

# **Pre-Training**

**NLP: Fall 2024**

**Anoop Sarkar**

# Preliminaries

# Word structure and subword models

- NLP used to model the vocabulary in simplistic ways based on English
- Tokenize based on spaces into a sequence of "words"
- All novel words at test time were mapped to [UNK] (unknown token)



# Byte Pair Encoding algorithm

- Learn a vocabulary of parts of words (subwords)
- Vocabulary of subwords is produced before training a model on the training dataset (larger the better)
- At training and test time the vocabulary is split up into a sequence of known subwords
- Byte Pair Encoding (BPE) algorithm (takes max merges as input)
  - Init subwords with individual characters/bytes and "end of word" token.
  - Using the training data find most common adjacent subwords, merge and add to list of subwords
  - Replace all pairs of characters with new subword token; iterate until max merges

# Word structure and subword models

- Common words are kept as part of the vocabulary (ignore morphology)
- Rarer words are split up into subword tokens
- In the worst case, words are split up into characters (or bytes)



# Pre-training Transformers

Representation Learning

# Brief History of Pre-training

## 1960 to 2015

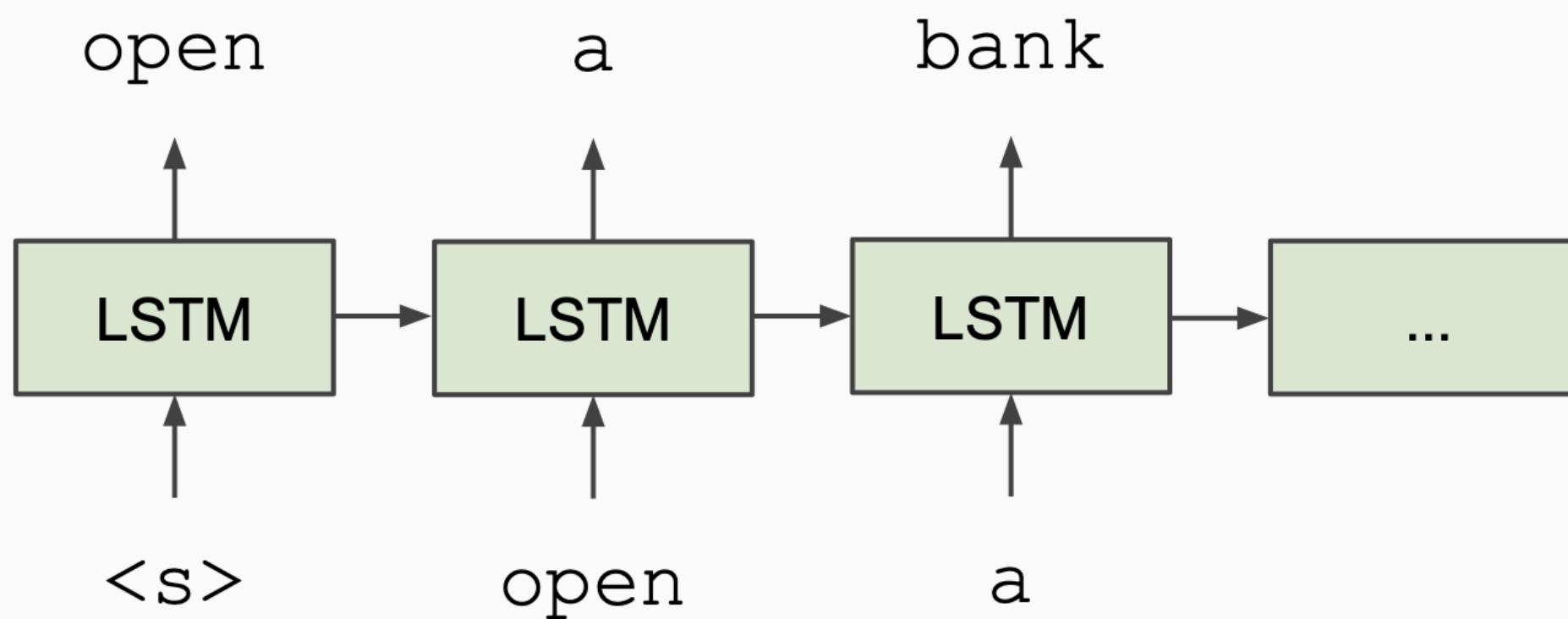
- Singular Value Decomposition (1960s):
  - Take matrix  $M \in |V| \times |V|$  of word co-occurrence counts
  - Use SVD to map  $M = USV^T$  truncate to  $|V| \times k$  initial singular values
  - Use truncated  $U$  use as word embeddings.
- Word2Vec/GloVe (2010):
  - Continuous Bag of Words (CBOW) - context words predict target word
  - Skip-gram - target word predicts each context word

# Semi-supervised Sequence Learning

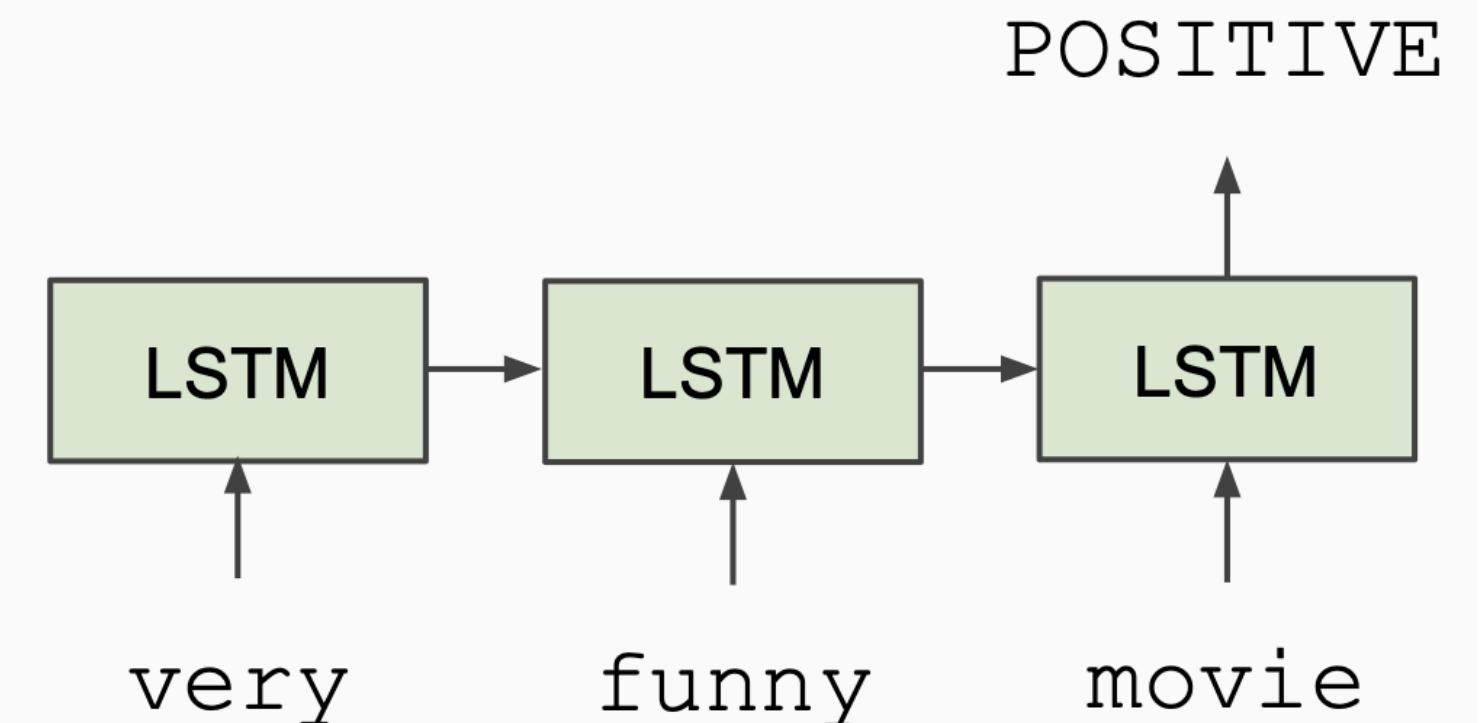
**Andrew M. Dai**  
Google Inc.  
`adai@google.com`

**Quoc V. Le**  
Google Inc.  
`qvl@google.com`

## Train LSTM Language Model



## Fine-tune on Classification Task



# Deep contextualized word representations

ELMO

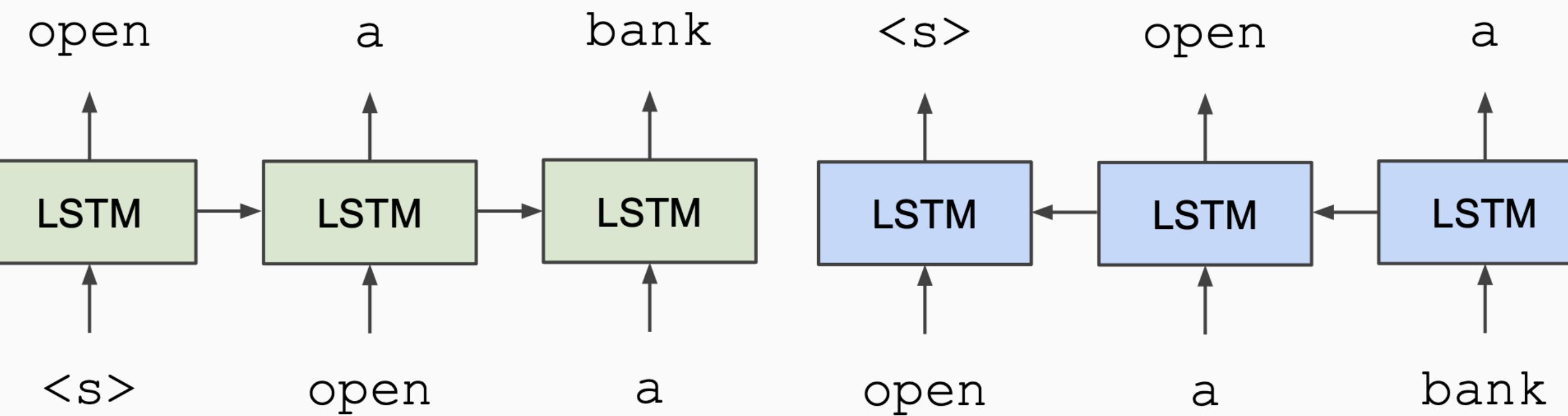
**Matthew E. Peters<sup>†</sup>, Mark Neumann<sup>†</sup>, Mohit Iyyer<sup>†</sup>, Matt Gardner<sup>†</sup>,**  
`{matthewp, markn, mohiti, mattg}@allenai.org`

**Christopher Clark<sup>\*</sup>, Kenton Lee<sup>\*</sup>, Luke Zettlemoyer<sup>†\*</sup>**  
`{csquared, kentonl, lsz}@cs.washington.edu`

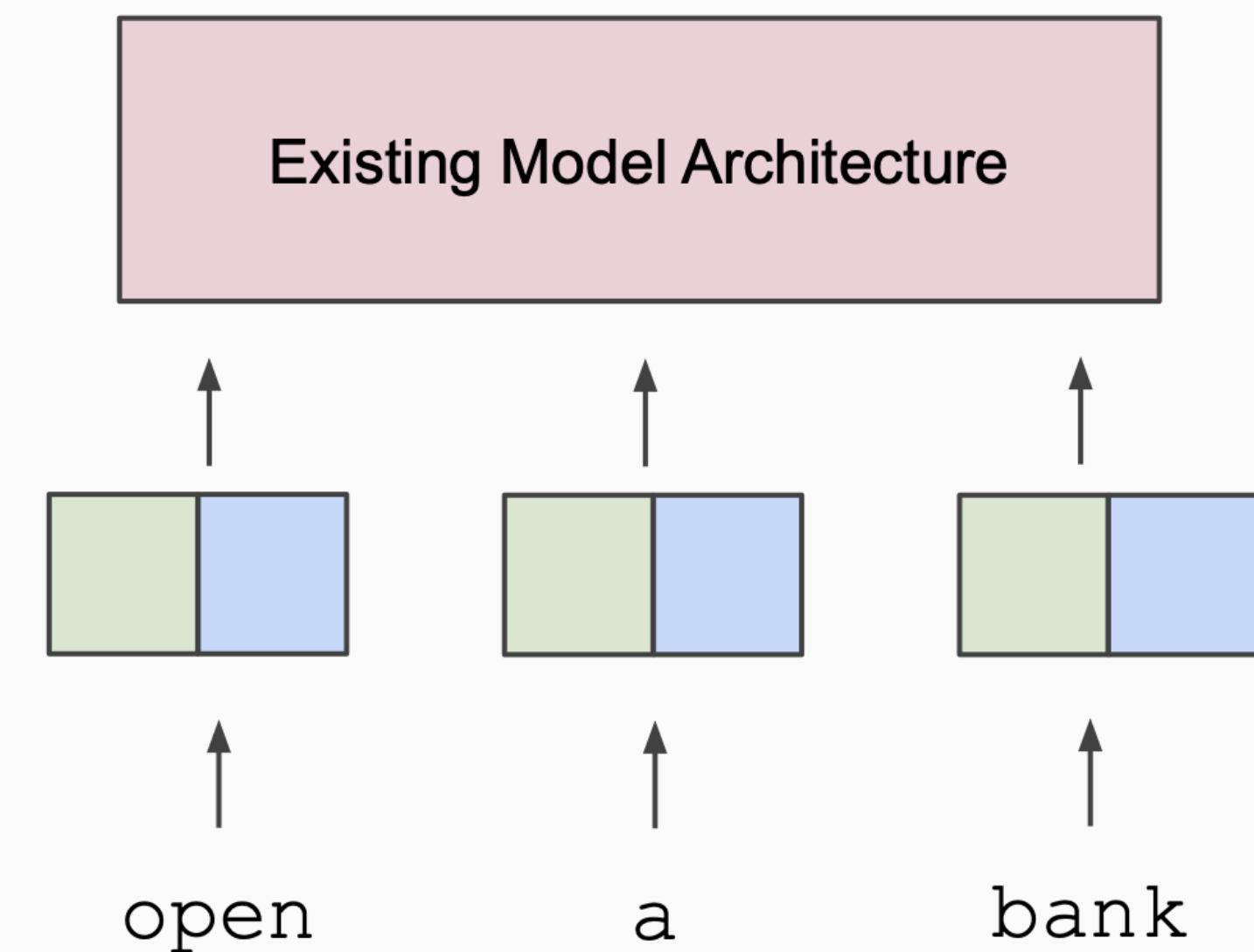
<sup>†</sup>Allen Institute for Artificial Intelligence

<sup>\*</sup>Paul G. Allen School of Computer Science & Engineering, University of Washington

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



## Apply as “Pre-trained Embeddings”



---

# Improving Language Understanding by Generative Pre-Training

---

GPT1

**Alec Radford**

OpenAI

[alec@openai.com](mailto:alec@openai.com)

**Karthik Narasimhan**

OpenAI

[karthikn@openai.com](mailto:karthikn@openai.com)

**Tim Salimans**

OpenAI

[tim@openai.com](mailto:tim@openai.com)

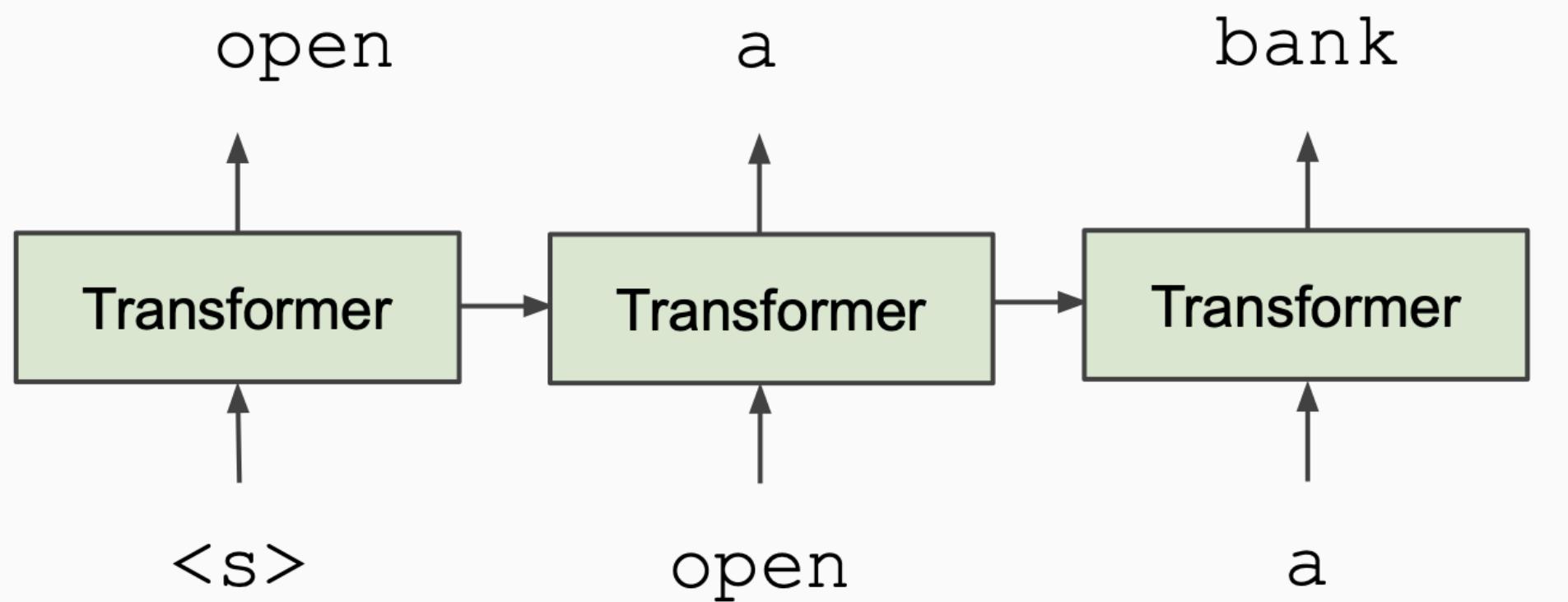
**Ilya Sutskever**

OpenAI

[ilyasu@openai.com](mailto:ilyasu@openai.com)

GPT1

## Train Deep (12-layer) Transformer LM



## Fine-tune on Classification Task

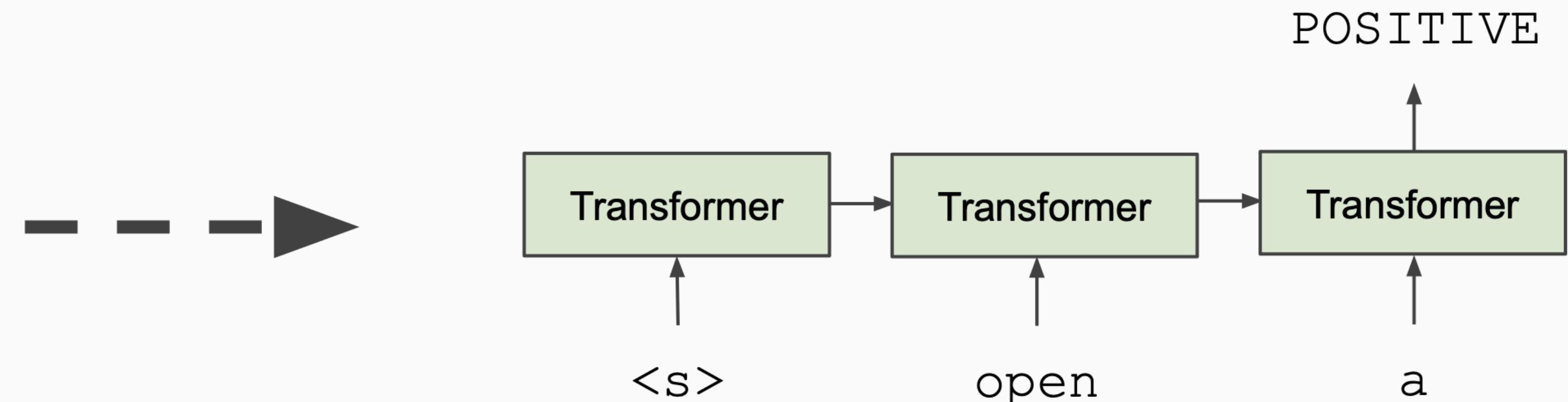


Fig from J. Devlin BERT slides

See also ULMFit: <https://arxiv.org/abs/1801.06146>

# GPT1

## Pre-training an autoregressive language model

- Start with a large amount of unlabeled data  $\mathcal{U} = \{u_1, \dots, u_n\}$
- Pre-training objective: Maximize the likelihood of predicting the next token

$$\bullet \quad L_i(\mathcal{U}) = \sum_i \log P(u_i | u_{i-k}, \dots, u_{i-1}; \Theta)$$

BooksCorpus: 7K  
unpublished books  
(1B words)

- This is equivalent to training a Transformer decoder

$$\bullet \quad h_0 = U \boxed{W_e} + W_p$$

$U = (u_{-k}, \dots, u_{-1})$  is the context vector of tokens

$n$  is the number of Transformer layers

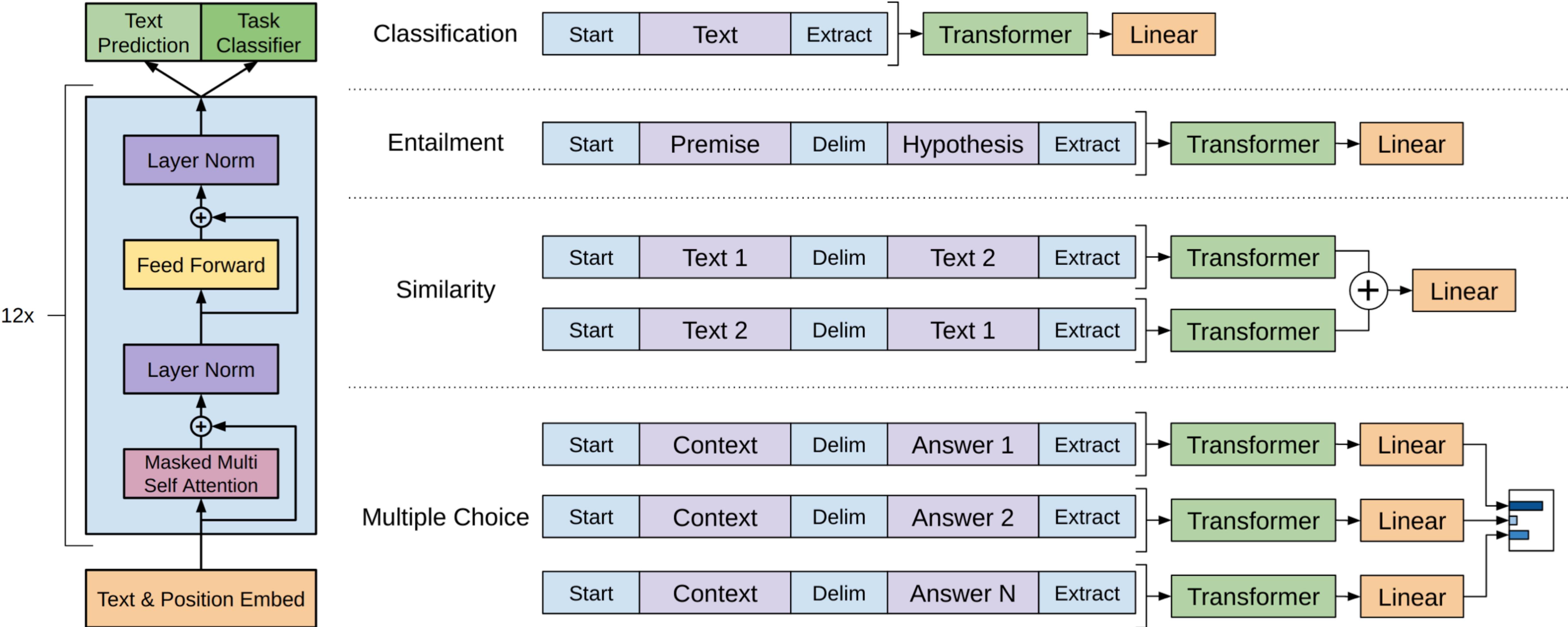
$W_e$  is the token embedding matrix

$$\bullet \quad h_\ell = \text{transformer\_block}(h_{\ell-1}) \forall \ell \in [1, n]$$

$W_p$  is the position embedding matrix

$$\bullet \quad P(u) = \text{softmax}(h_n \boxed{W_e^T})$$

- Directionality is needed to generate a well-formed probability distribution



<b>Dataset</b>	<b>Task</b>	<b>SOTA</b>	<b>GPT1</b>
SNLI	Textual entailment	89.3	89.9
MNLI matched	Textual entailment	80.6	82.1
MNLI mismatched	Textual entailment	80.1	81.4
SciTail	Textual entailment	83.3	88.3
QNLI	Textual entailment	82.3	88.1
RTE	Textual entailment	61.7	56.0
STS-B	Semantic similarity	81.0	82.0
QQP	Semantic similarity	66.1	70.3
MRPC	Semantic similarity	86.0	82.3
RACE	Reading comprehension	53.3	59.0
ROCStories	Commonsense reasoning	77.6	86.5
COPA	Commonsense reasoning	71.2	78.6
SST-2	Sentiment analysis	93.2	91.3
CoLA	Linguistic acceptability	35.0	45.4
GLUE	Multi task benchmark	68.9	72.8

# **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

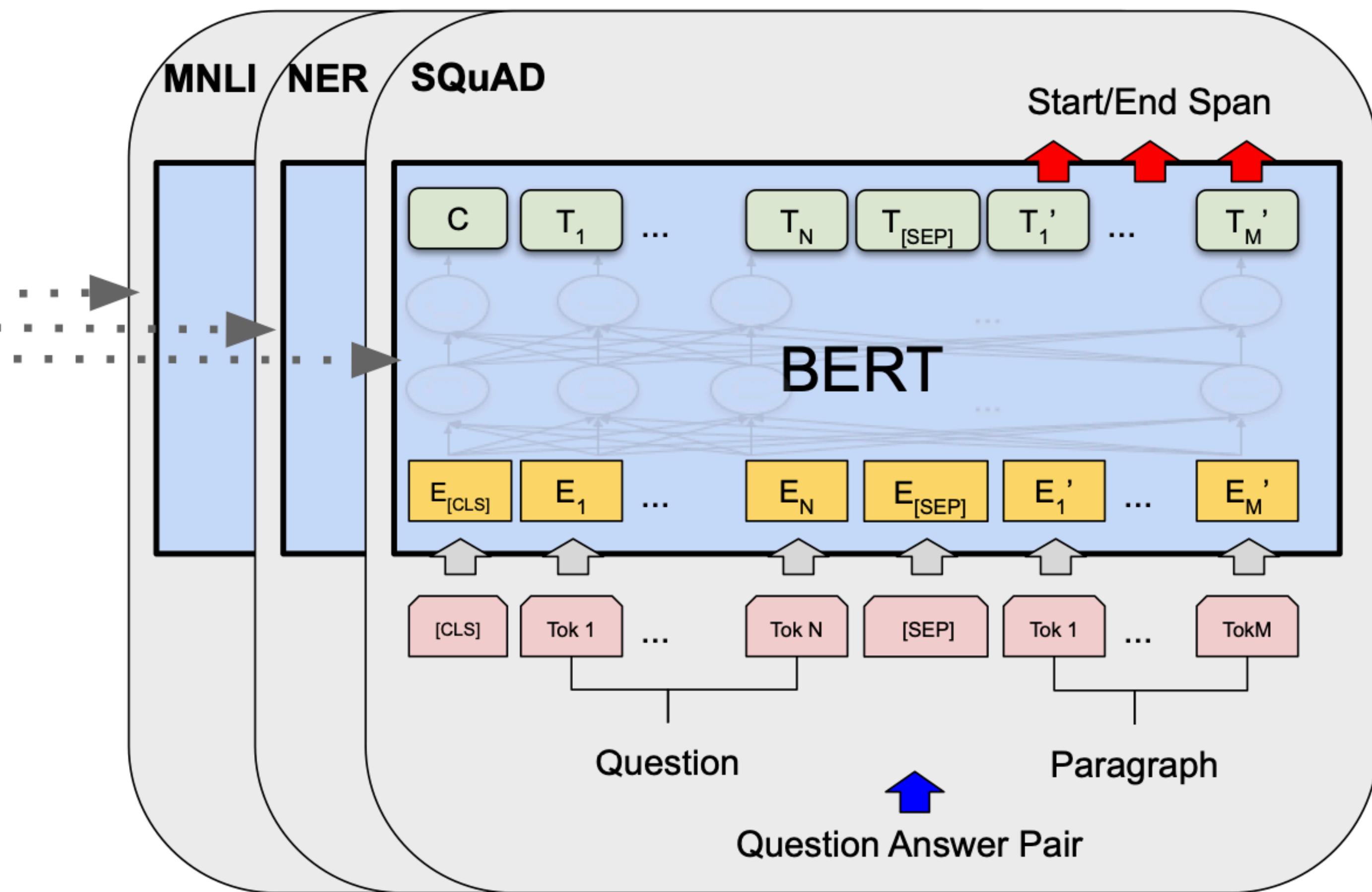
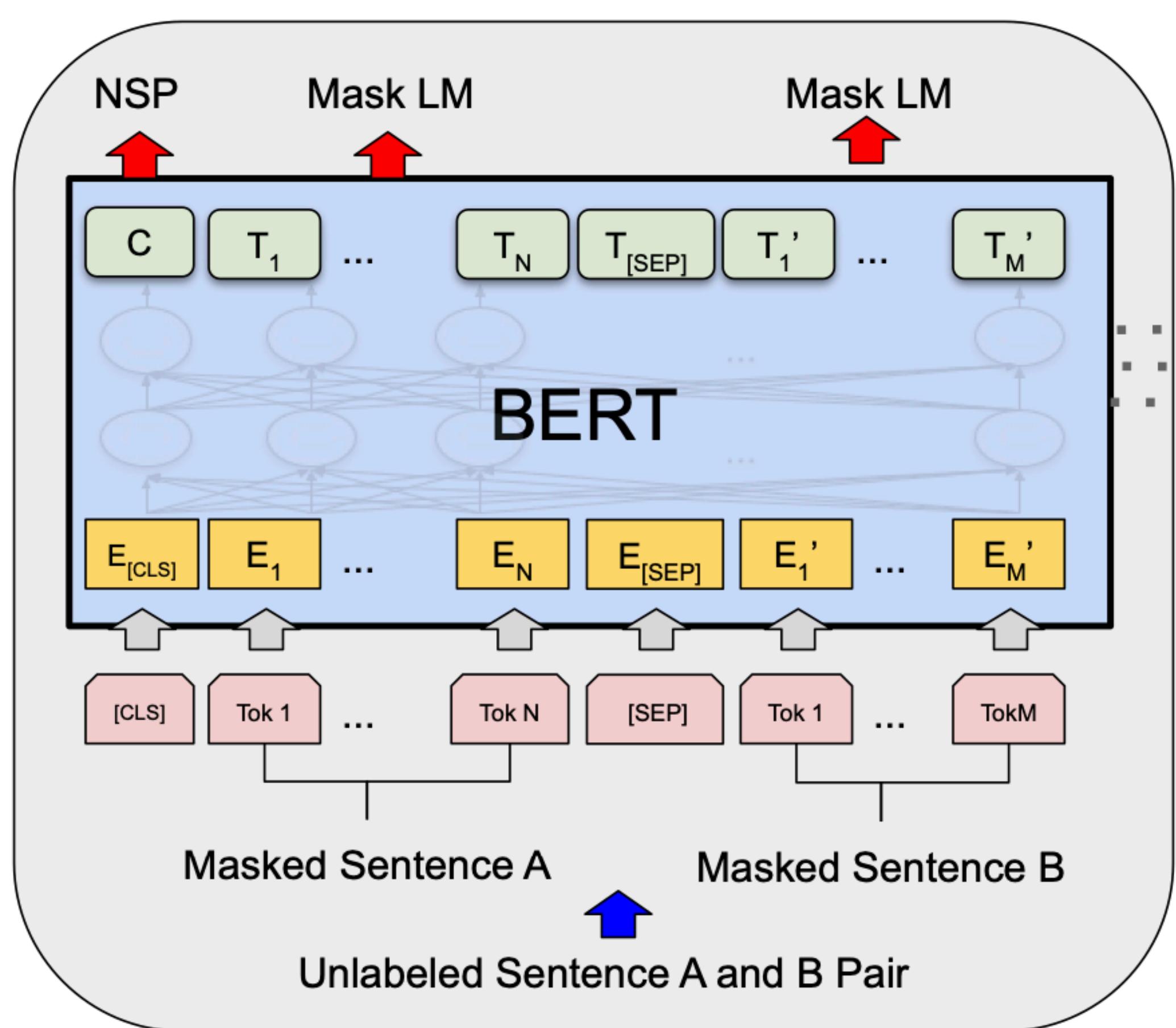


Fig from J. Devlin BERT slides

# Transformer encoder

## BERT model architecture

- Multi-headed self attention (models context)
- Feed-forward layers (non-linear hierarchical feature representation learning)
- LayerNorm and residuals (allows training of deep networks)
- Positional embeddings (allows model to learn relative position representation)

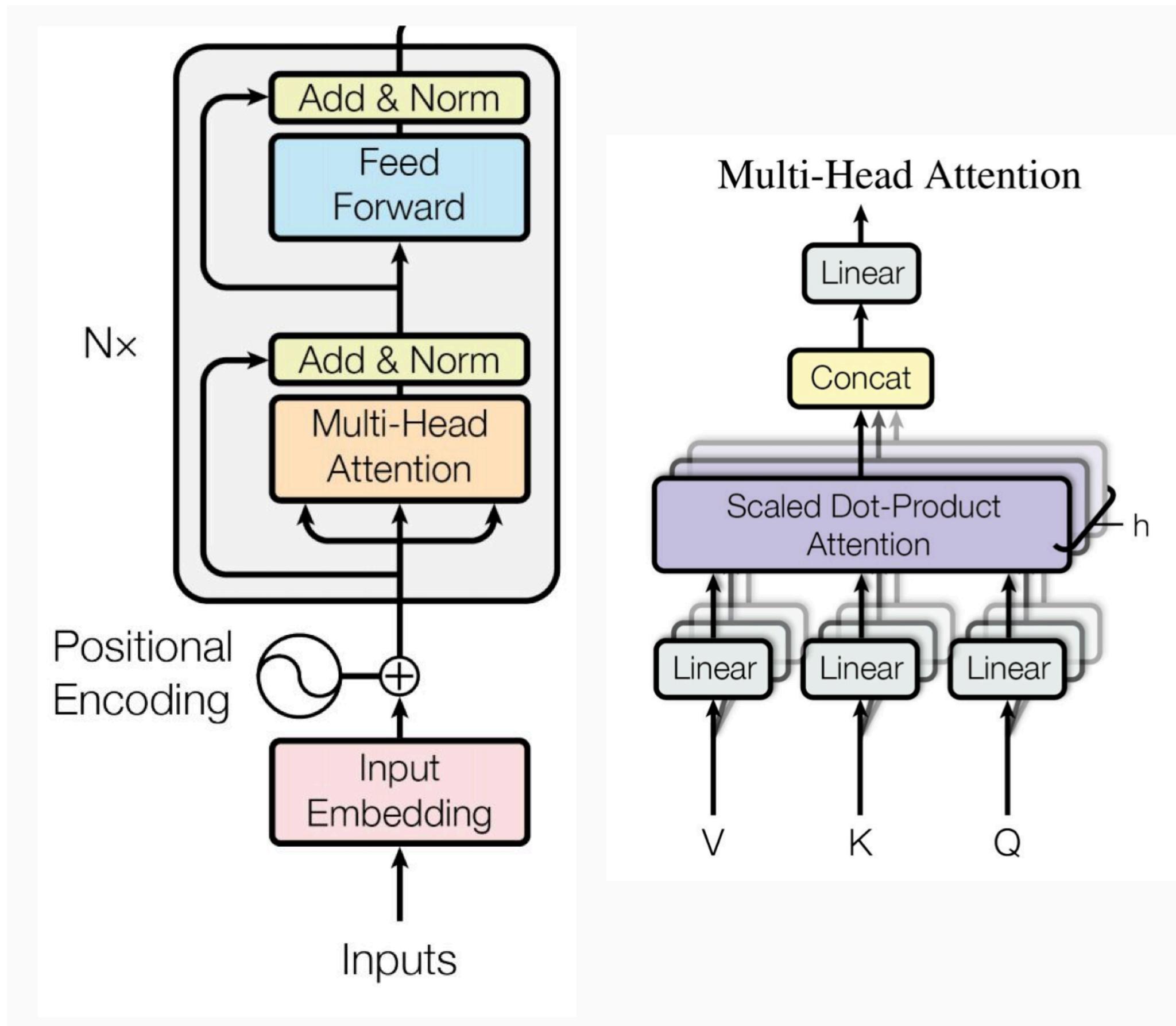


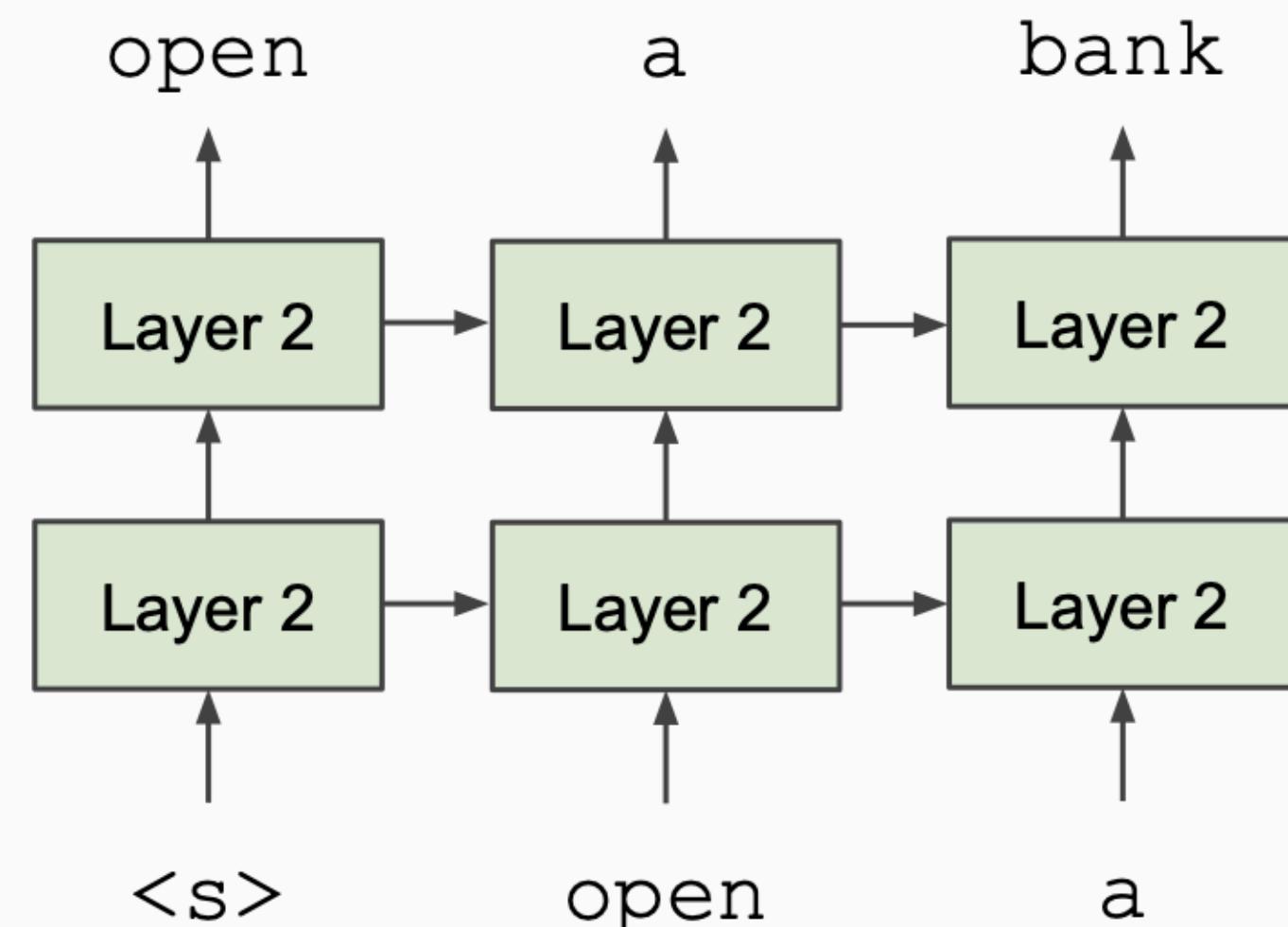
Fig from J. Devlin BERT slides

# Directionality

## Unidirectional context (GPT) vs Bidirectional context (ELMO)

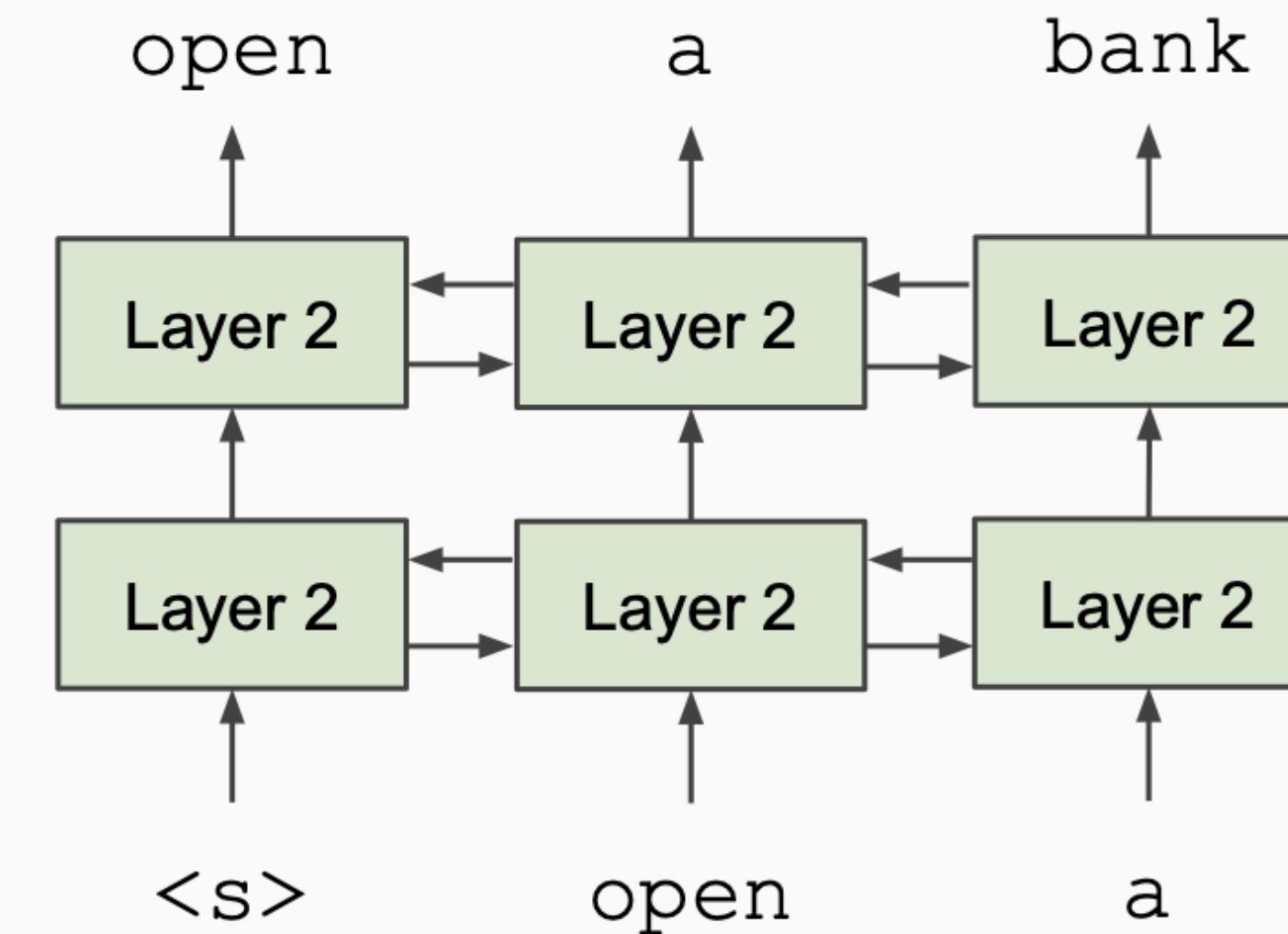
### Unidirectional context

Build representation incrementally



### Bidirectional context

Words can “see themselves”



# Bidirectional representation learning without probabilities

- Use the entire sentence context
- Don't worry about probabilities just solve a task and learn parameters
- Solution: use two loss functions
  1. Language model but masking a single arbitrary token at a time.
    - Called the cloze task (Taylor 1953) aka Masked language modeling
  2. Next sentence prediction (based on the Skip-Thought Vectors paper)

<https://psycnet.apa.org/record/1955-00850-001>

<https://arxiv.org/abs/1506.06726>

# Masked LM

- Loss function to train a Transformer
- Keep most of the sentences intact. Mask out k% of the input tokens (k=15)
- Predict the masked tokens
- Too little masking: too many epochs needed to train a good representation
- Too much masking: not enough context to predict the token



# Masked LM

**Problem: Mask token is never used for any fine-tuning task**

- Predict 15% of the tokens but do not replace tokens with [MASK] 100% of the time (for those 15% of tokens)

- Instead:

1. 80% of the time replace with [MASK] and predict the right token

store  
↑  
went to the [MASK]

2. 10% of the time replace with a random word and predict the right token

store  
↑  
went to the running

3. 10% of the time keep the token unchanged and predict

store  
↑  
went to the store

# Next Sentence Prediction (NSP)

## Learning sentence representations

- BERT is always provided with two sentences at a time during training separated by a [SEP] token: [CLS] Sentence A [SEP] Sentence B
- Replace 50% of Sentence B with a random sentence
- Otherwise use the Sentence B that follows Sentence A
- Loss function: Predict if Sentence B follows Sentence A or not

**Sentence A** = The man went to the store.

**Sentence B** = He bought a gallon of milk.

**Label** = IsNextSentence

**Sentence A** = The man went to the store.

**Sentence B** = Penguins are flightless.

**Label** = NotNextSentence

## 30K subword vocabulary

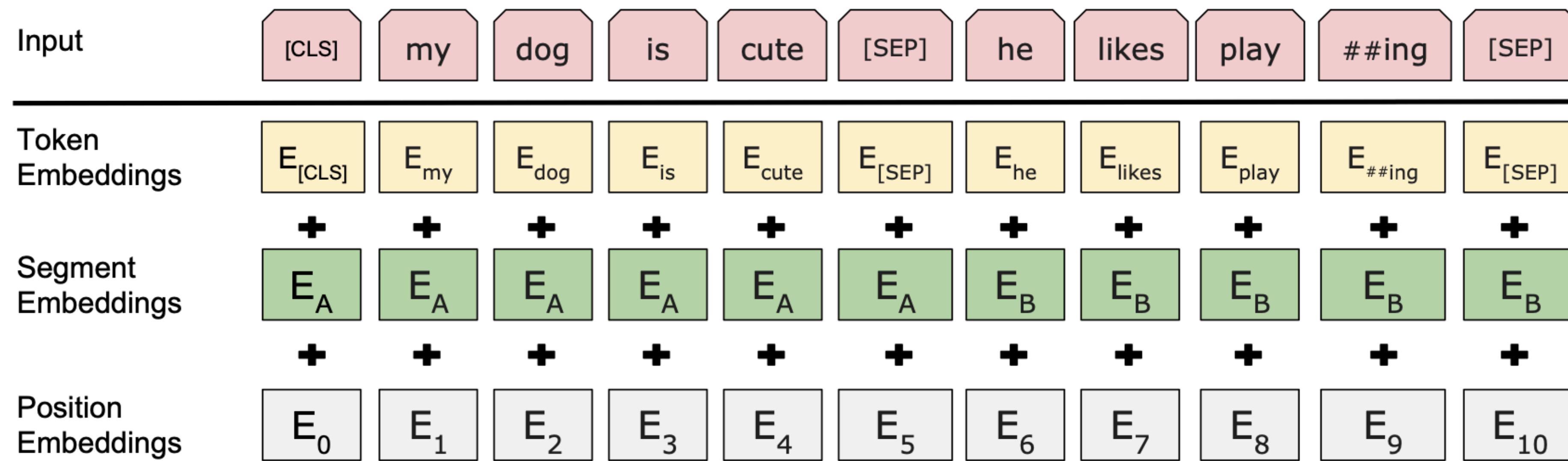
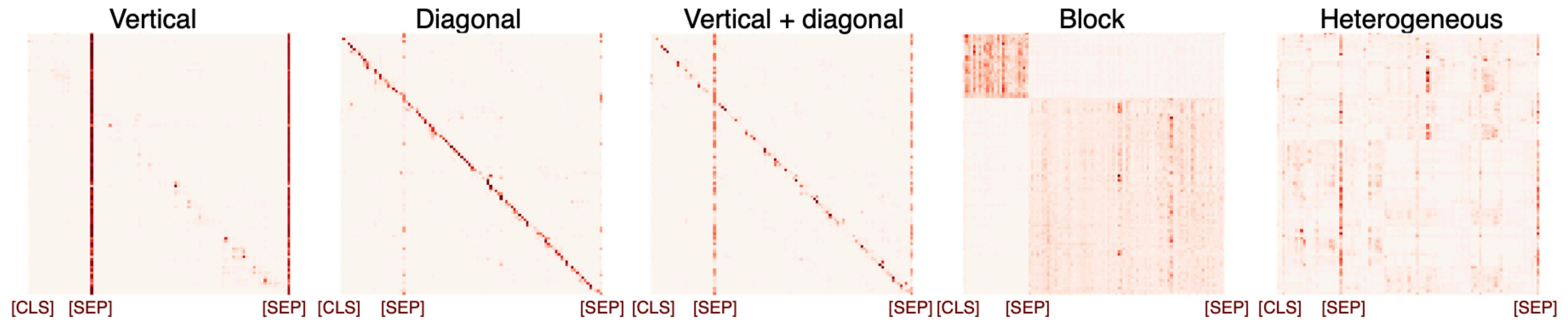


Figure 2: BERT input representation. The input embeddings are the sum of the token embeddings, the segmentation embeddings and the position embeddings.



(a) Reference: typical BERT self-attention patterns by Kovaleva et al. (2019).

# Model and Training

- **Data:** Wikipedia (2.5B tokens) + BooksCorpus (800M tokens)
- **Batch size:** 131,072 tokens
  - 1024 sequences × 128 length
  - 256 sequences × 512 length
- **Training time:** 1M steps (~40 epochs)
- **Optimizer:** AdamW, 1e-4 learning rate, linear decay
- **BERT-base:** 12 layer, 768 hidden, 12 attention heads. 110M parameters (=GPT1)
- **BERT-large:** 24 layer, 1024 hidden, 16 attention heads. 340M parameters

# Fine-tuning procedure

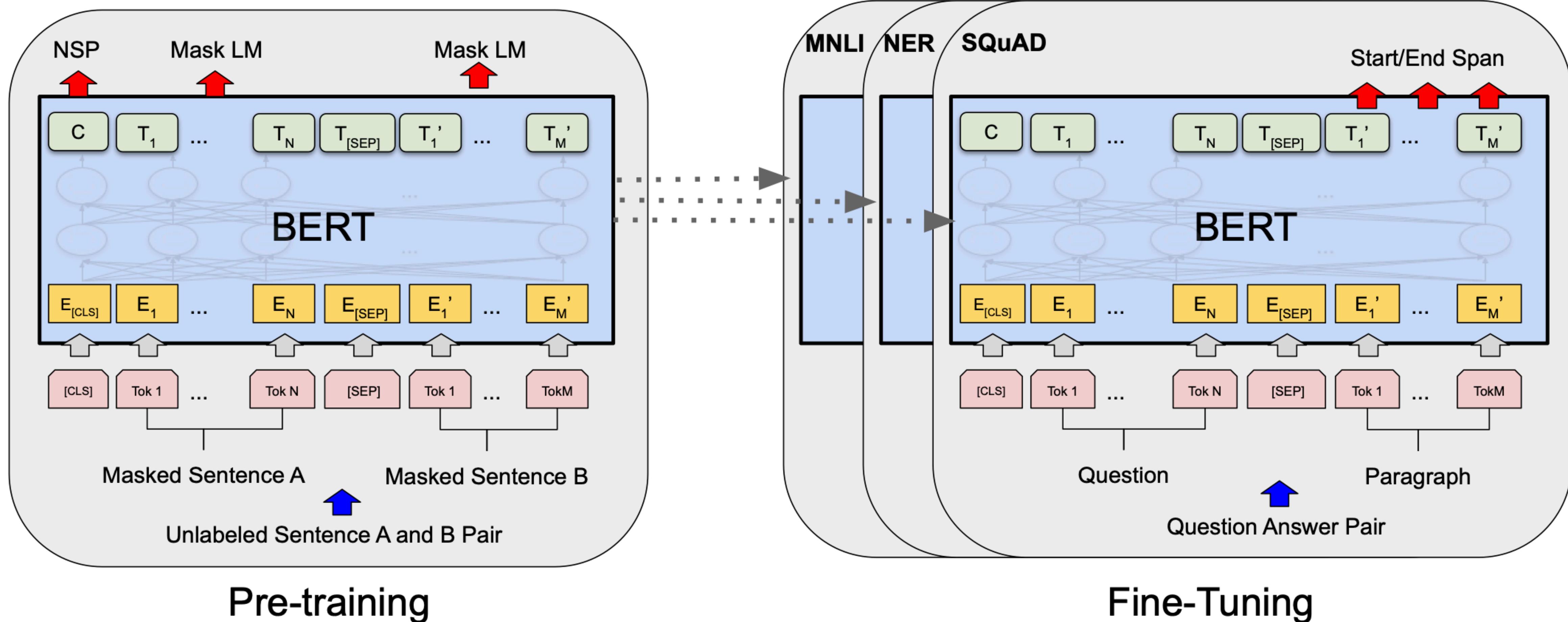
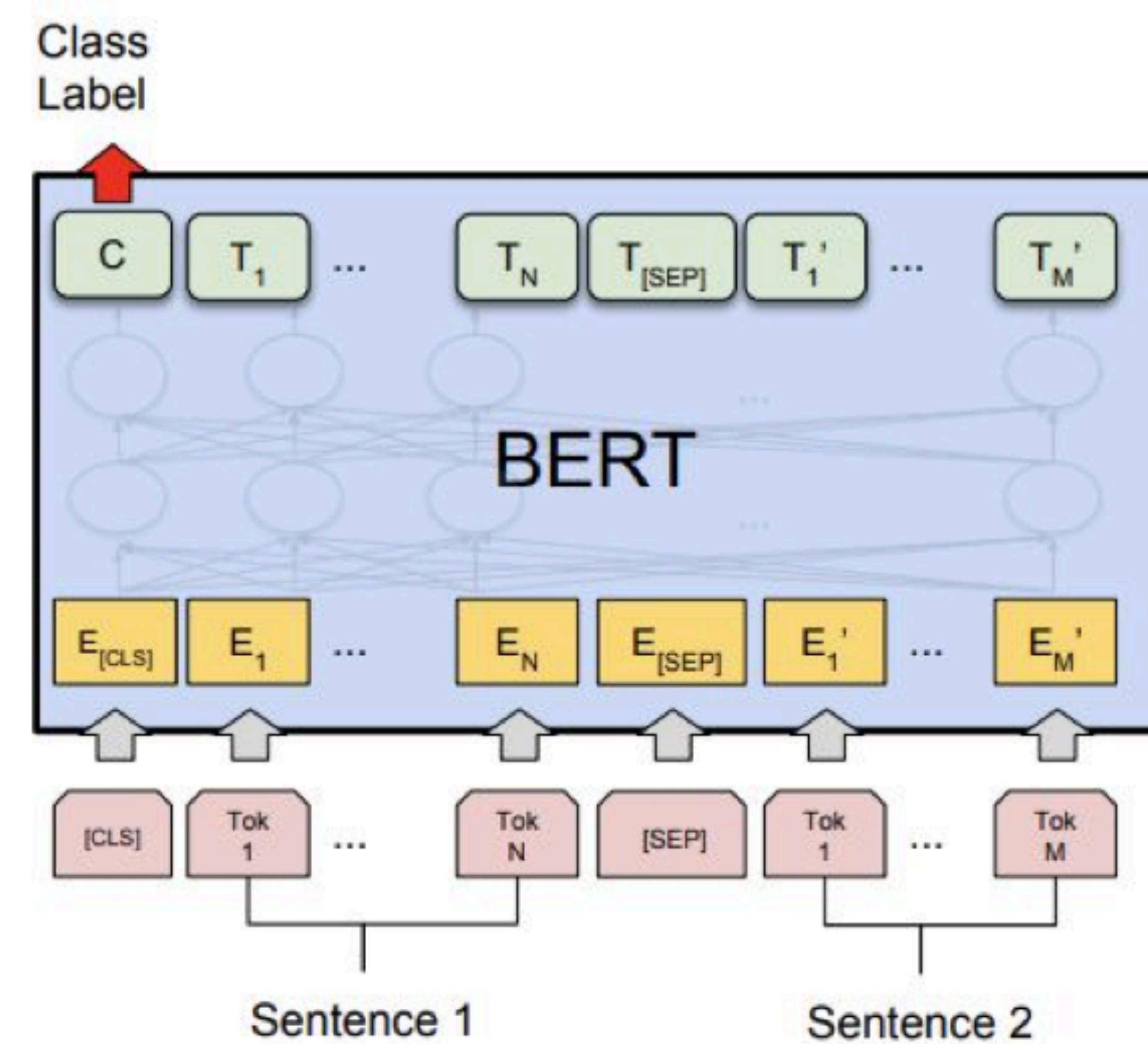


Fig from J. Devlin BERT slides

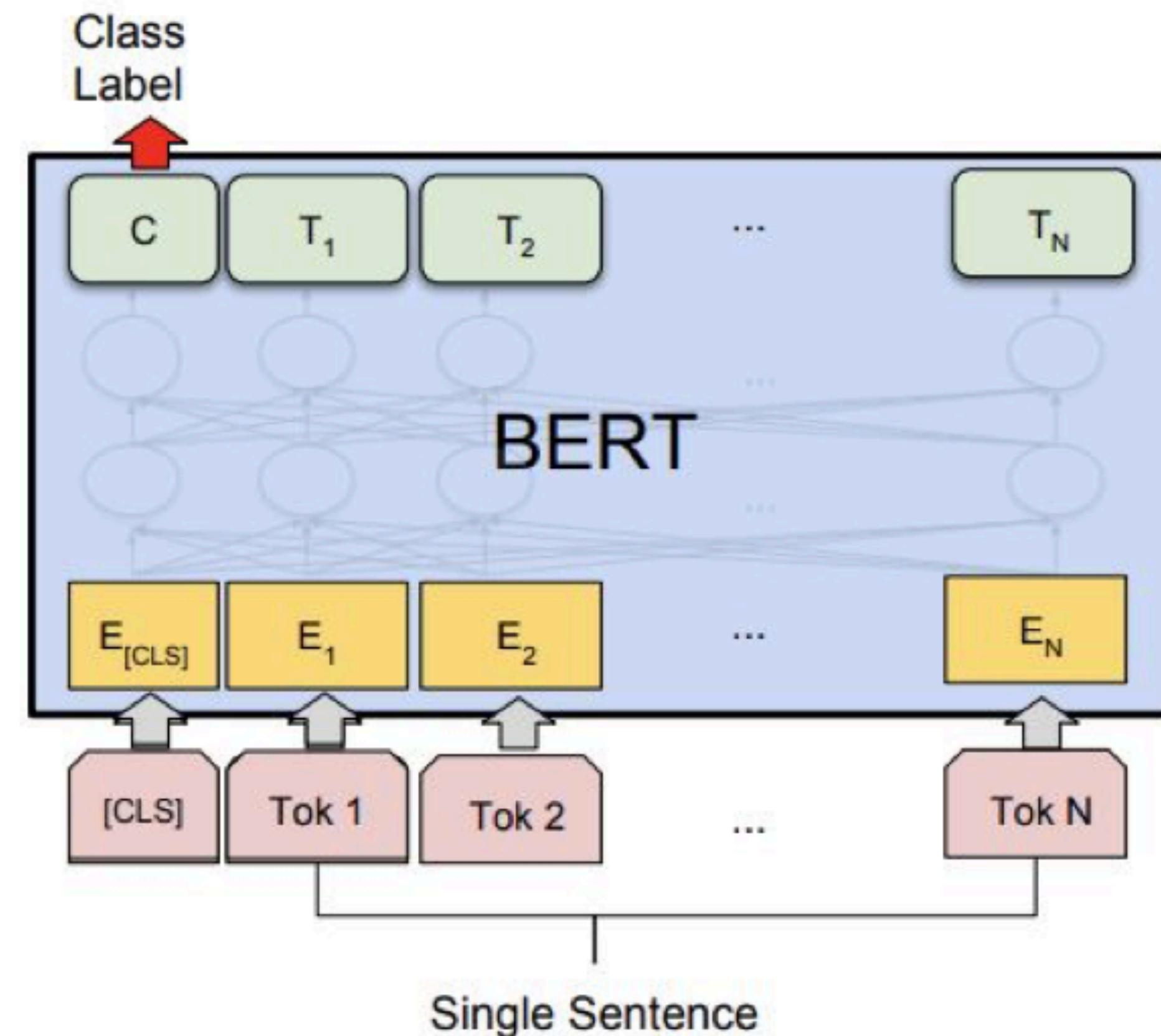
# Fine-tuning for sentence pair classification



(a) Sentence Pair Classification Tasks:  
MNLI, QQP, QNLI, STS-B, MRPC,  
RTE, SWAG

Fig from J. Devlin BERT slides

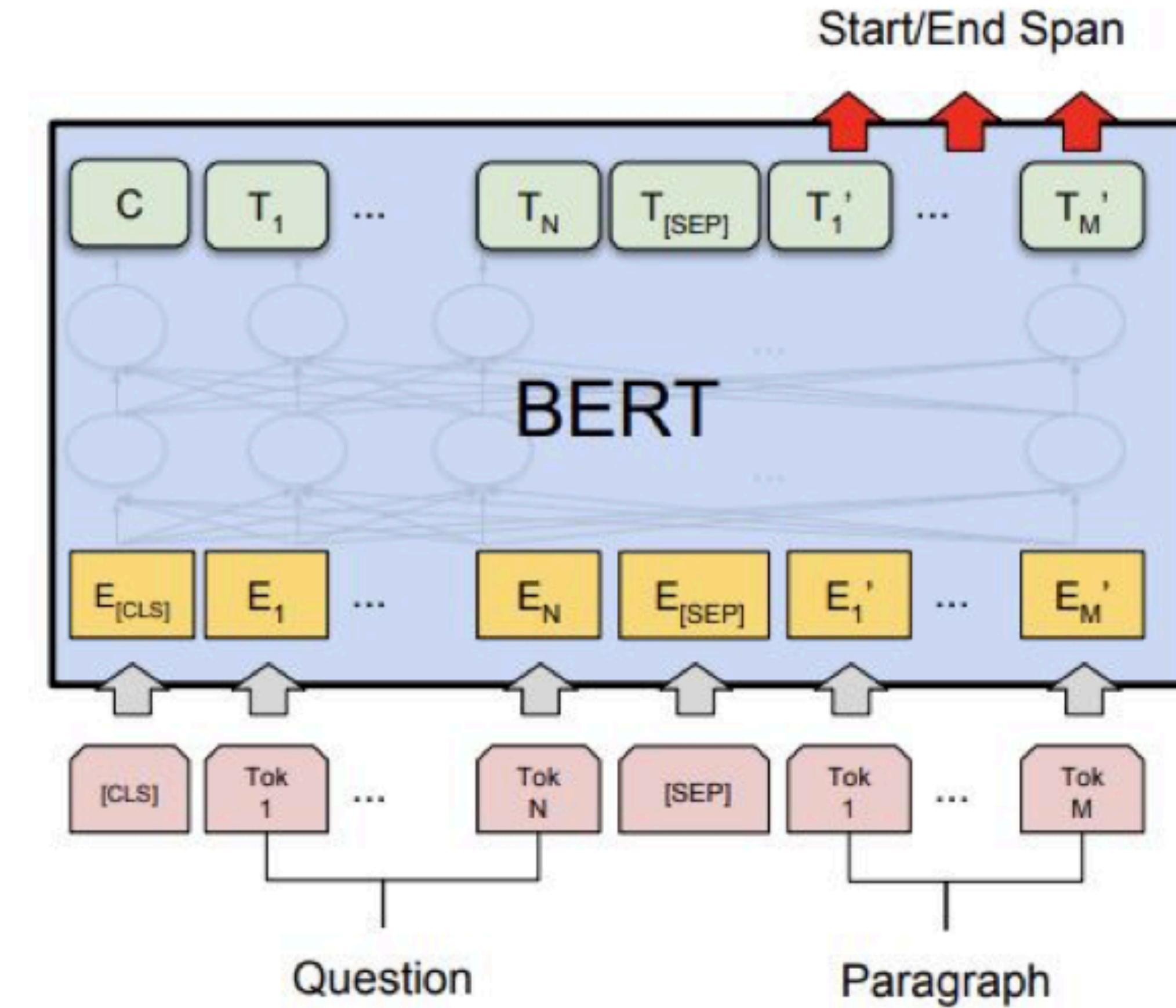
# Fine-tuning for single sentence classification



(b) Single Sentence Classification Tasks:  
SST-2, CoLA

Fig from J. Devlin BERT slides

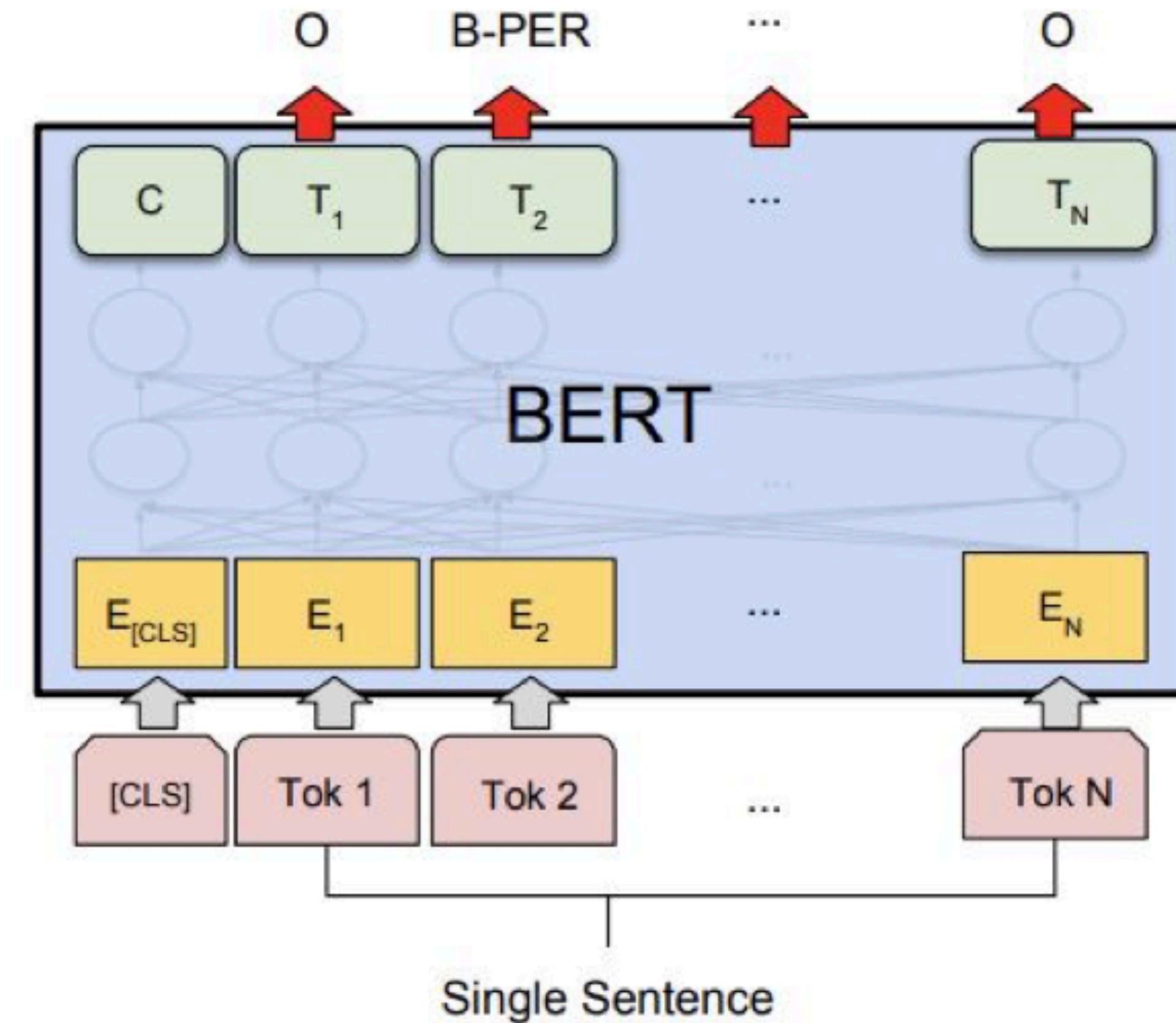
# Fine-tuning for question answering tasks



(c) Question Answering Tasks:  
SQuAD v1.1

Fig from J. Devlin BERT slides

# Fine-tuning for single sentence tagging tasks



(d) Single Sentence Tagging Tasks:  
CoNLL-2003 NER

Fig from J. Devlin BERT slides

# GLUE Results

System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	Average
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.9	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	88.1	91.3	45.4	80.0	82.3	56.0	75.2
BERT <sub>BASE</sub>	84.6/83.4	71.2	90.1	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>91.1</b>	<b>94.9</b>	<b>60.5</b>	<b>86.5</b>	<b>89.3</b>	<b>70.1</b>	<b>81.9</b>

## MultiNLI

Premise: Hills and mountains are especially sanctified in Jainism.

Hypothesis: Jainism hates nature.

Label: Contradiction

## CoLa

Sentence: The wagon rumbled down the road.

Label: Acceptable

Sentence: The car honked down the road.

Label: Unacceptable

# Directionality and training time

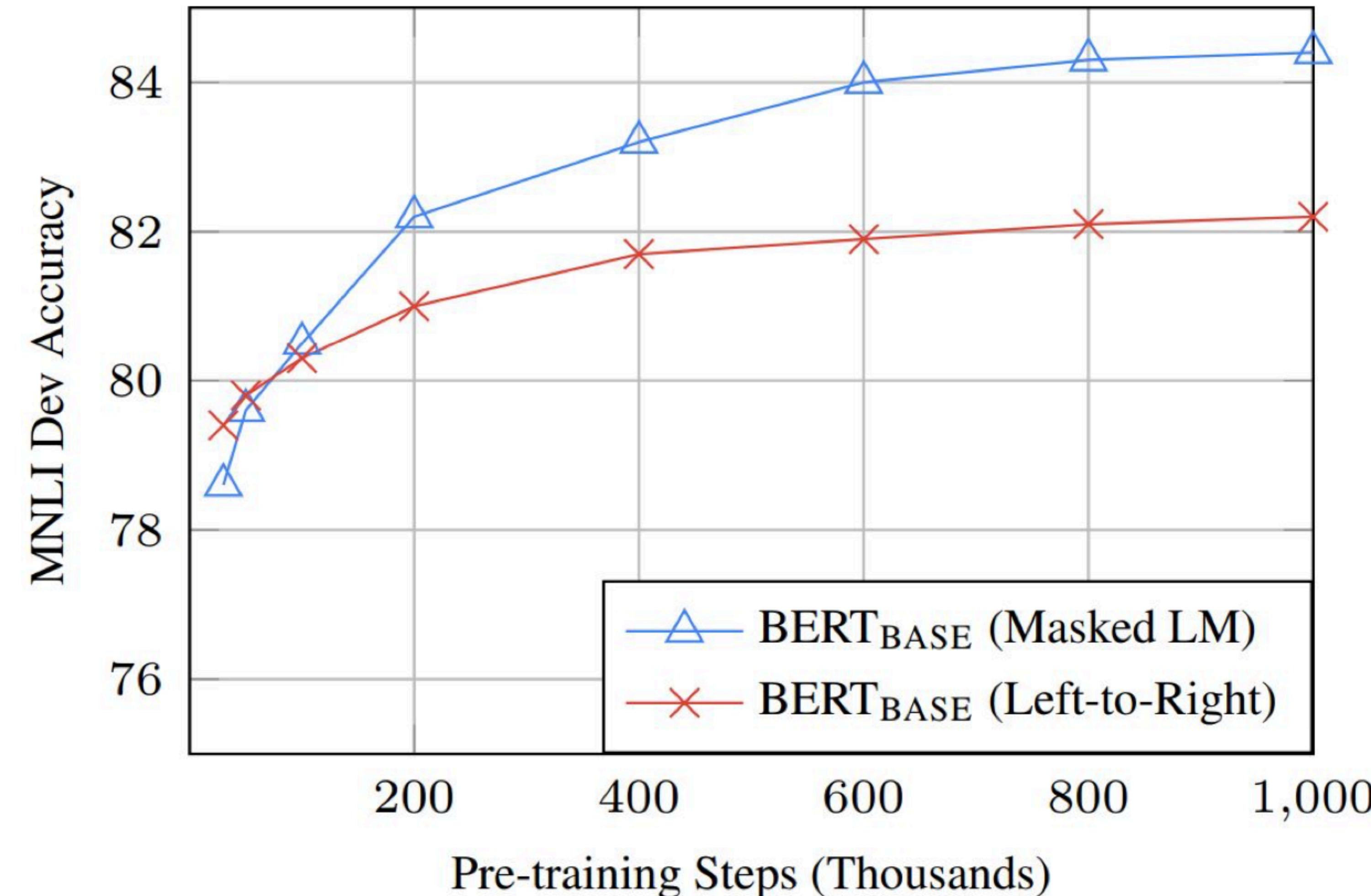


Fig from J. Devlin BERT slides

# Effect of Model Size

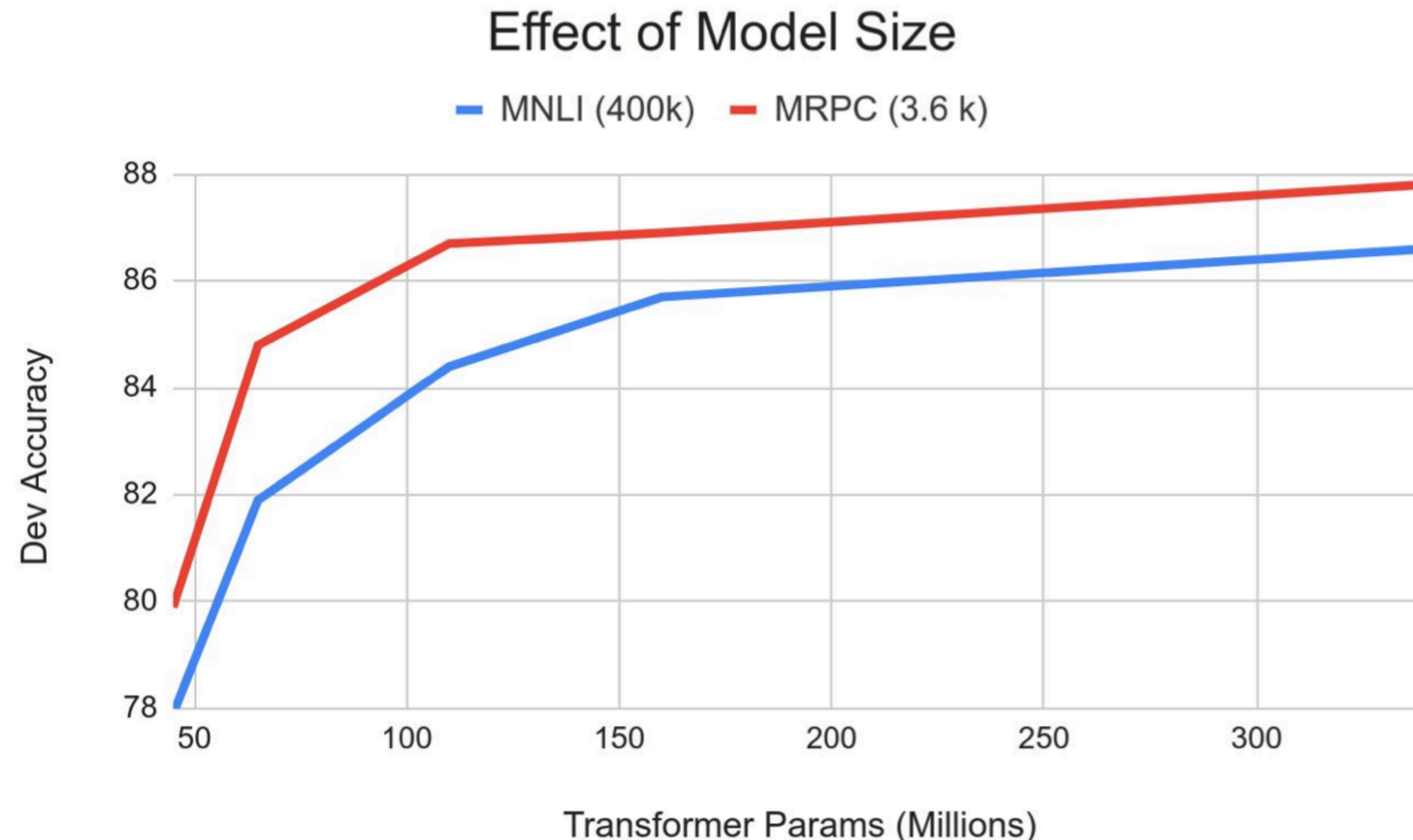


Fig from J. Devlin BERT slides

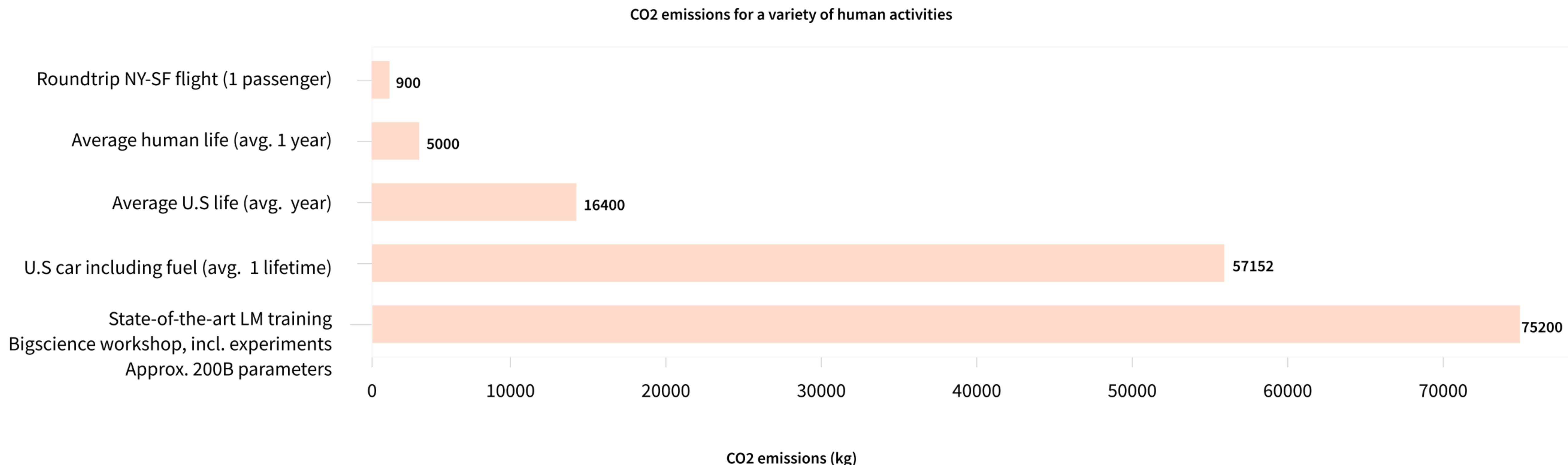
# Open Source Release

**BERT was successful due to full open-source release**

- BERT-base and BERT-large released under a permissive license (Apache 2.0)
- Model-only release (not part of a larger codebase): open source DL toolkits
- No dependencies except TensorFlow or PyTorch
- Abstracted so all you had to do was import a single module
- End-to-end examples to train SoTA models on many tasks
- Comprehensive README and readable, well-documented code
- Good support (for first few months)

# Environmental Impact

Model name	Number of parameters	Power consumption	CO <sub>2</sub> eq emissions
GPT-3	175B	1,287 MWh	502 tons
Gopher	280B	1,066 MWh	352 tons
OPT	175B	324 MWh	70 tons
BLOOM	176B	433 MWh	25 tons



# BERT Extensions

# RoBERTa

<https://arxiv.org/abs/1907.11692>

Liu+ 2019

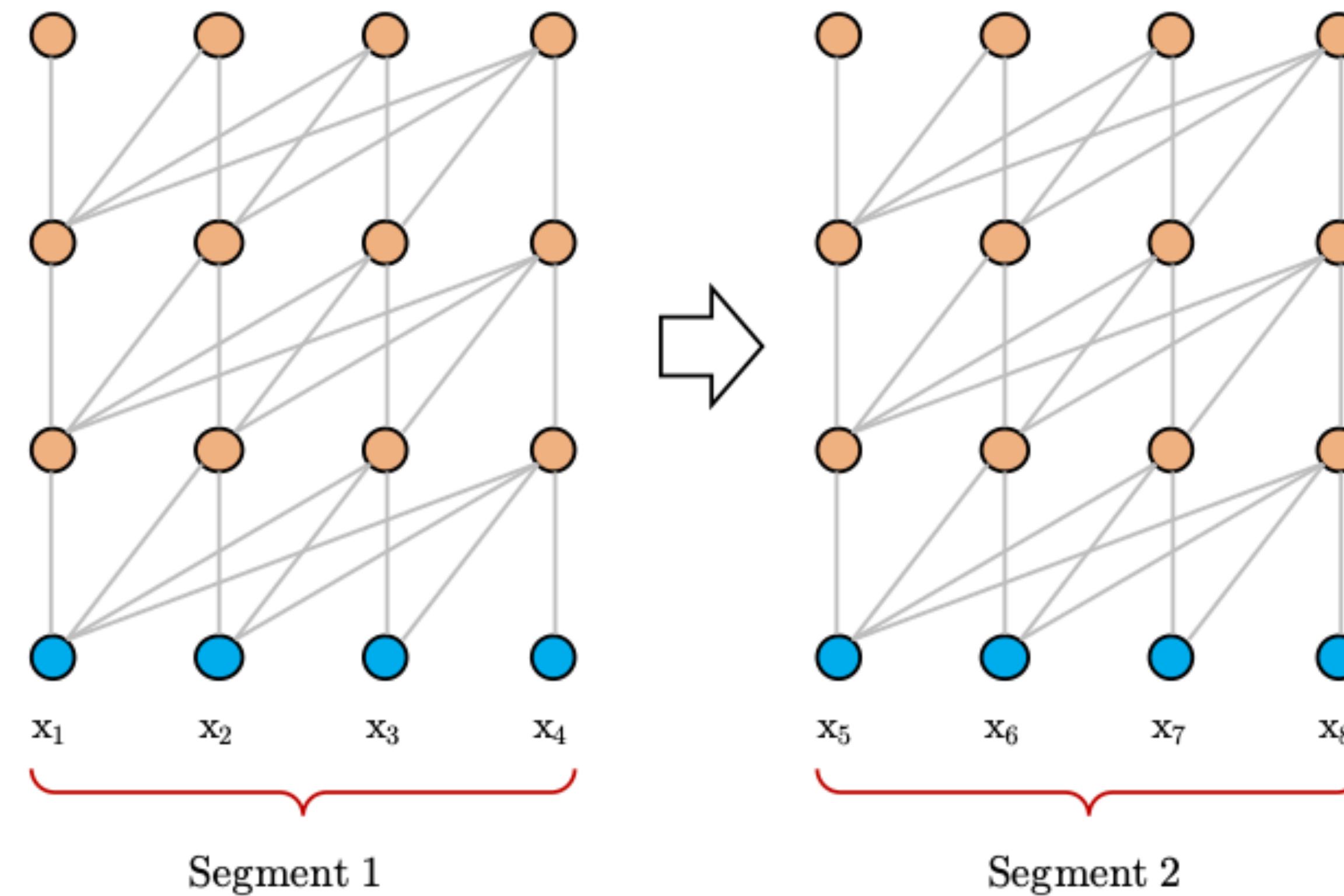
- Robustly optimized BERT pre-training: dynamic masking; train on text blocks
- Train BERT on more data and for more epochs
  - Even on same data, training for longer helps
  - More data leads to a better model
- Remove Next Sentence Prediction (NSP) loss

	MNLI	QNLI	QQP	RTE	SST	MRPC	CoLA	STS	WNLI	Avg
<i>Single-task single models on dev</i>										
BERT <sub>LARGE</sub>	86.6/-	92.3	91.3	70.4	93.2	88.0	60.6	90.0	-	-
XLNet <sub>LARGE</sub>	89.8/-	93.9	91.8	83.8	95.6	89.2	63.6	91.8	-	-
RoBERTa	<b>90.2/90.2</b>	<b>94.7</b>	<b>92.2</b>	<b>86.6</b>	<b>96.4</b>	<b>90.9</b>	<b>68.0</b>	<b>92.4</b>	<b>91.3</b>	-

# Transformer-XL

Dai+ 2019

- Vanilla Model



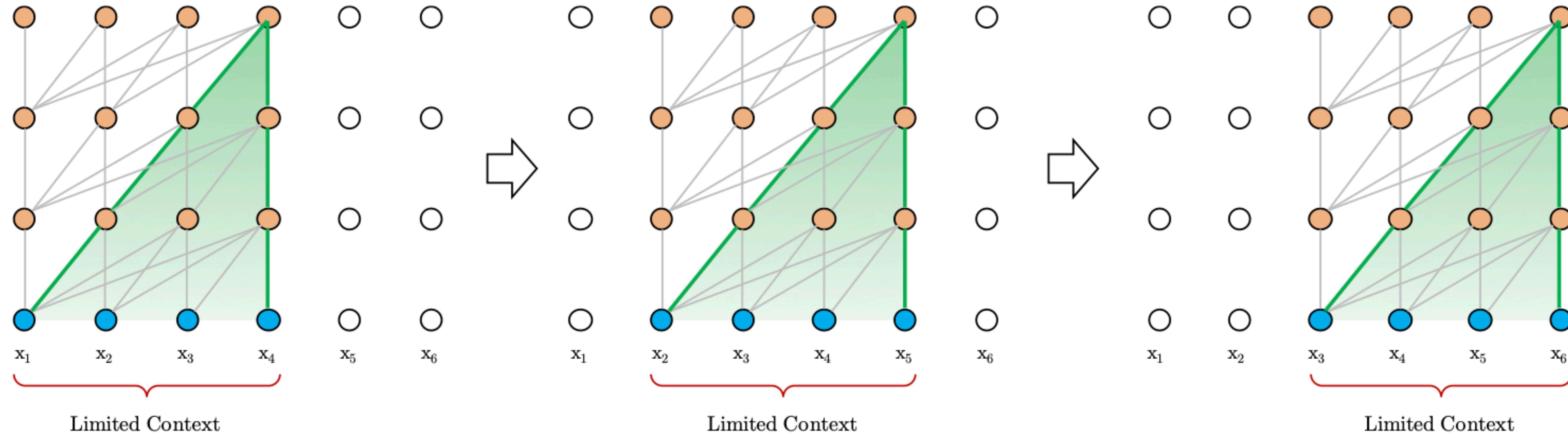
(a) Train phase.

<https://arxiv.org/abs/1901.02860>

# Transformer-XL

## Dai+ 2019

- Vanilla Model



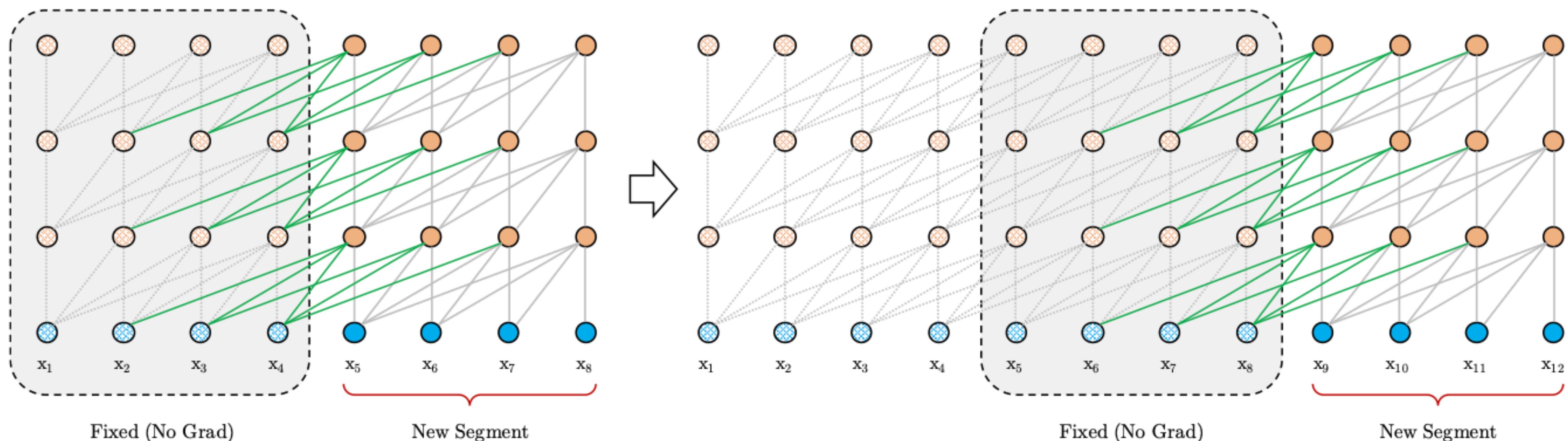
(b) Evaluation phase.

<https://arxiv.org/abs/1901.02860>

# Transformer-XL

## Dai+ 2019

- Autoregressive LM (different from GPT)
- segment level recurrence (reuse states) + relative positional embeddings



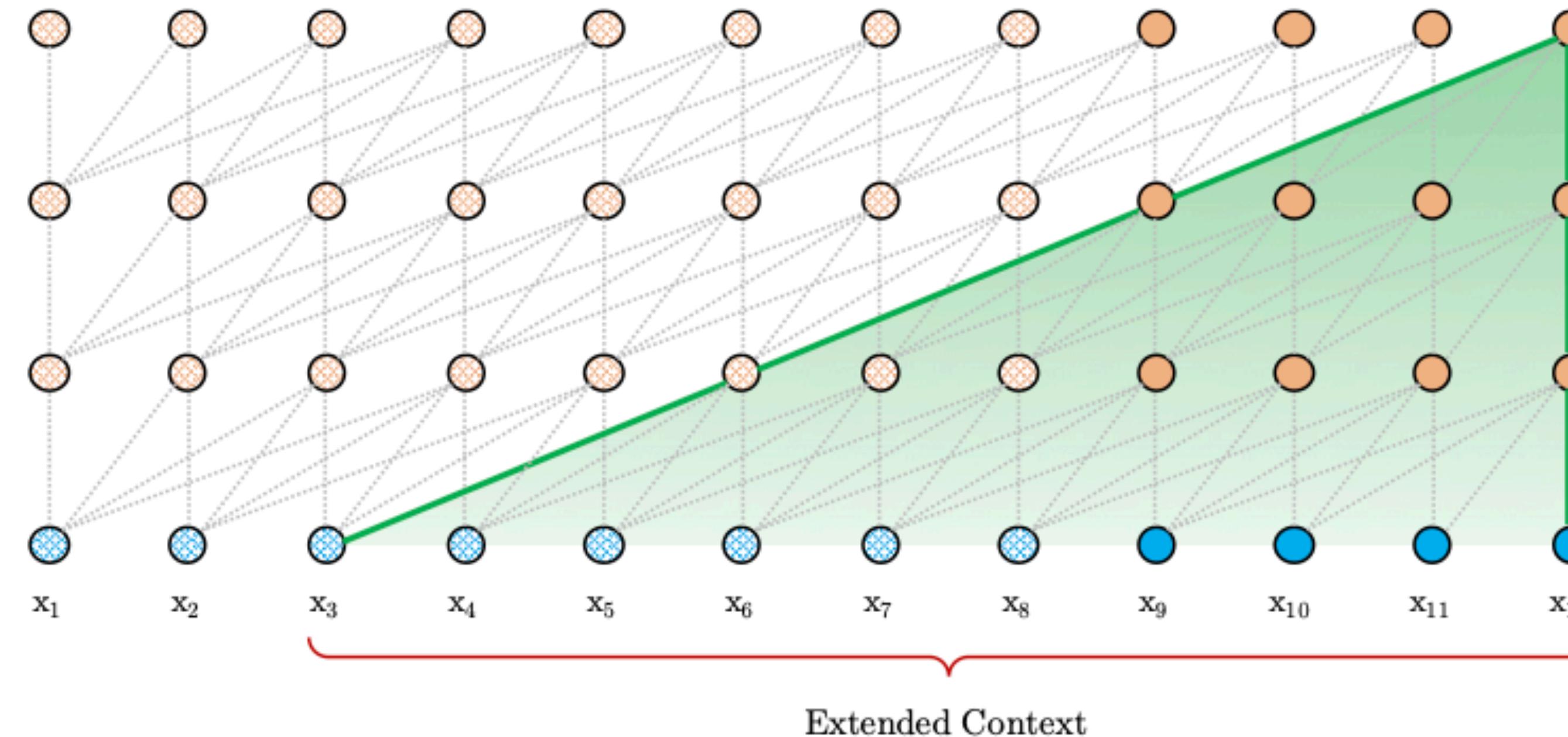
(a) Training phase.

<https://arxiv.org/abs/1901.02860>

# Transformer-XL

Dai+ 2019

- Autoregressive LM (different from GPT)



(b) Evaluation phase.

<https://arxiv.org/abs/1901.02860>

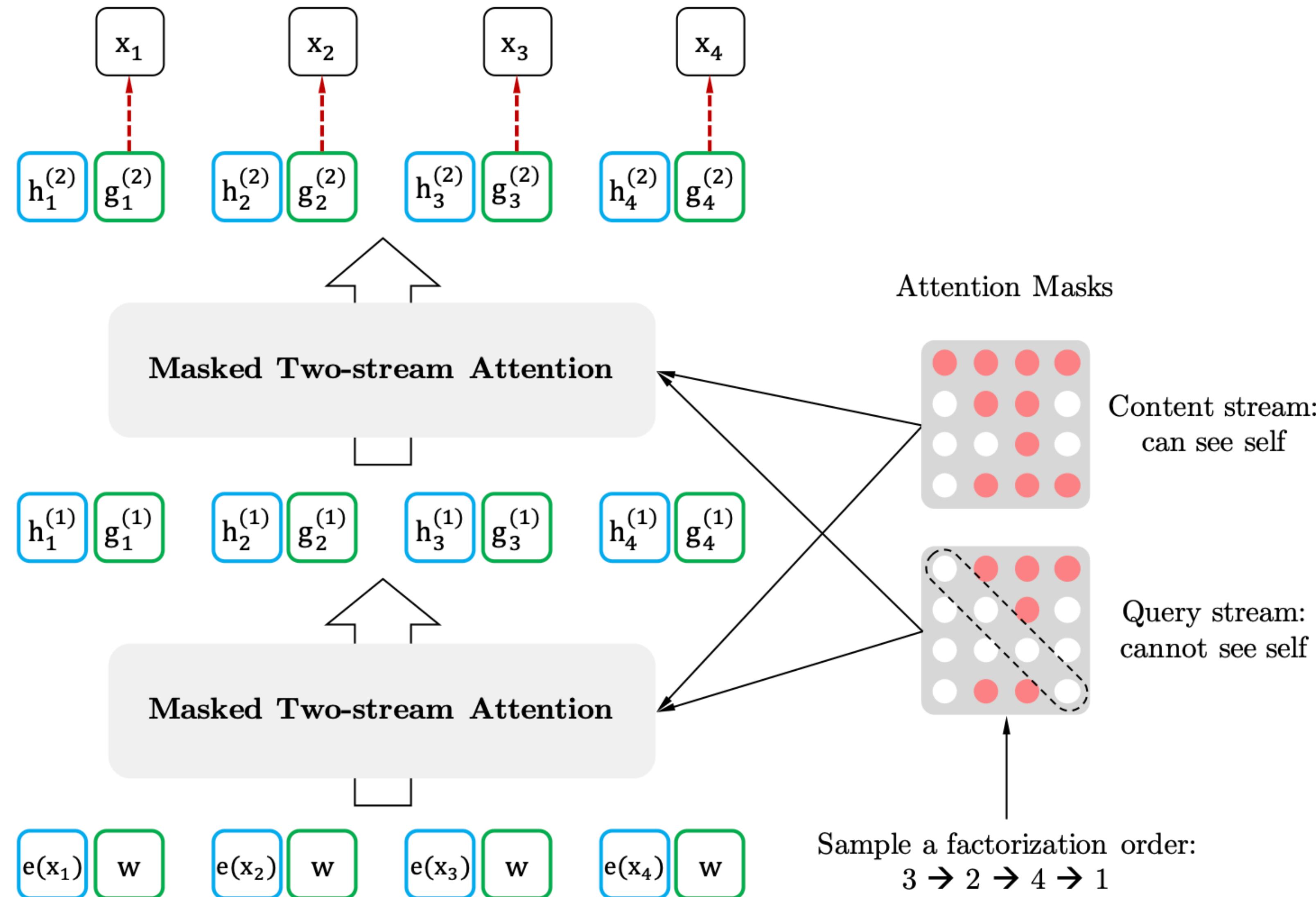
# XLNet

Yang+ 2019

<https://arxiv.org/abs/1906.08237>

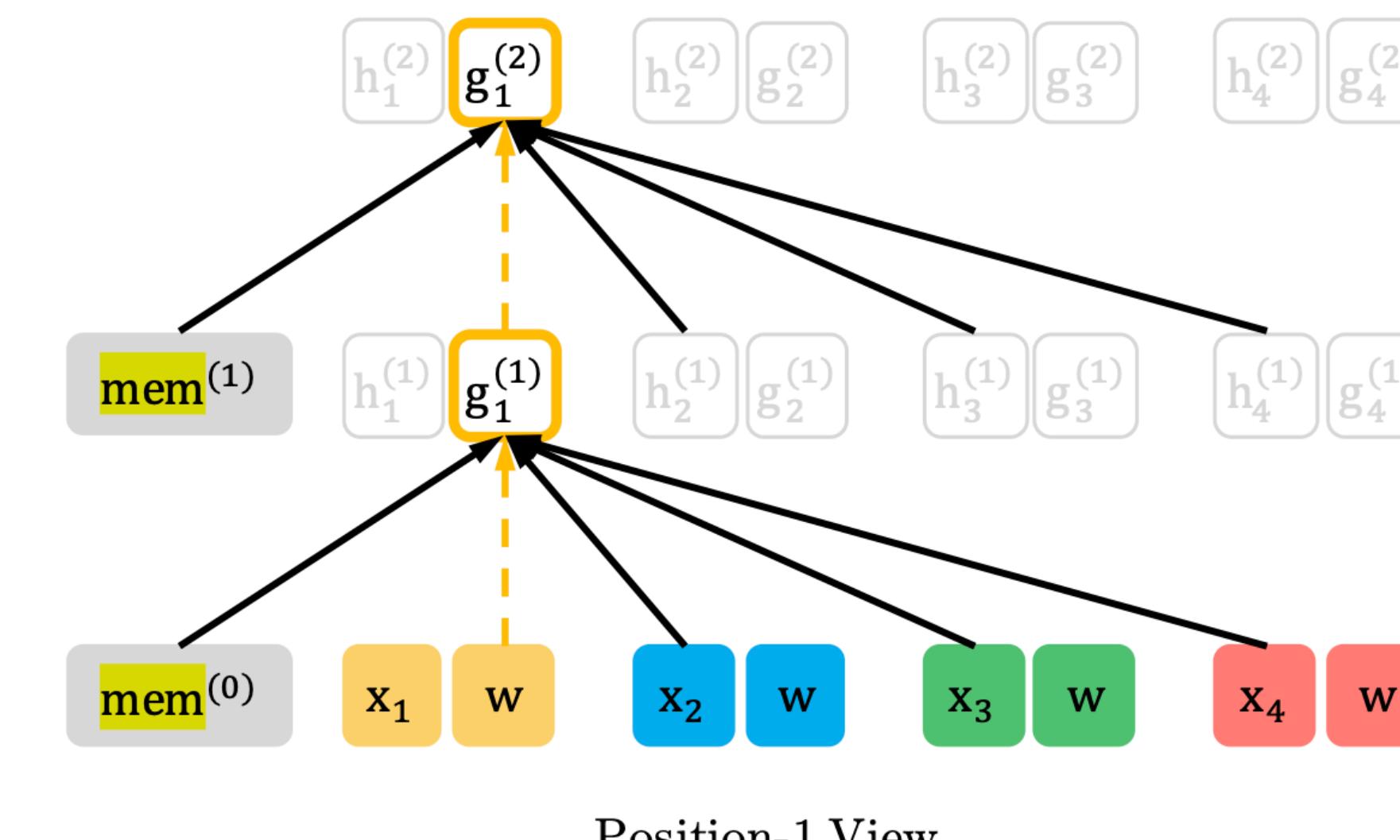
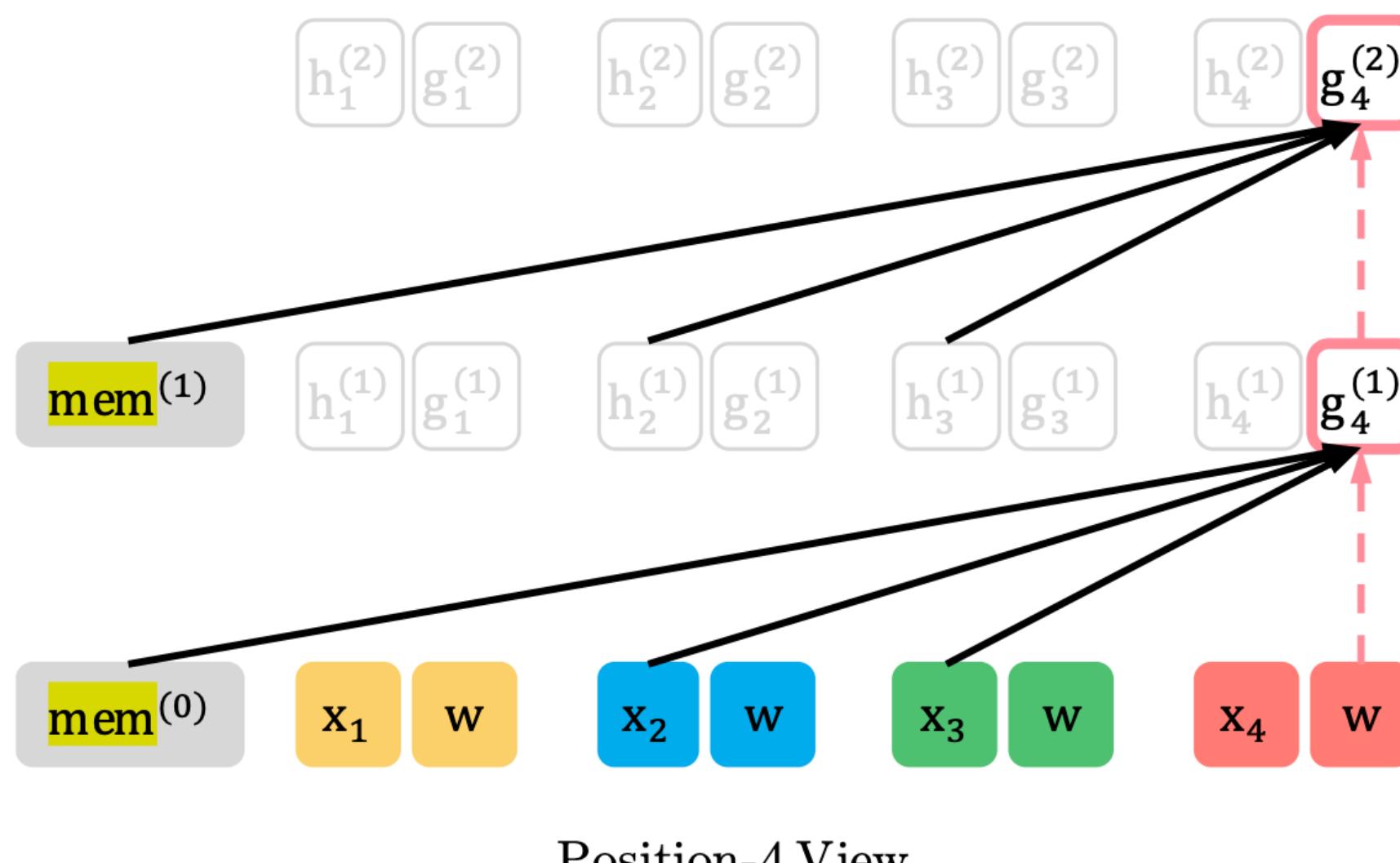
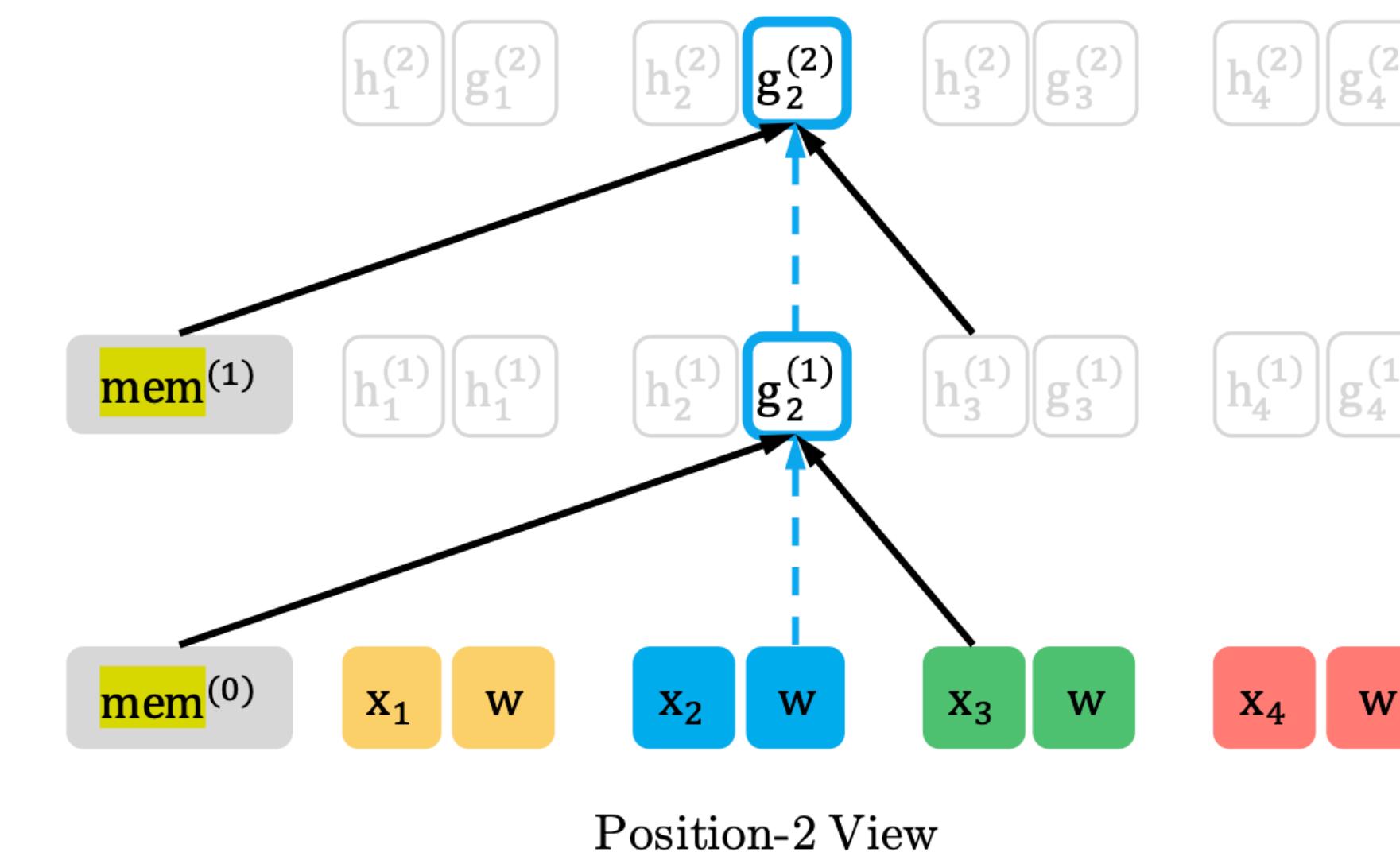
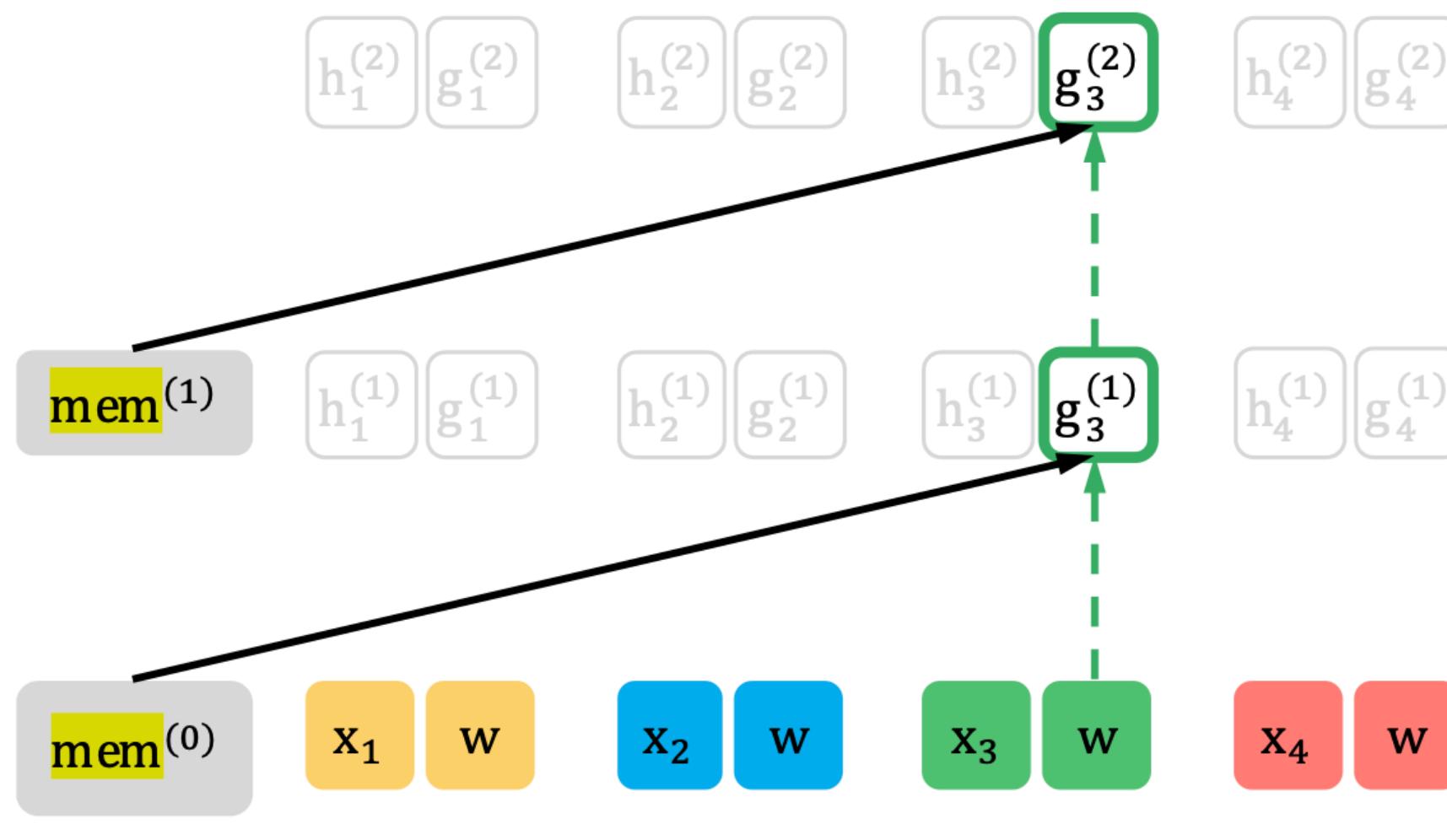
- Relative position embeddings (using auto-regressive TransformerXL)
  - Absolute attention: position 4 → 5; position 128 → 129
  - Relative attention: position  $t \rightarrow (t - 1)$
- Mask prediction over all token positions using permutation on factorization order (sample a factorization order: 3 → 2 → 1 → 4)
  - Two stream self-attention: standard and query on [MASK] token
  - Permute only factorization order, not sequence order

# XLNet



# XLNet

Split View of the Query Stream  
 (Factorization order: 3 → 2 → 4 → 1)



# XLNet

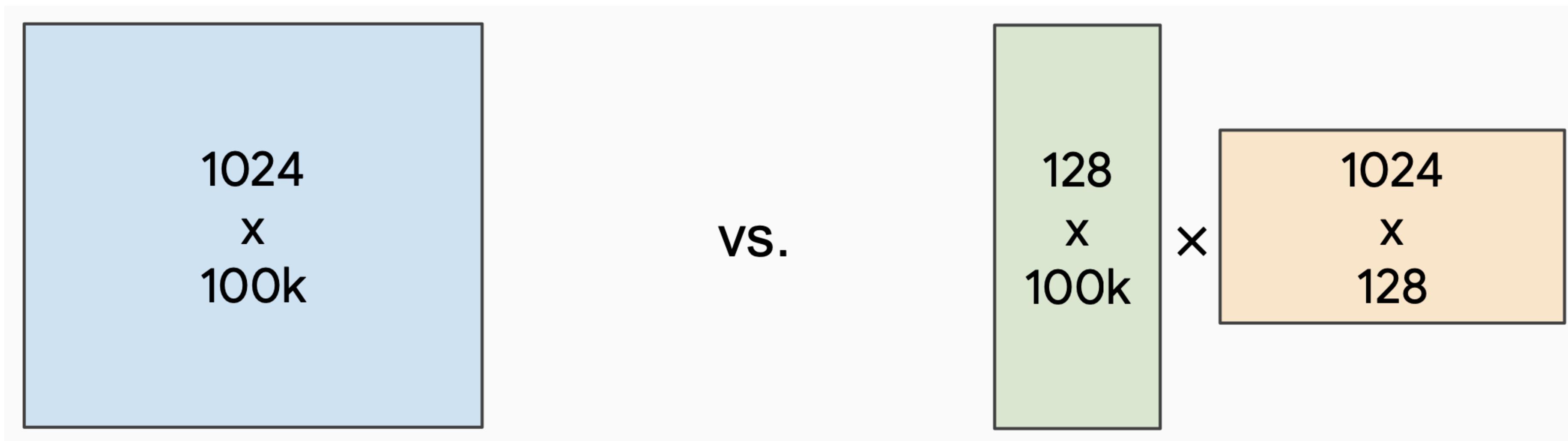
Model	MNLI	QNLI	QQP	RTE	SST-2	MRPC	CoLA	STS-B
<i>Single-task single models on dev</i>								
BERT [2]	86.6/-	92.3	91.3	70.4	93.2	88.0	60.6	90.0
RoBERTa [21]	90.2/90.2	94.7	92.2	<b>86.6</b>	96.4	<b>90.9</b>	68.0	92.4
XLNet	<b>90.8/90.8</b>	<b>94.9</b>	<b>92.3</b>	85.9	<b>97.0</b>	90.8	<b>69.0</b>	<b>92.5</b>

# ALBERT

## Lan+ 2019

<https://arxiv.org/abs/1909.11942>

- Factorized embedding parameterization
  - Use small embedding size (128) and project to Transformer hidden size (1024) using a parameter matrix



# ALBERT

<https://arxiv.org/abs/1909.11942>

- Cross-layer parameter sharing
  - $h^{\ell+1}$  parameters are shared with  $h^\ell$

Models	MNLI	QNLI	QQP	RTE	SST	MRPC	CoLA	STS
<i>Single-task single models on dev</i>								
BERT-large	86.6	92.3	91.3	70.4	93.2	88.0	60.6	90.0
XLNet-large	89.8	93.9	91.8	83.8	95.6	89.2	63.6	91.8
RoBERTa-large	90.2	94.7	<b>92.2</b>	86.6	96.4	<b>90.9</b>	68.0	92.4
ALBERT (1M)	90.4	95.2	92.0	88.1	96.8	90.2	68.7	92.7
ALBERT (1.5M)	<b>90.8</b>	<b>95.3</b>	<b>92.2</b>	<b>89.2</b>	<b>96.9</b>	<b>90.9</b>	<b>71.4</b>	<b>93.0</b>

# ALBERT

<https://arxiv.org/abs/1909.11942>

- Light on parameters; not necessarily faster than BERT

	Model	Parameters	SQuAD1.1	SQuAD2.0	MNLI	SST-2	RACE	Avg	Speedup
BERT	base	108M	90.4/83.2	80.4/77.6	84.5	92.8	68.2	82.3	4.7x
	large	334M	92.2/85.5	85.0/82.2	86.6	93.0	73.9	85.2	1.0
ALBERT	base	12M	89.3/82.3	80.0/77.1	81.6	90.3	64.0	80.1	5.6x
	large	18M	90.6/83.9	82.3/79.4	83.5	91.7	68.5	82.4	1.7x
	xlarge	60M	92.5/86.1	86.1/83.1	86.4	92.4	74.8	85.5	0.6x
	xxlarge	235M	<b>94.1/88.3</b>	<b>88.1/85.1</b>	<b>88.0</b>	<b>95.2</b>	<b>82.3</b>	<b>88.7</b>	0.3x

# T5

<https://arxiv.org/abs/1910.10683>

## Raffel+ 2019

- Ablation study on many aspects of pre-training and fine-tuning
  - Model size (bigger is better; 11B parameters)
  - Amount of training data (more is better)
  - Domain / cleanliness of training data [-ve]
  - Pre-training objective (e.g. span length of masked text) [-ve]
  - Ensemble models [-ve]
  - Fine-tuning recipe (e.g. only allow top k layers to fine-tune) [-ve]
  - Multi-task training [-ve]

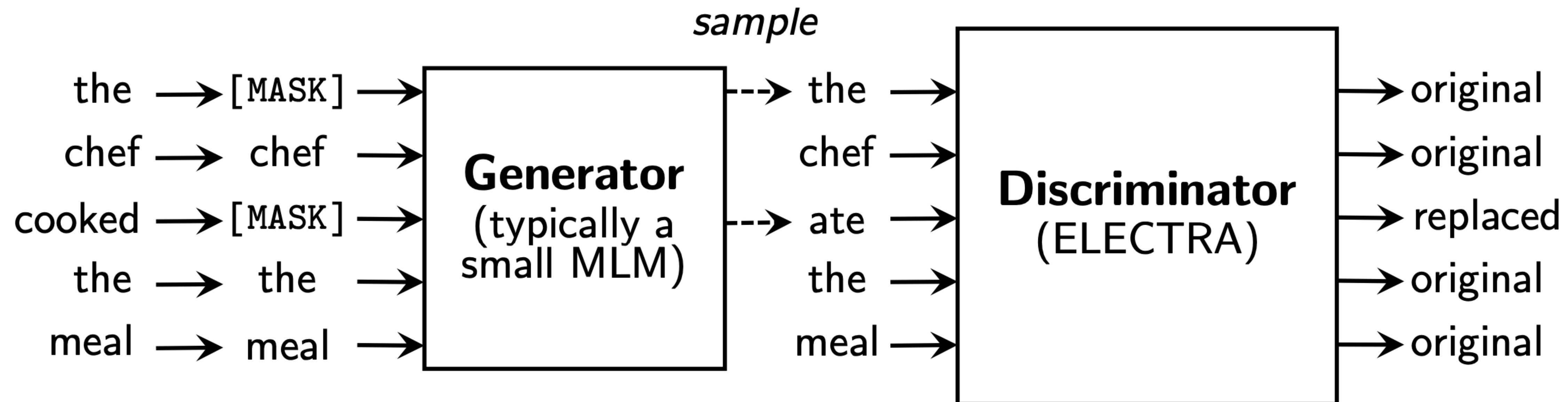


# ELECTRA

Clark+ 2020

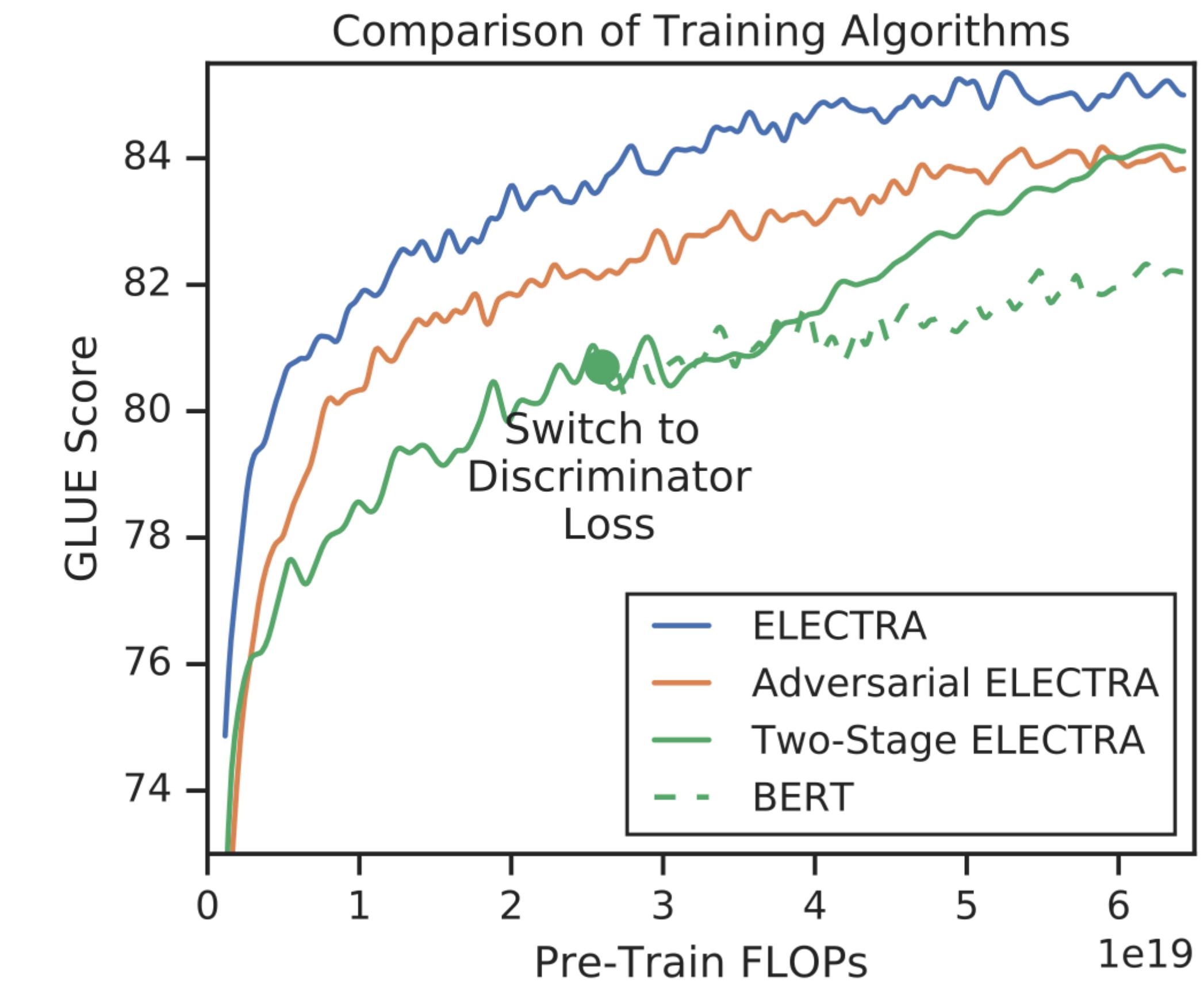
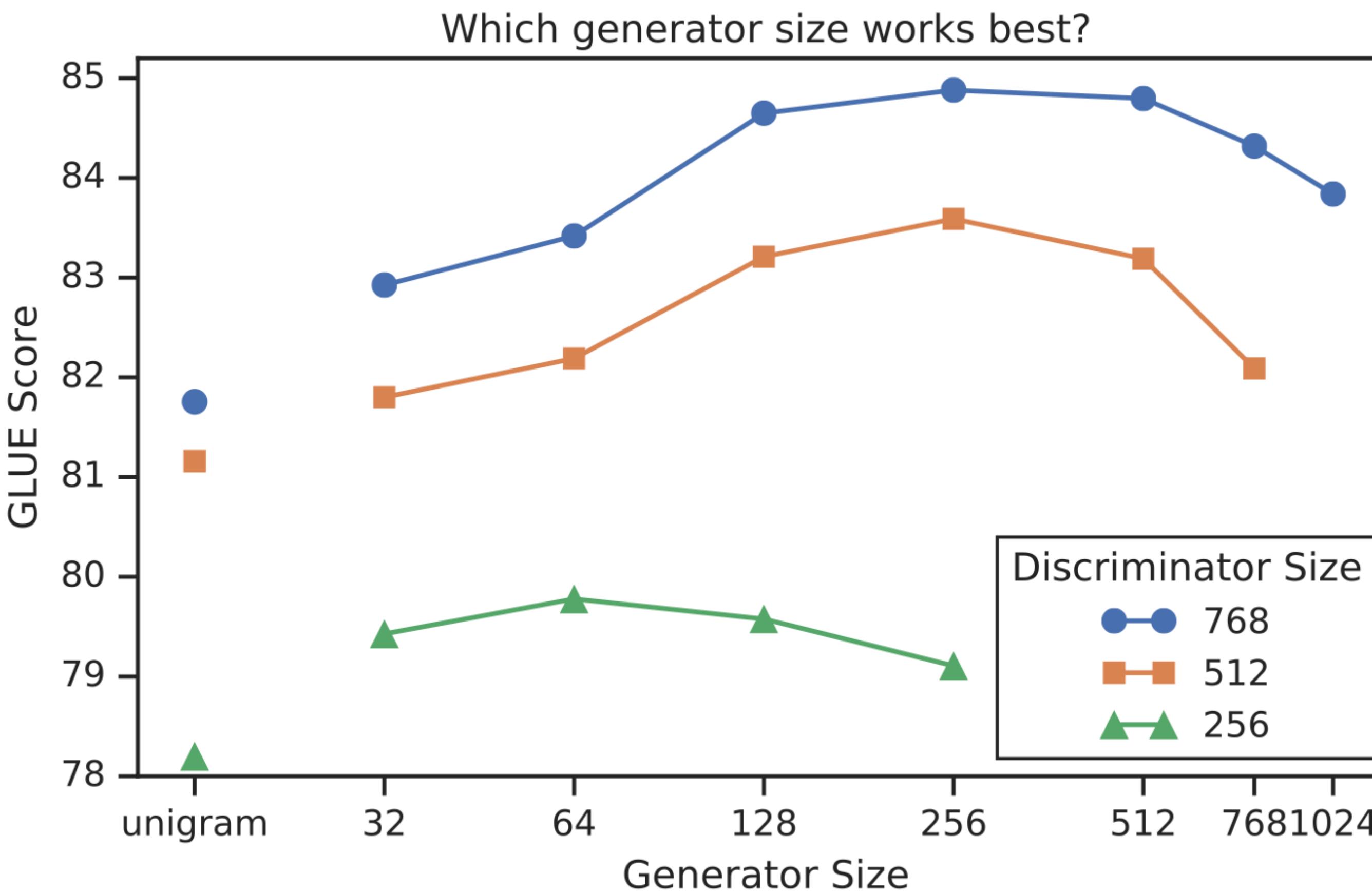
<https://arxiv.org/abs/2003.10555>

- Train model to discriminate locally plausible text from real text



# ELECTRA

<https://arxiv.org/abs/2003.10555>



# ELECTRA

<https://arxiv.org/abs/2003.10555>

Model	Train FLOPs	Params	CoLA	SST	MRPC	STS	QQP	MNLI	QNLI	RTE	Avg.
BERT	1.9e20 (0.27x)	335M	60.6	93.2	88.0	90.0	91.3	86.6	92.3	70.4	84.0
RoBERTa-100K	6.4e20 (0.90x)	356M	66.1	95.6	<b>91.4</b>	92.2	92.0	89.3	94.0	82.7	87.9
RoBERTa-500K	3.2e21 (4.5x)	356M	68.0	96.4	90.9	92.1	92.2	90.2	94.7	86.6	88.9
XLNet	3.9e21 (5.4x)	360M	69.0	<b>97.0</b>	90.8	92.2	92.3	90.8	94.9	85.9	89.1
BERT (ours)	7.1e20 (1x)	335M	67.0	95.9	89.1	91.2	91.5	89.6	93.5	79.5	87.2
ELECTRA-400K	7.1e20 (1x)	335M	<b>69.3</b>	96.0	90.6	92.1	<b>92.4</b>	90.5	94.5	86.8	89.0
ELECTRA-1.75M	3.1e21 (4.4x)	335M	69.1	96.9	90.8	<b>92.6</b>	<b>92.4</b>	<b>90.9</b>	<b>95.0</b>	<b>88.0</b>	<b>89.5</b>

# Other BERT Extensions

- Many many extensions to BERT; too many to cover here; mostly pre-training
  - Auto-regressive BERT variants (BART; XLM; etc.)
  - SpanBERT; Entity-based BERT (LUKE; SpanLinkBERT)
  - Mainly training on more data, or different data, slight variants (Megatron)
- Efficient fine-tuning (covered separately)
- Efficient inference
  - Distillation of BERT models (covered separately)