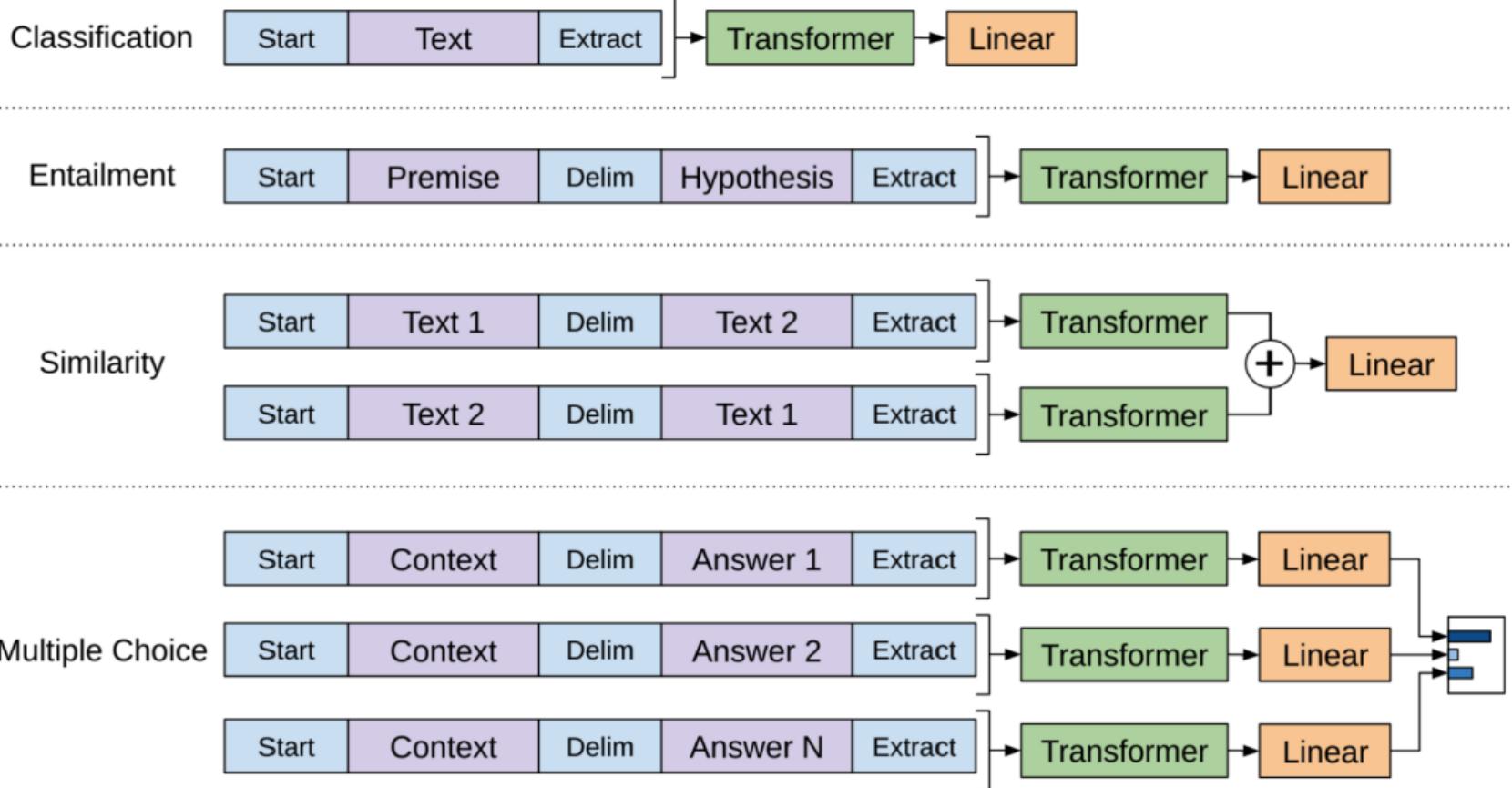
Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [15]	23.75			
Deep-Att + PosUnk [32]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [31]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [8]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [26]	26.03	40.56	$2.0 \cdot 10^{19}$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

# Generative Pre-Training (GPT)



- **Unsupervised** pre-training through a language model:  
$$\min_{\Theta} \hat{\mathbb{E}} - \log \prod_{j=1}^m p(\mathbf{x}_j | \mathbf{x}_1, \dots, \mathbf{x}_{j-1}; \Theta)$$
  - given the context consisting of previous tokens  $\mathbf{x}_1, \dots, \mathbf{x}_{j-1}$ , predict the current token  $\mathbf{x}_j$
- **Supervised** fine-tuning with task-aware transformations:  
$$\min_{W_y} \min_{\Theta} \hat{\mathbb{E}} - \log p(\mathbf{y} | X, \Theta) - \lambda \cdot \hat{\mathbb{E}} \log p(X | \Theta),$$

where  $p(\mathbf{y} | X, \Theta) = \langle \mathbf{y}, \text{softmax}(\mathbf{h}_m^{(L)} W_y) \rangle,$

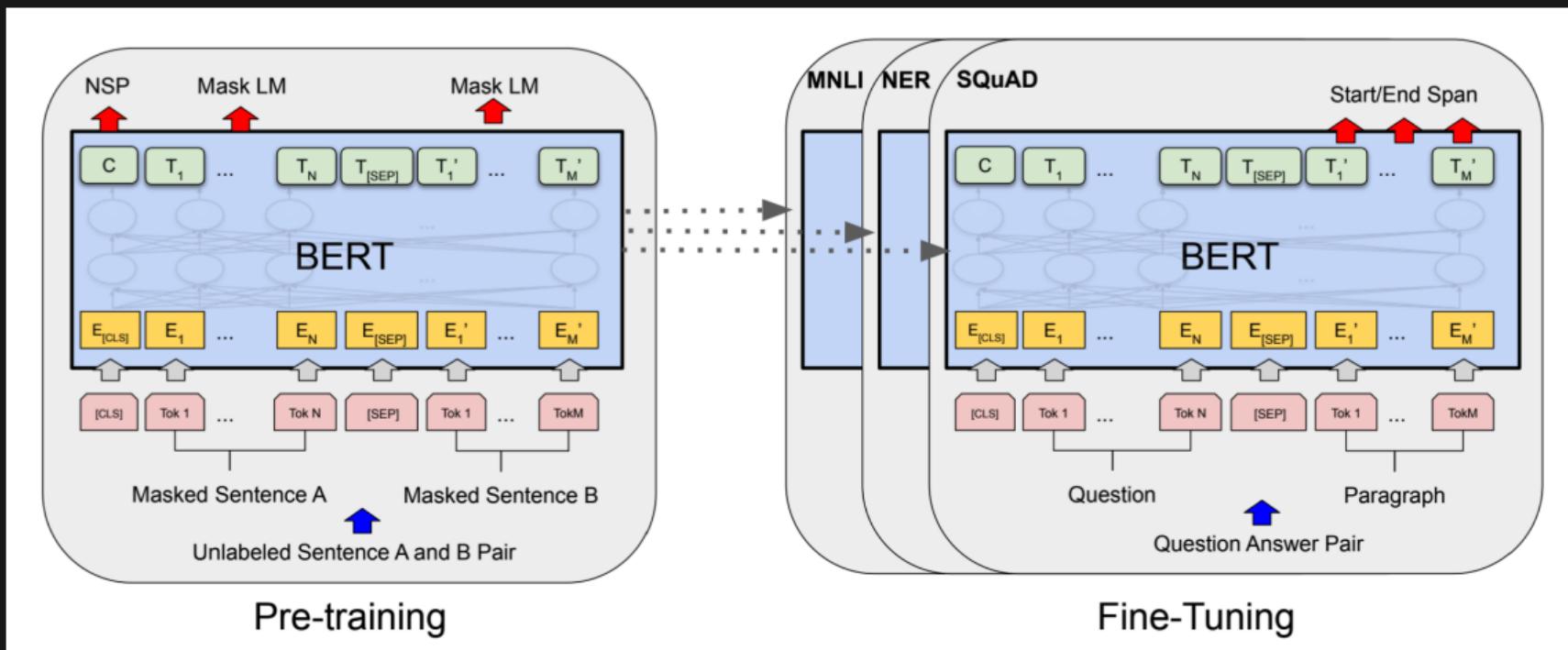


Method	Classification		Semantic Similarity		GLUE	
	CoLA (mc)	SST2 (acc)	MRPC (F1)	STSB (pc)	QQP (F1)	
Sparse byte mLSTM [16]	-	<b>93.2</b>	-	-	-	-
TF-KLD [23]	-	-	<b>86.0</b>	-	-	-
ECNU (mixed ensemble) [60]	-	-	-	<b>81.0</b>	-	-
Single-task BiLSTM + ELMo + Attn [64]	<u>35.0</u>	90.2	80.2	55.5	<u>66.1</u>	64.8
Multi-task BiLSTM + ELMo + Attn [64]	18.9	91.6	83.5	72.8	<u>63.3</u>	<u>68.9</u>
Finetuned Transformer LM (ours)	<b>45.4</b>	91.3	82.3	<b>82.0</b>	<b>70.3</b>	<b>72.8</b>

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

Method	Story Cloze	RACE-m	RACE-h	RACE
val-LS-skip [55]	76.5	-	-	-
Hidden Coherence Model [7]	<u>77.6</u>	-	-	-
Dynamic Fusion Net [67] (9x)	-	55.6	49.4	51.2
BiAttention MRU [59] (9x)	-	<u>60.2</u>	<u>50.3</u>	<u>53.3</u>
Finetuned Transformer LM (ours)	<b>86.5</b>	<b>62.9</b>	<b>57.4</b>	<b>59.0</b>

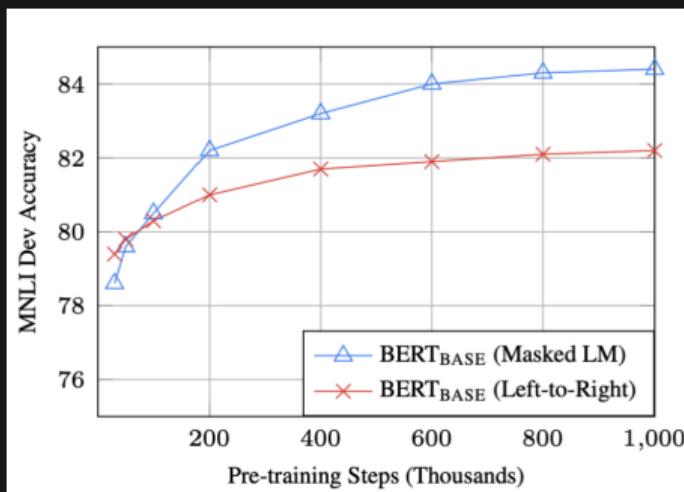
# Bidirectional Encoder Representations from Transformers



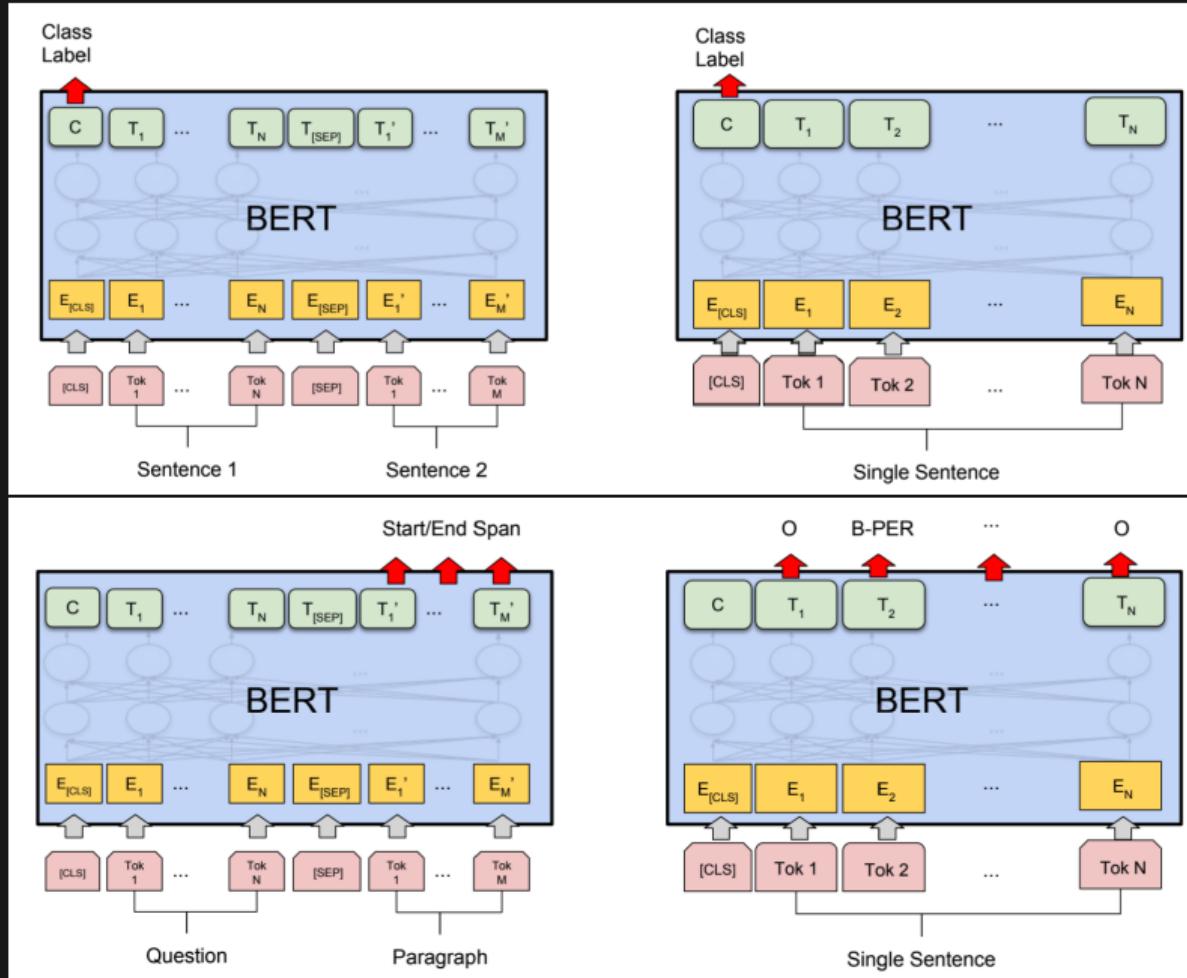
J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*. 2019, pp. 4171–4186.

# Mask Language Model

- Randomly select 15% input tokens, change to [Mask]
- Add softmax to predict the [Mask] tokens
- Actually 12% replaced with [Mask], 1.5% with random



MASK	Masking Rates			Dev Set Results		
	SAME	RND	MNLI	Fine-tune	NER	Feature-based
80%	10%	10%	84.2	95.4	94.9	
100%	0%	0%	84.3	94.9	94.0	
80%	0%	20%	84.1	95.2	94.6	
80%	20%	0%	84.4	95.2	94.7	
0%	20%	80%	83.7	94.8	94.6	
0%	0%	100%	83.6	94.9	94.6	



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
BERT <sub>BASE</sub>	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
BERT <sub>LARGE</sub>	<b>86.7/85.9</b>	<b>72.1</b>	<b>92.7</b>	<b>94.9</b>	<b>60.5</b>	<b>86.5</b>	<b>89.3</b>	<b>70.1</b>	<b>82.1</b>

Hyperparams				Dev Set Accuracy			
#L	#H	#A	LM (ppl)	MNLI-m	MRPC	SST-2	
3	768	12	5.84	77.9	79.8	88.4	
6	768	3	5.24	80.6	82.2	90.7	
6	768	12	4.68	81.9	84.8	91.3	
12	768	12	3.99	84.4	86.7	92.9	
12	1024	16	3.54	85.7	86.9	93.3	
24	1024	16	3.23	86.6	87.8	93.7	

### Zero-shot

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

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

### One-shot

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

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

### Few-shot

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

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

### Fine-tuning

The model is trained via repeated gradient updates using a large corpus of example tasks.

- 1 sea otter => loutre de mer ← example #1

gradient update

- 1 peppermint => menthe poivrée ← example #2

gradient update

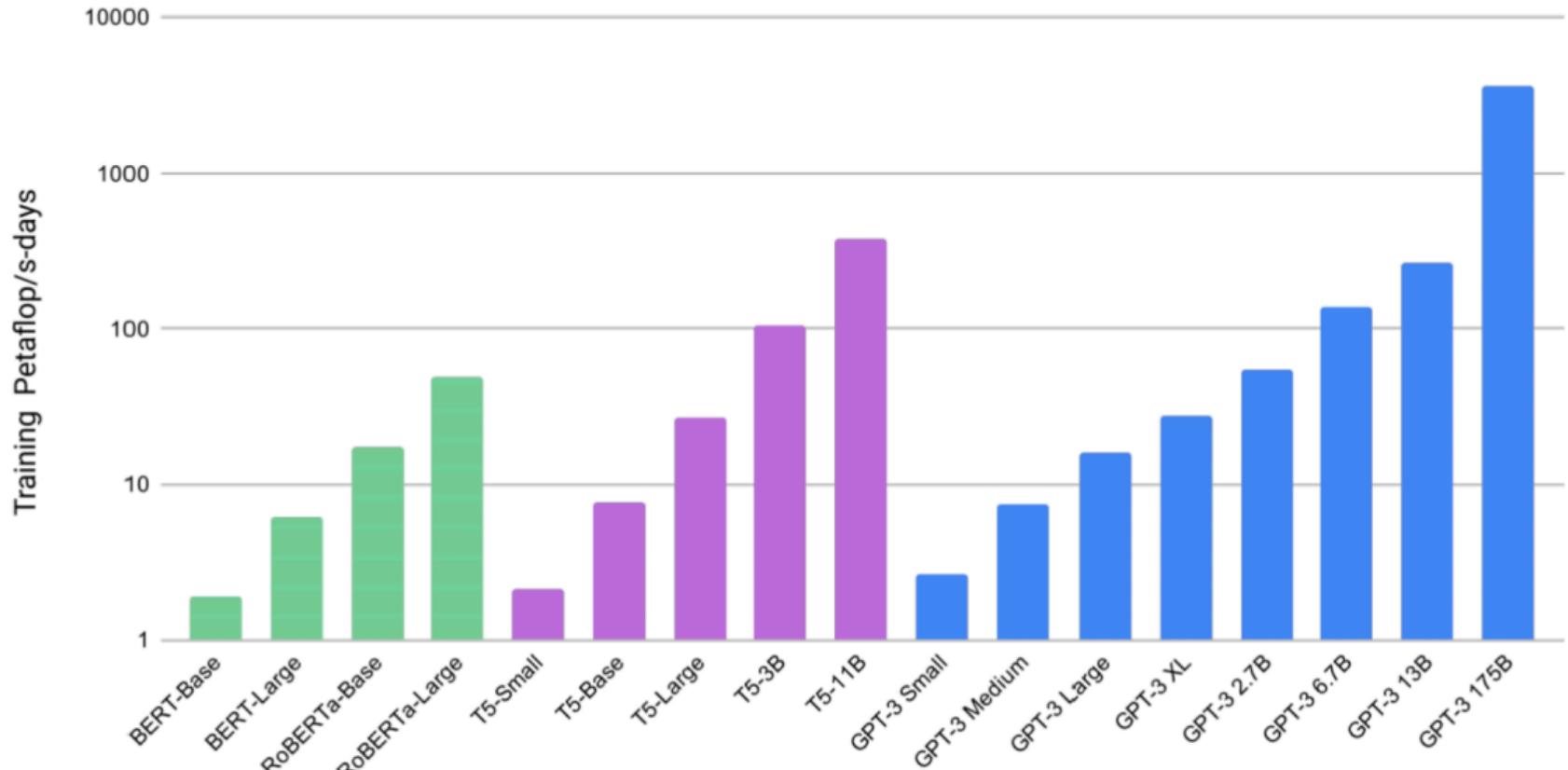
- 1 plush giraffe => girafe peluche ← example #N

gradient update

- 1 cheese => ..... ← prompt

<https://beta.openai.com/examples>

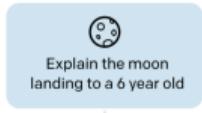
## Total Compute Used During Training



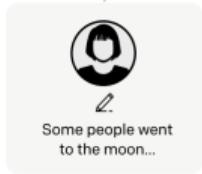
## Step 1

**Collect demonstration data, and train a supervised policy.**

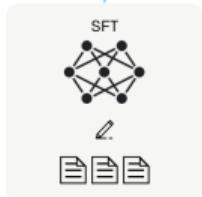
A prompt is sampled from our prompt dataset.



A labeler demonstrates the desired output behavior.



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



## Step 2

**Collect comparison data, and train a reward model.**

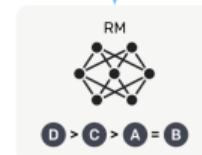
A prompt and several model outputs are sampled.



A labeler ranks the outputs from best to worst.



This data is used to train our reward model.



## Step 3

**Optimize a policy against the reward model using reinforcement learning.**

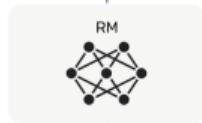
A new prompt is sampled from the dataset.



The policy generates an output.



Once upon a time...

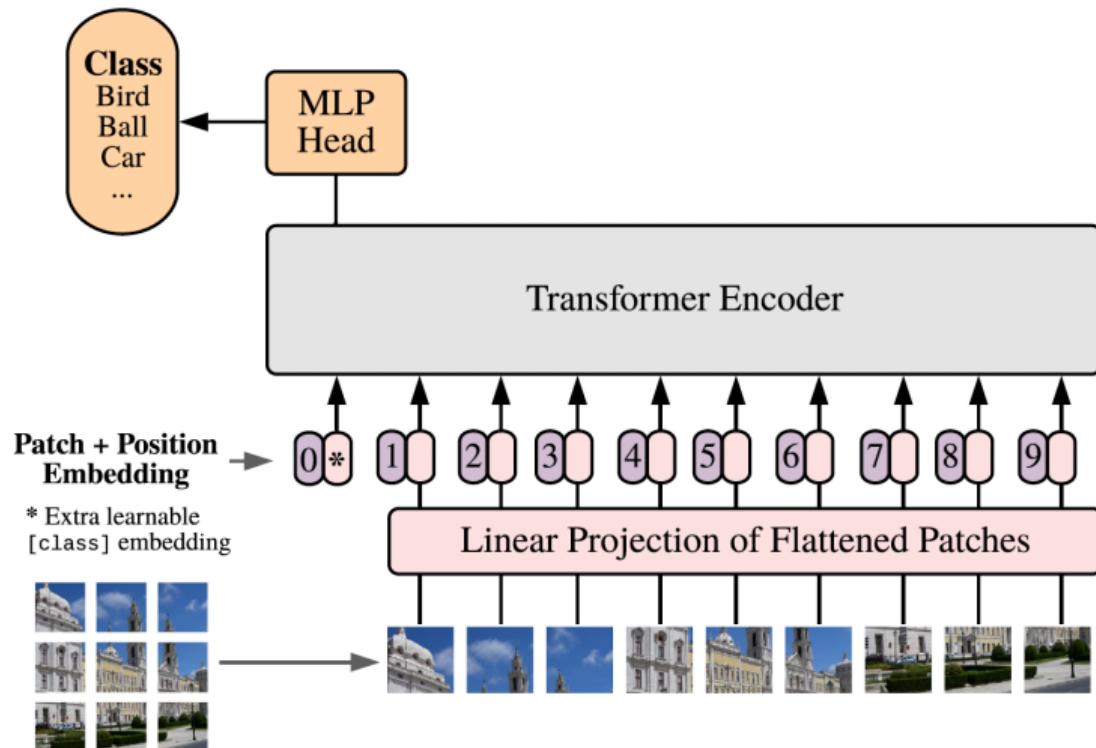


The reward model calculates a reward for the output.

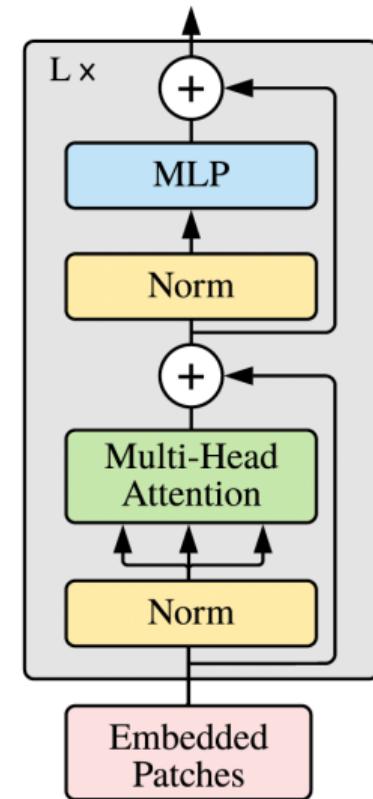
$r_k$

The reward is used to update the policy using PPO.

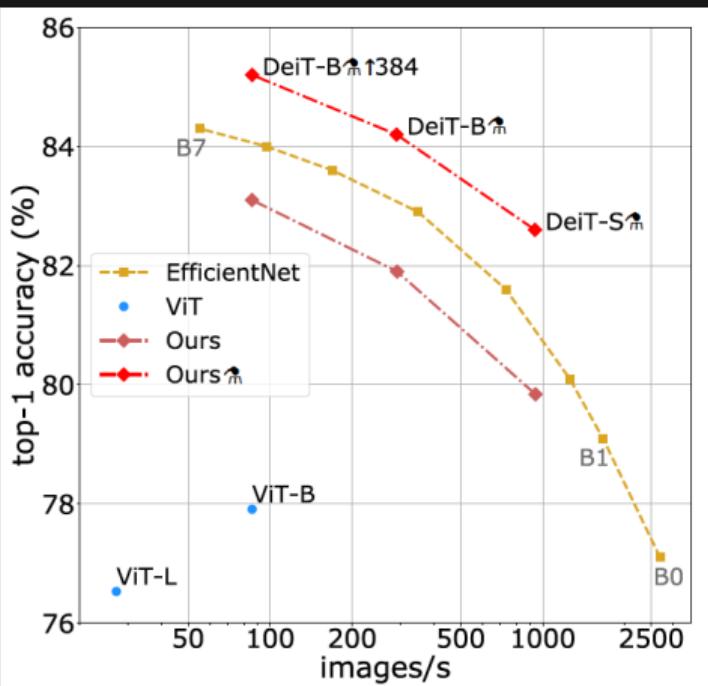
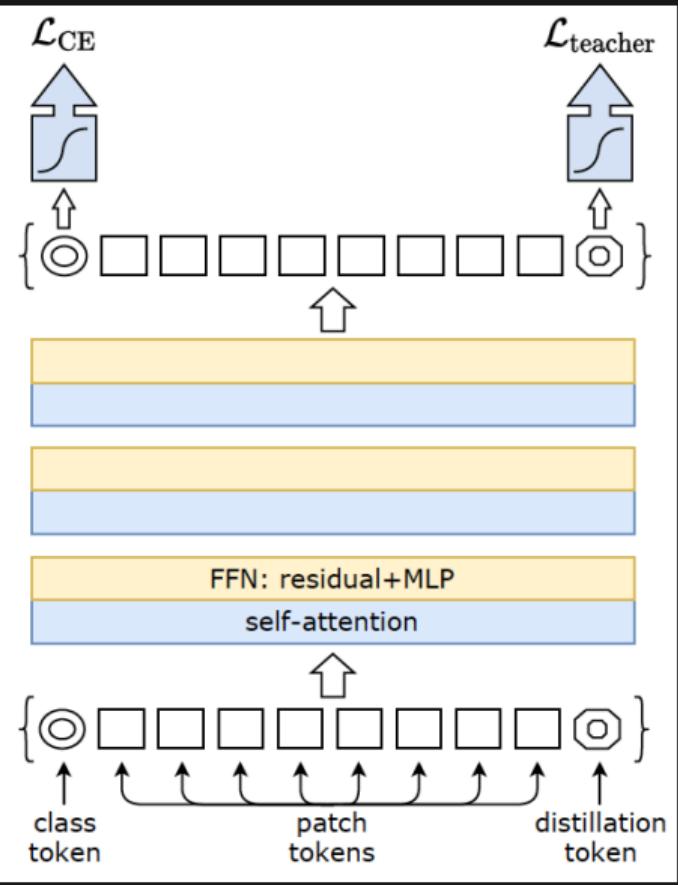
## Vision Transformer (ViT)



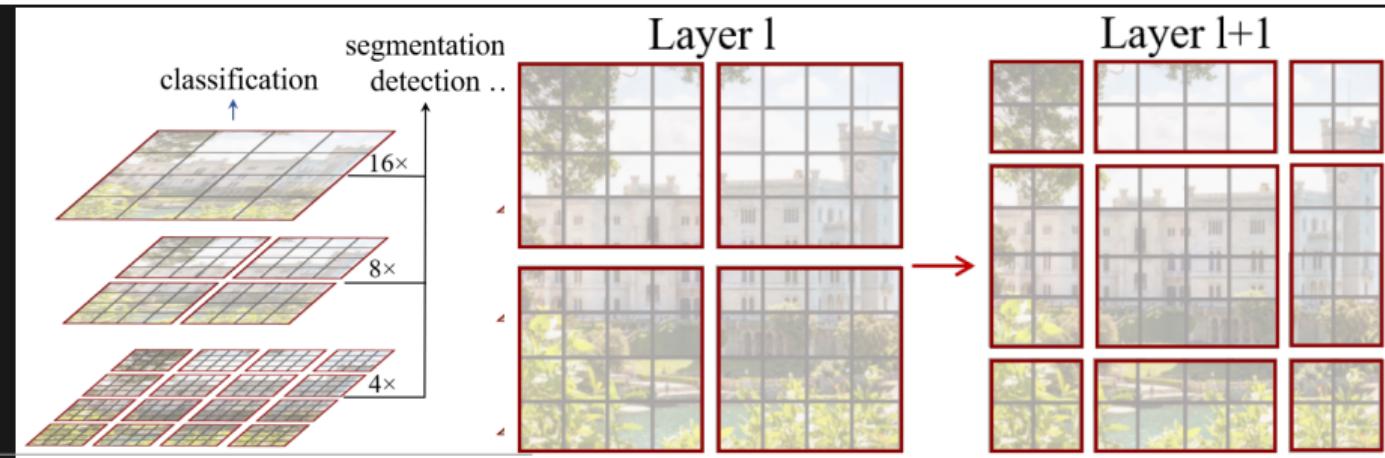
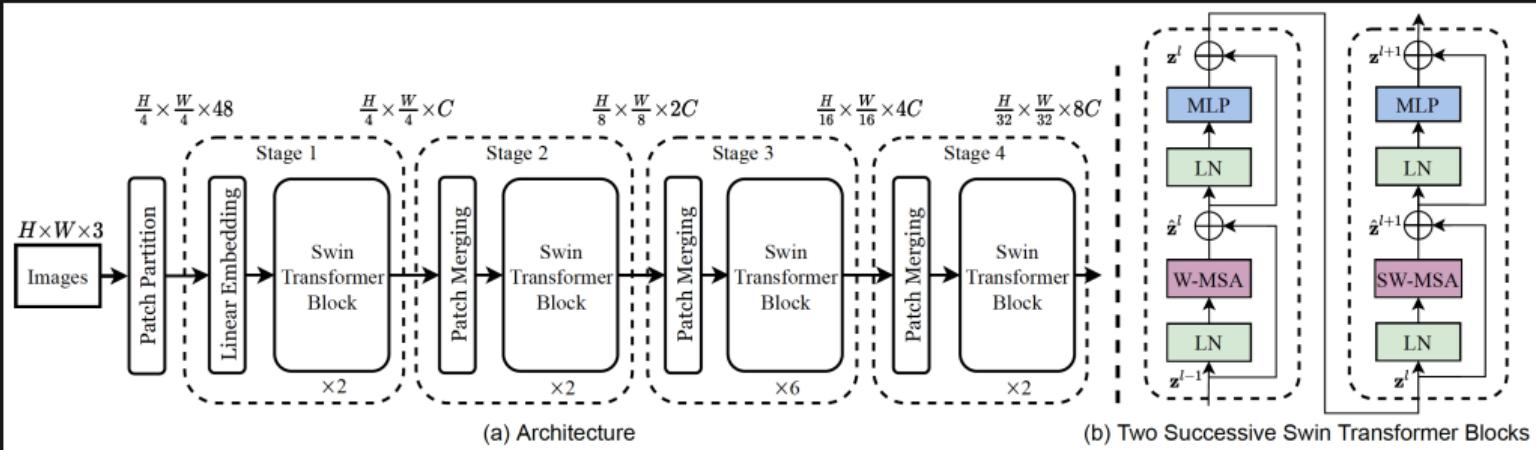
## Transformer Encoder



A. Dosovitskiy et al. "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". In: *International Conference on Learning Representations*. 2021.



H. Touvron et al. "Training data-efficient image transformers & distillation through attention". In: *Proceedings of the 38th International Conference on Machine Learning*. 2021, pp. 10347–10357.



Z. Liu et al. "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows". In: *IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021, pp. 9992–10002, Z. Liu et al. "Swin Transformer V2: Scaling Up Capacity and Resolution". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, pp. 11999–12009.

