CMPT 983

Grounded Natural Language Understanding

February 28, 2021

Pretraining with transformers

Today

- Pretraining with transformers
  - Review of transformers
  - Review of language pretraining with transformers (BERT)
  - Multimodal transformers

Review of Transformers

Transformers are hot!

10 Novel Applications using Transformers [DL]

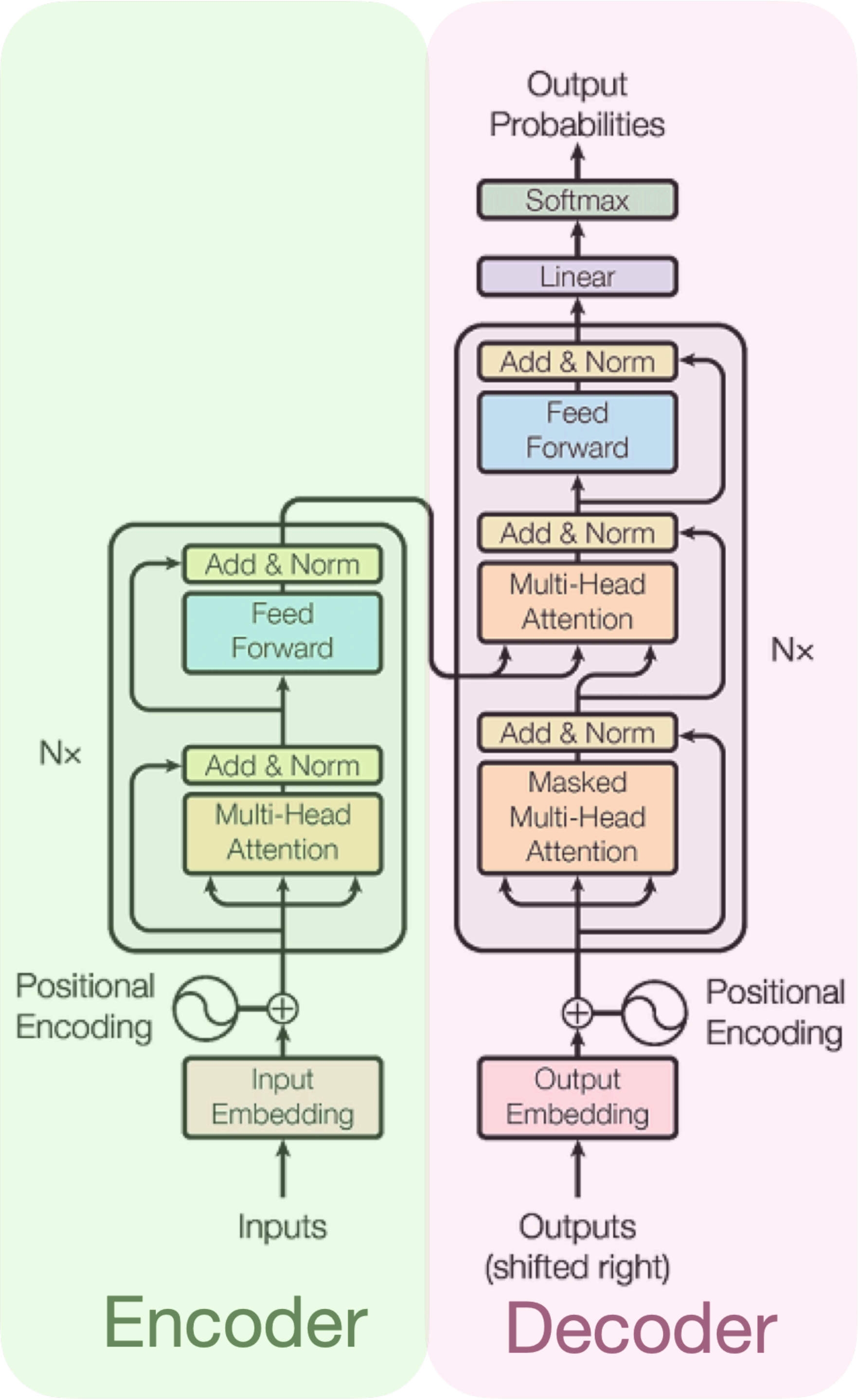
Transformers have had a lot of success in training neural language models. In the past few weeks, we've seen several trending papers with code applying Transformers to new types of task:

- **Transformer for Image Synthesis** - Esser et al. (2020)
- **Transformer for Multi-Object Tracking** - Sun et al. (2020)
- **Transformer for Music Generation** - Hsiao et al. (2021)
- **Transformer for Dance Generation with Music** - Huang et al. (2021)
- **Transformer for 3D Object Detection** - Bhattacharyya et al. (2021)
- **Transformer for Point-Cloud Processing** - Guo et al. (2020)
- **Transformer for Time-Series Forecasting** - Lim et al. (2020)
- **Transformer for Vision-Language Modeling** - Zhang et al. (2021)
- **Transformer for Lane Shape Prediction** - Liu et al. (2020)
- **Transformer for End-to-End Object Detection** - Zhu et al. (2021)

PapersWithCode newsletter (1/20/2021)
https://paperswithcode.com/newsletter/3

Transformers

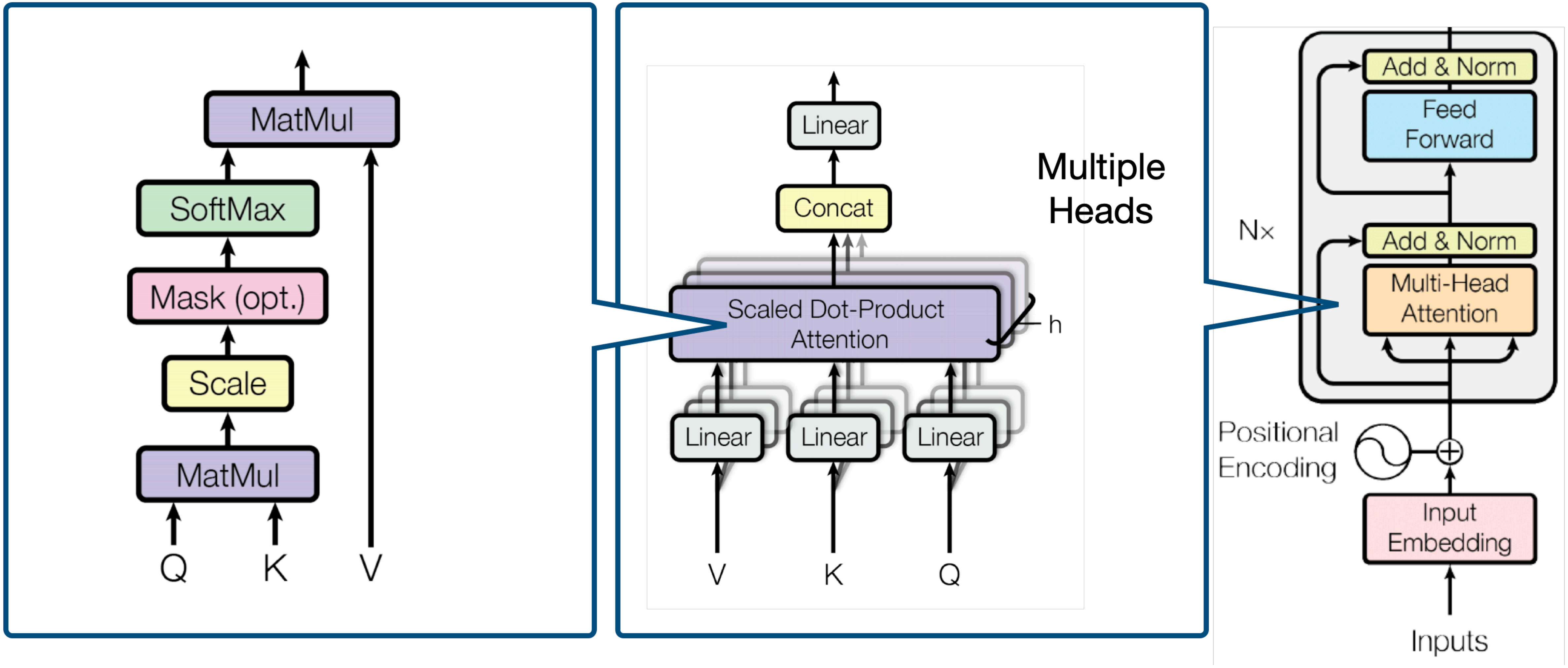
- NIPS'17: Attention is All You Need
- Originally proposed for NMT (encoder-decoder framework)
- Key idea: **Multi-head self-attention**
- No recurrence structure so training can be parallelized



Modelling Sequences -- Transformers

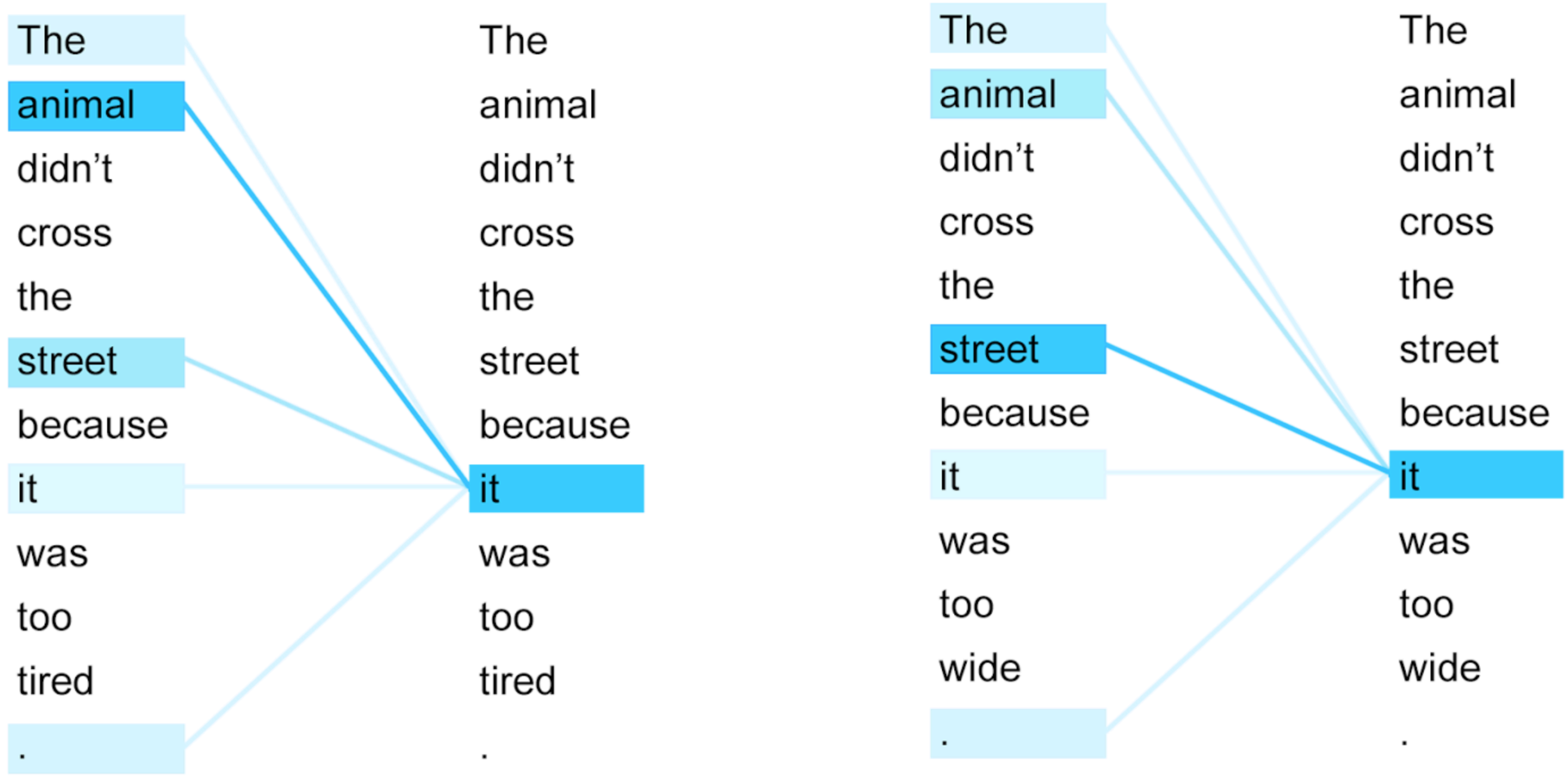
Scaled Dot-Product Attention

self-attention



Self-attention

- Attention (correlation) with different parts of itself



- Transformers: modules with scaled dot-product self-attention

Types of attention scores

Attention function, f

$$a_i = g(\boldsymbol{c}_i, \boldsymbol{z})$$
$$\boldsymbol{\alpha} = \mathrm{softmax}(\boldsymbol{a})$$
$$\hat{\boldsymbol{c}} = \sum_{i=1}^{k} \alpha_i \, \boldsymbol{c}_i$$

- Dot-product attention:
  $g(c_i, z) = z^\top c_i$
- Scaled dot-product attention:
  $g(c_i, z) = z^\top c_i / \sqrt{d}$
- Bilinear / multiplicative attention:
  $g(c_i, z) = z^\top W c_i \in \mathbb{R}$
  where W is a weight matrix
- Additive attention (essentially MLP):
  $g(c_i, z) = v^\top \tanh\left(W_1 c_i + W_2 z\right)$
  where W_1 , W_2 are weight matrices
  and v is a weight vector

Query-key-value view of attention

Attention function, f

$$a_i = g(\boldsymbol{c}_i, \boldsymbol{z})$$
$$\boldsymbol{\alpha} = \text{softmax}(\boldsymbol{a})$$
$$\hat{\boldsymbol{c}} = \sum_{i=1}^{k} \alpha_i \, \boldsymbol{c}_i$$

→

Attention function, f

$$a_i = g(\boldsymbol{k}_i, \boldsymbol{q})$$
$$\boldsymbol{\alpha} = \text{softmax}(\boldsymbol{a})$$
$$\hat{v} = \sum_{i=1}^{k} \alpha_i \, \boldsymbol{v}_i$$

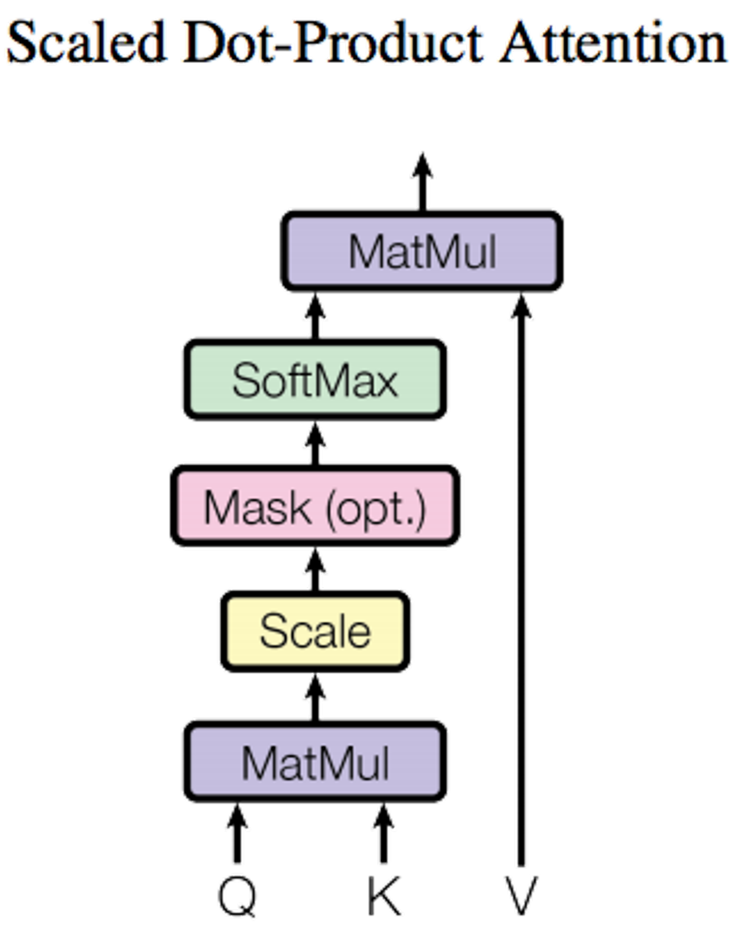
Projected query,key,value →

$$\boldsymbol{q} = W_Q \, \boldsymbol{z}$$
$$\boldsymbol{k}_i = W_K \, \boldsymbol{c}_i$$
$$\boldsymbol{v}_i = W_V \, \boldsymbol{c}_i$$

→ Matrix form

$$\boldsymbol{Q} = W_Q \, \boldsymbol{Z}^T$$
$$K = W_K \, C^T$$
$$V = W_V \, C^T$$

Scaled Dot Product Attention



Let $X \in \mathbb{R}^{M \times d_X}$ be a matrix of context vector

Let $Y \in \mathbb{R}^{N \times d_Y}$ be a matrix of input vectors

$\boldsymbol{SDPAttention(X, Y)}$:

$$Q = W_Q X^T \qquad W_Q \in \mathbb{R}^{d_h \times d_X}$$

$$K = W_K Y^T \qquad W_K \in \mathbb{R}^{d_h \times d_Y}$$

$$V = W_V Y^T \qquad W_K \in \mathbb{R}^{d_V \times d_Y}$$

$$Return \quad \hat{V} = softmax\left(\frac{Q^T K}{\sqrt{d_h}}\right) V$$

$\hat{V} \in \mathbb{R}^{M \times d_V}$ be a matrix of attended values

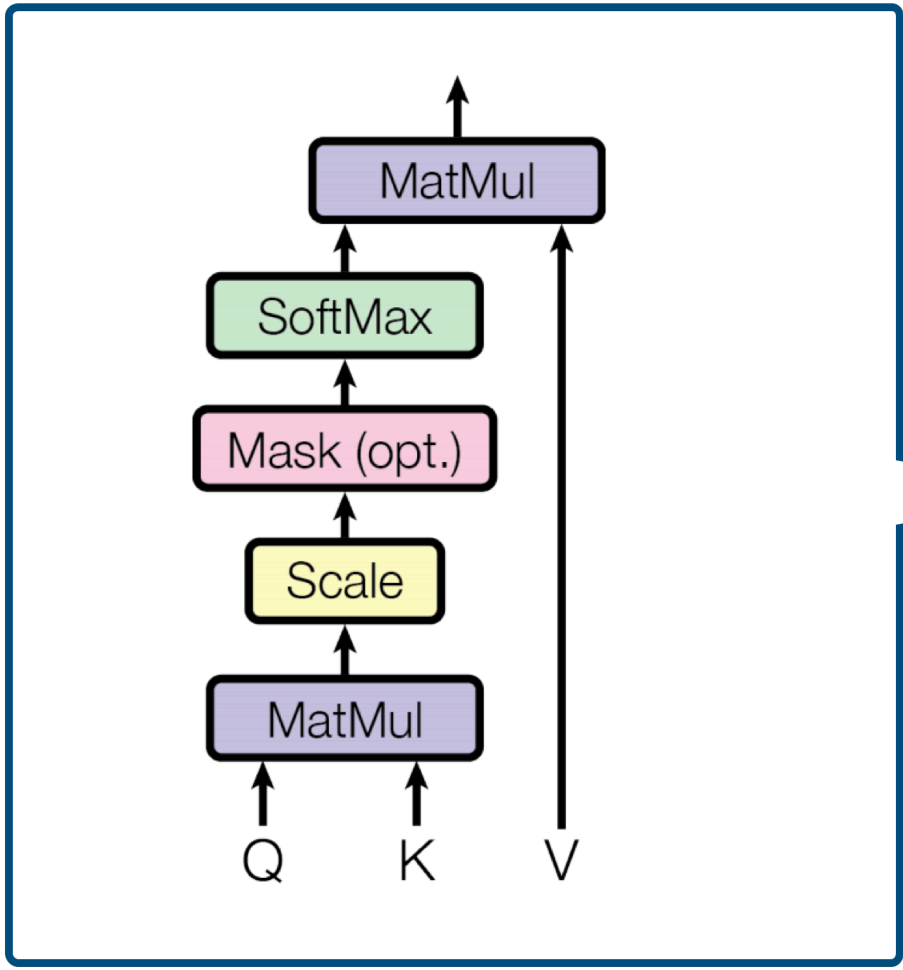
Attention Is All You Need https://arxiv.org/pdf/1706.03762.pdf

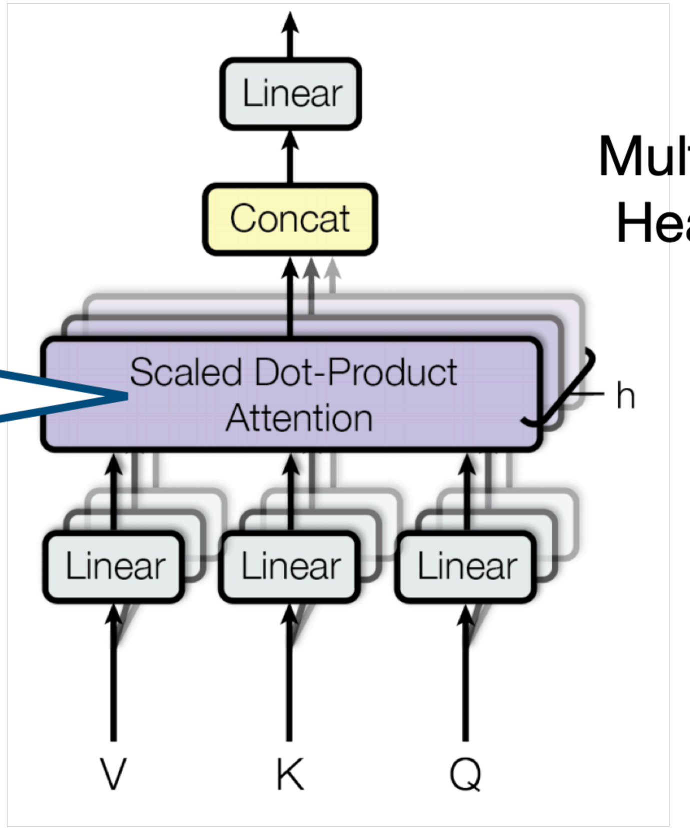
Modelling Sequences -- Transformers

Scaled Dot-Product Attention

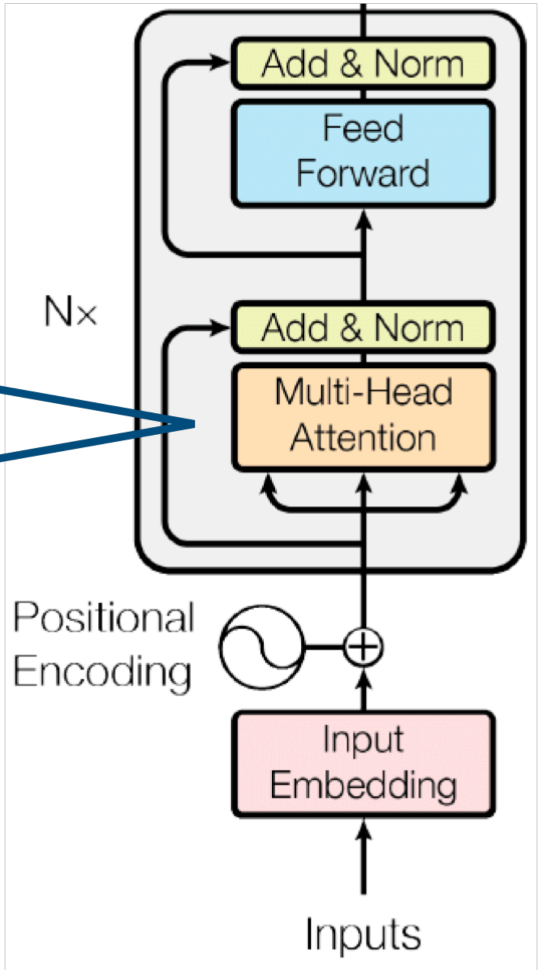
self-attention

$\boldsymbol{SDPAttention}(\boldsymbol{Y}, \boldsymbol{Y})$:





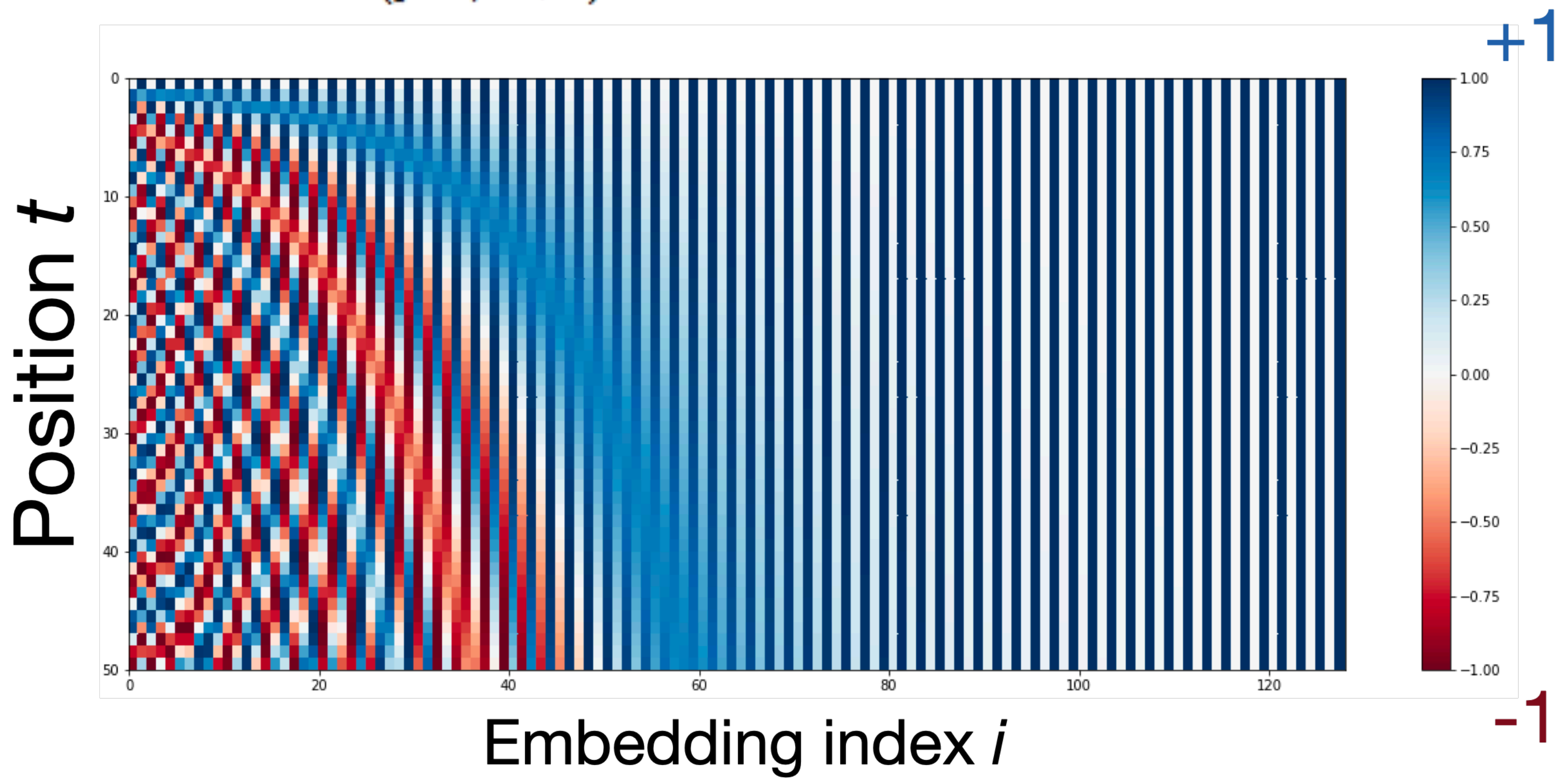
Multiple Heads



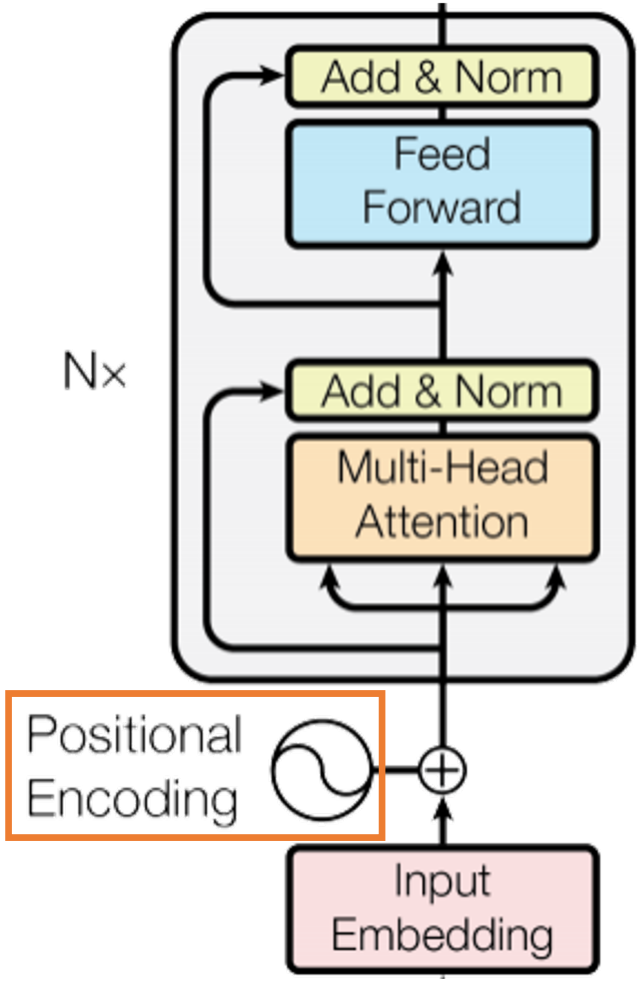
Transformers: Encoding position

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

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



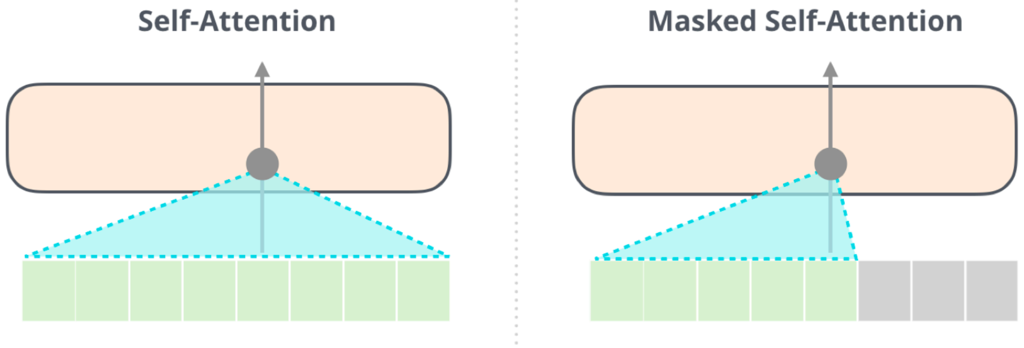
$SDPAttention(Y, Y)$:



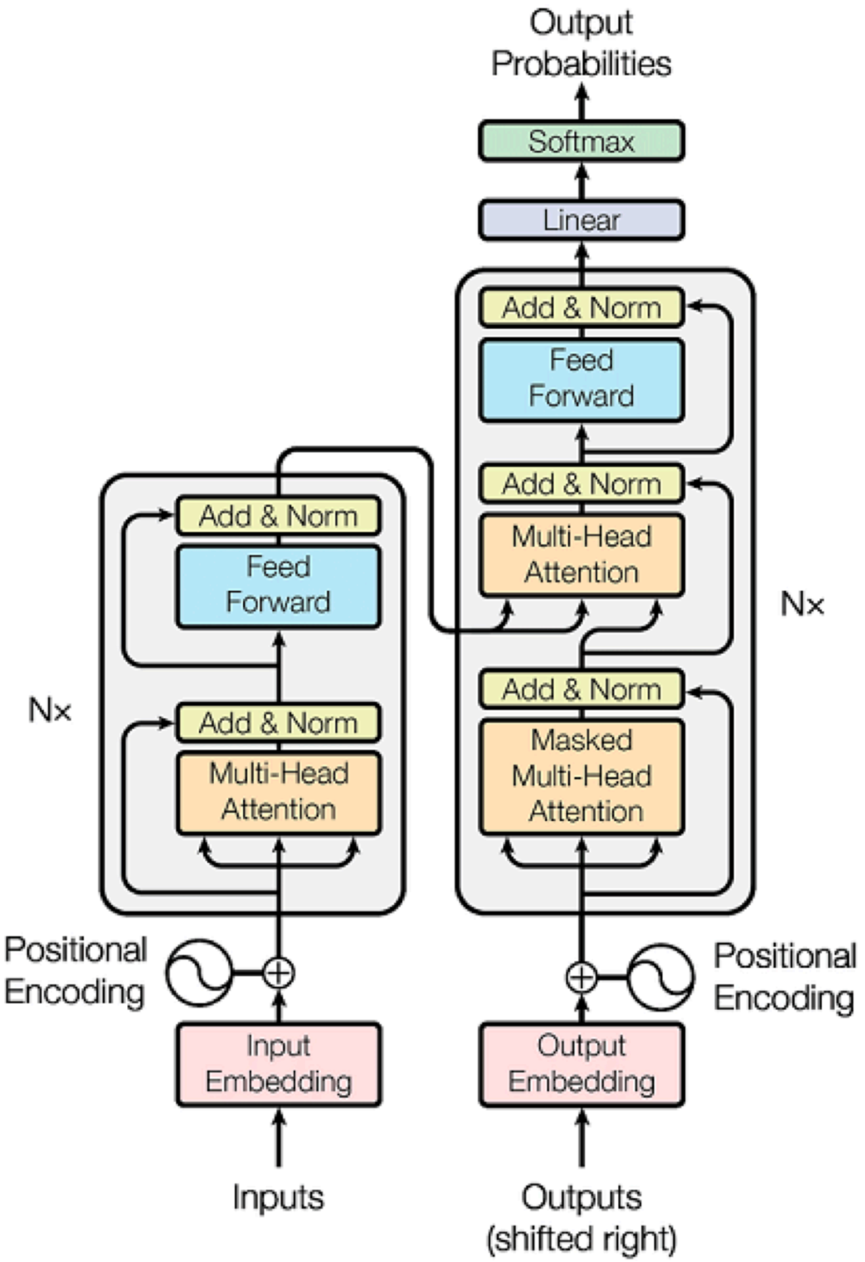
Attention Is All You Need https://arxiv.org/pdf/1706.03762.pdf

Transformers

- Encoder: Multi-headed self-attention
- Decoder
  - Masked self-attention



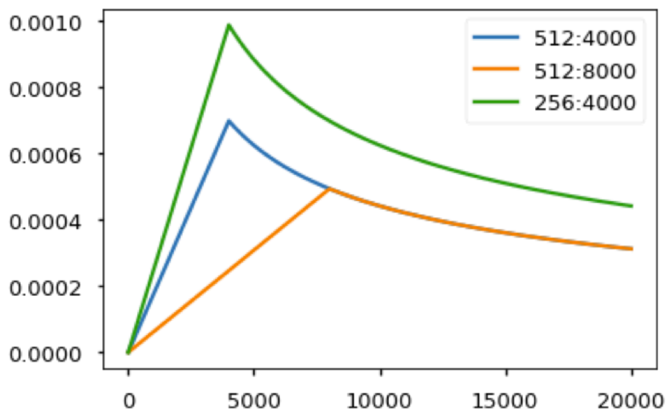
  - Cross attention
    - queries: previous decoder layer
    - keys/values: output of encoder
  - Autoregressive decoding



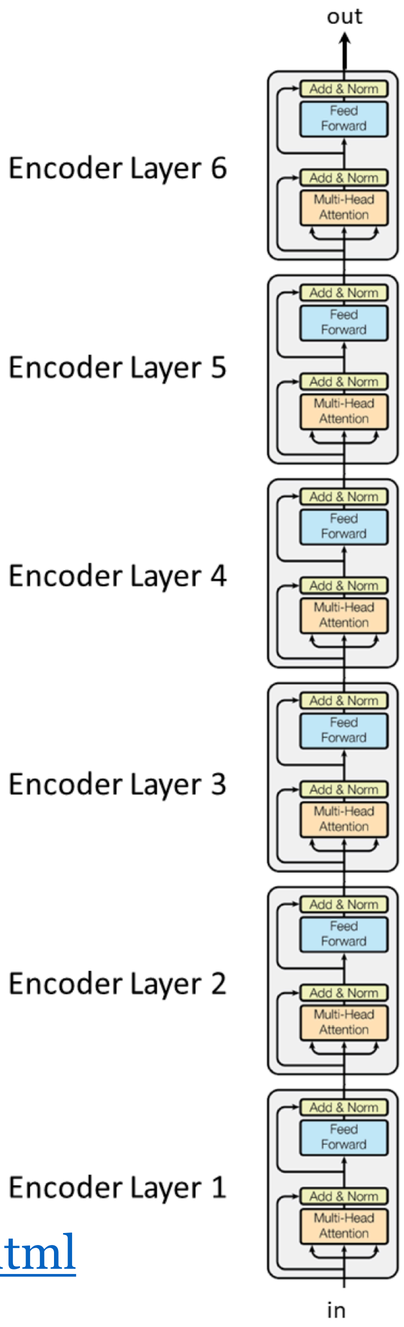
Encoder Decoder

Transformers

- Stacked into multi-layers
- Byte-pair encoding (BPE) / Word pieces
  - Subwords:
- Learning rate with warmup and decay



- Label smoothing: one-hot vector + noise



The Annotated Transformer http://nlp.seas.harvard.edu/2018/04/03/attention.html

A Jupyter notebook which explains how Transformer works line by line in PyTorch!

Other useful resources

Pytorch (https://pytorch.org/docs/stable/nn.html#transformer-layers

nn.Transformer:

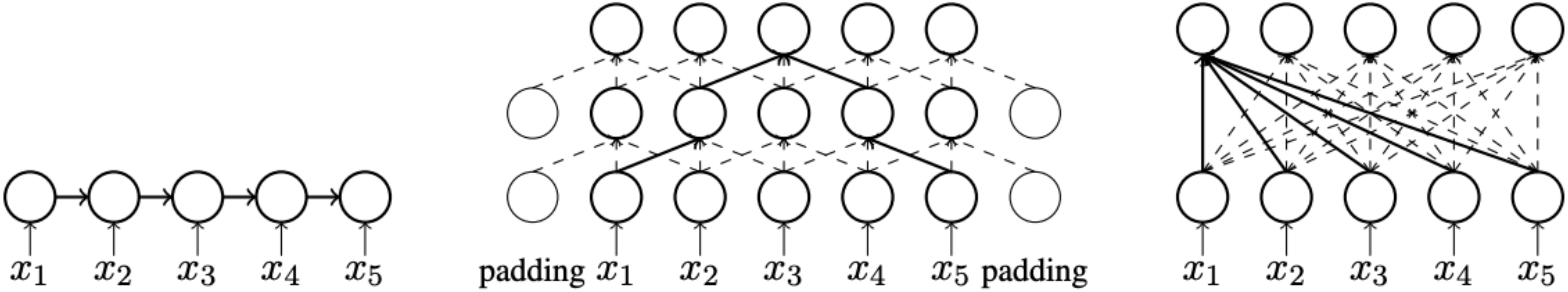
```
>>> transformer_model = nn.Transformer(nhead=16, num_encoder_layers=12)
>>> src = torch.rand((10, 32, 512))
>>> tgt = torch.rand((20, 32, 512))
>>> out = transformer_model(src, tgt)
```

nn.TransformerEncoder:

```
>>> encoder_layer = nn.TransformerEncoderLayer(d_model=512, nhead=8)
>>> transformer_encoder = nn.TransformerEncoder(encoder_layer, num_layers=6)
>>> src = torch.rand(10, 32, 512)
>>> out = transformer_encoder(src)
```

🤗 **Transformers** https://github.com/huggingface/transformers

RNNs vs CNNs vs Transformers



(a) RNN (b) CNN (c) Self-attention

Why Self-Attention? A Targeted Evaluation of Neural Machine Translation Architectures

Complexity of transformers

Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(log_k(n))$
Self-Attention (restricted)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

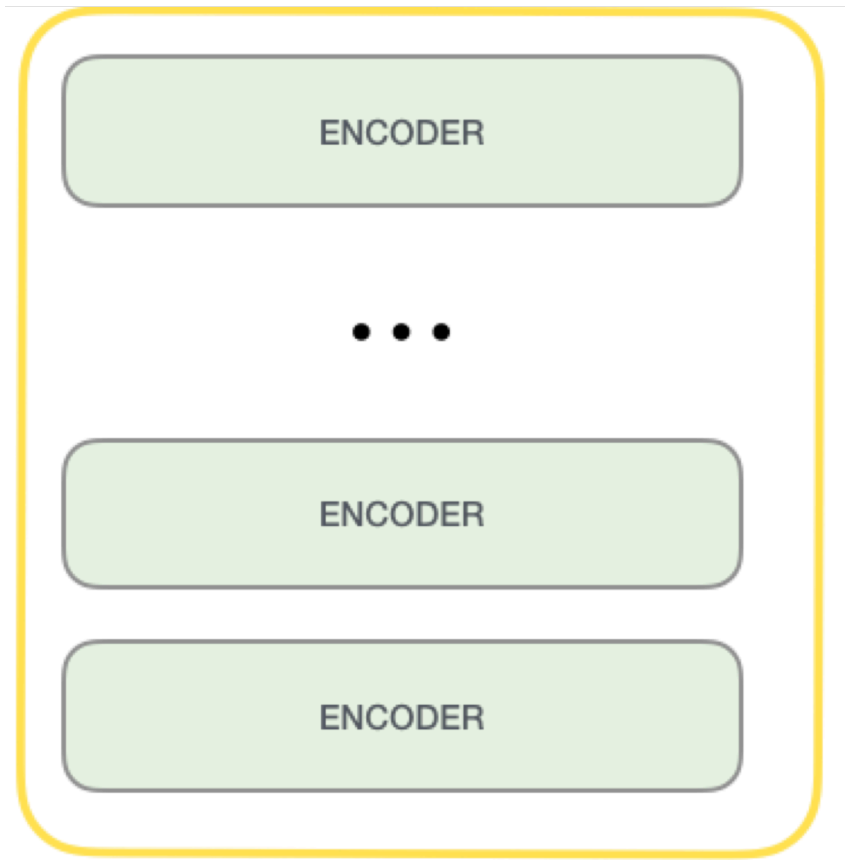
n: sequence length, *d*: representation dimensionality, *k*: kernel width, *r*: neighborhood size

Language modeling with transformers

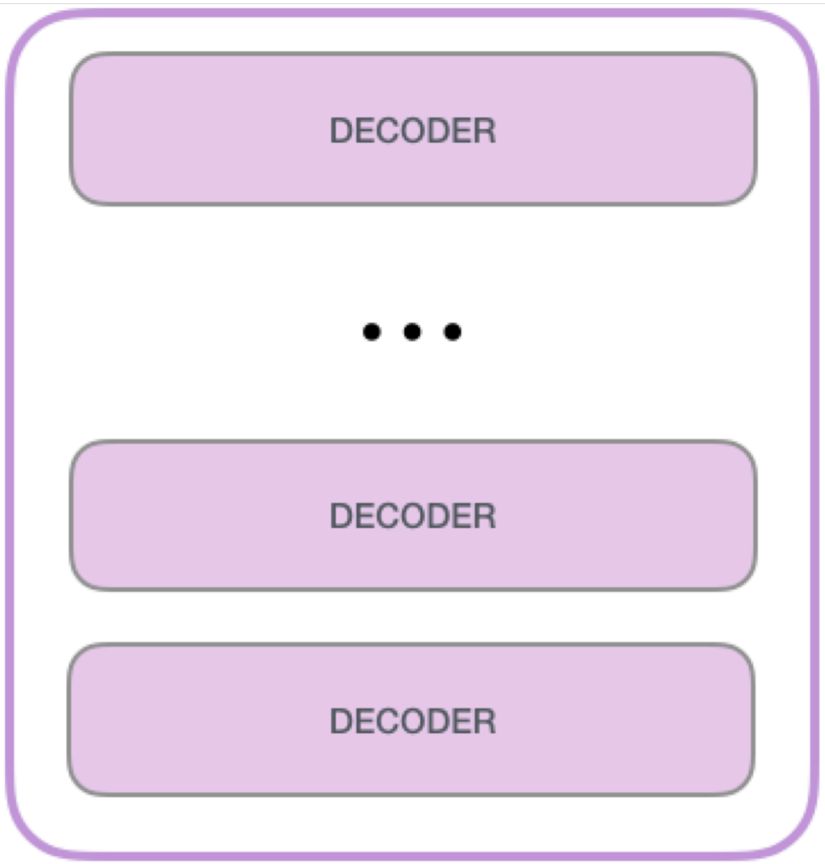
- Self-supervised Transformer based models shattered language understanding benchmarks in NLP in 2018.

  Trained on large text corpus with self-supervised objectives and then transferred.

  - BERT (built on Transformer encoders)

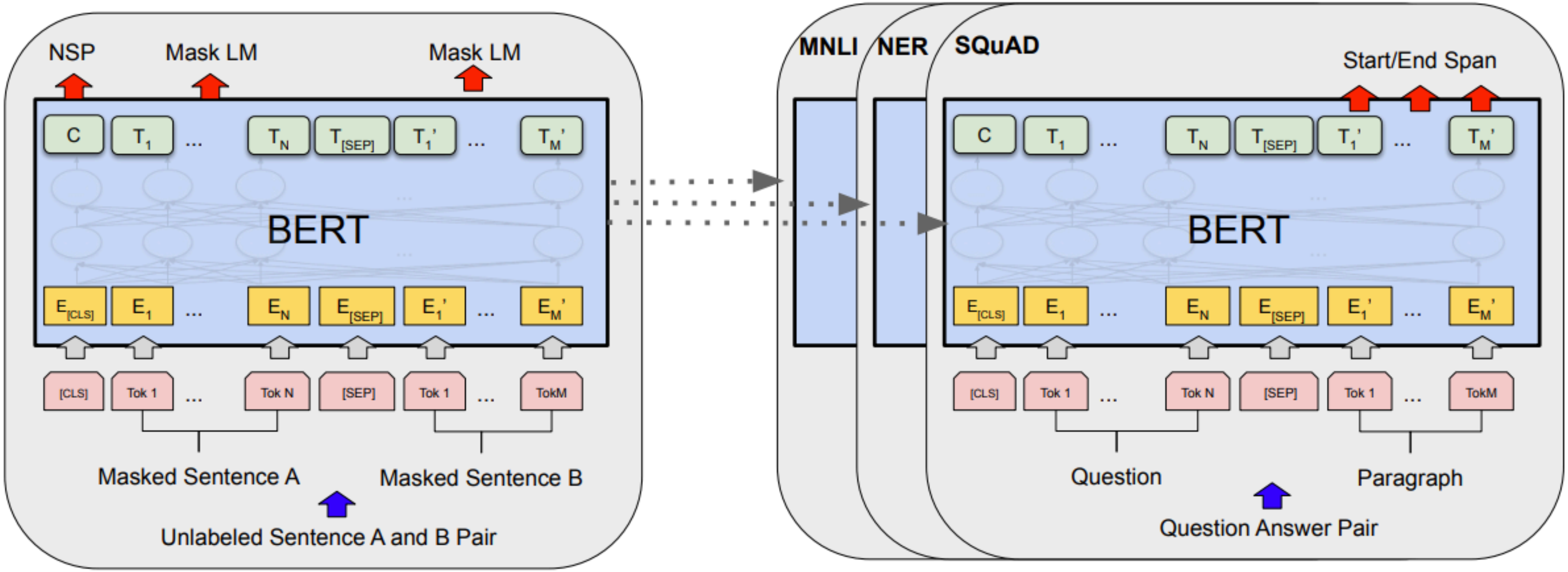


  - GPT-2 (built on Transformer decoders)



Pretraining

Task-specific fine-tuning



Pre-training

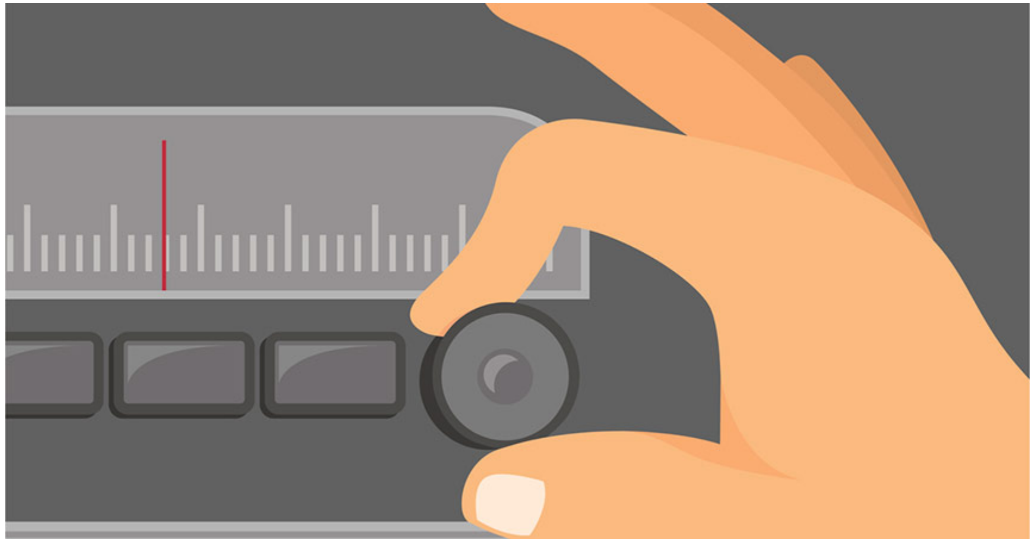
Fine-Tuning

Pretraining

- Big pile of data!
- Lots of resources to train!

Task-specific fine-tuning

- Small amount of annotated data specific to a task
- Start with pre-trained model

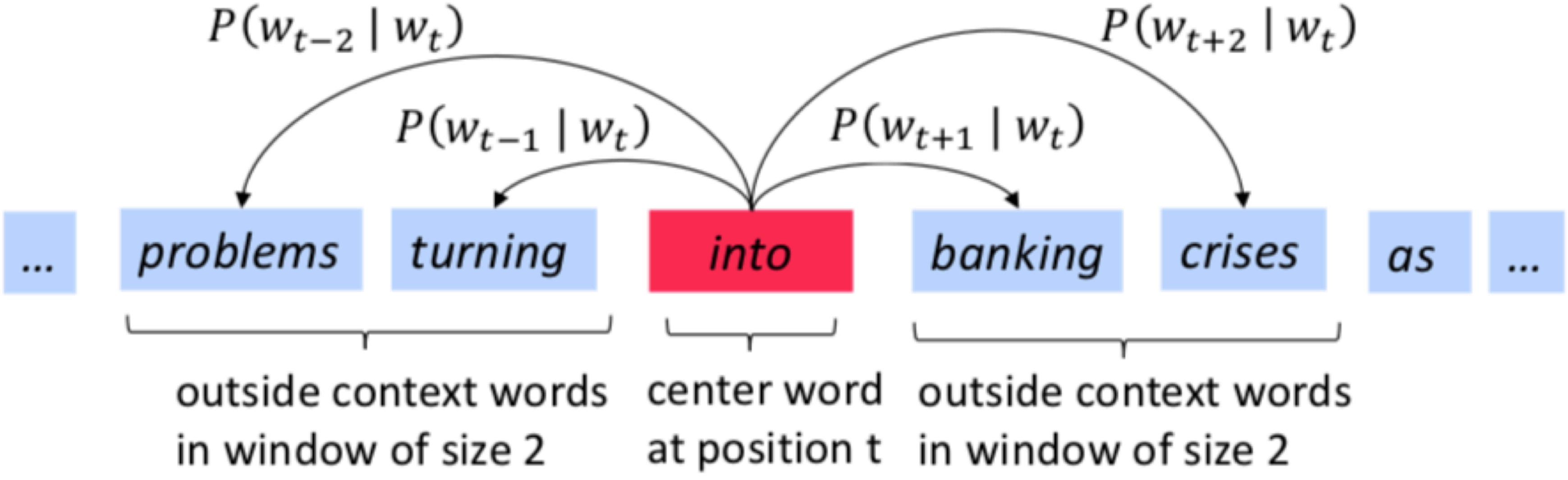


Pretraining

- Pretraining using classification
  - Vision backbones on ImageNet
  - Great but requires labeled images!
- Pretraining with autoencoders
  - Just find lots of data online
- Pretraining by masking – this is what we will look at

Pretraining for language

Recall: How are word embeddings learned?



Word2Vec:

- Skip gram: predict context words given center word
- CBOW: predict center word given context words

Pretraining for language

Language modeling

- Predict probability of a sequence (of tokens)

$$P(w_1 w_2 \ldots w_n) = \prod_i P(w_i | w_1 \ldots w_{i-1})$$

- Traditionally used statistical n-grams

$$P(w_i | w_1 \ldots w_{i-1}) \approx P(w_i | w_{i-k} \ldots w_{i-1})$$

- Now with neural models

$$P(w_i | w_1 \ldots w_{i-1}) \approx f(w_i | \phi(w_1 \ldots w_{i-1}))$$

- Can mask out any word

$$P(w_i | w_1 \ldots w_{i-1} w_{i+1} \ldots w_n)$$

$$\approx f(w_i | \phi(w_1 \ldots w_{i-1} w_{i+1} \ldots w_n))$$

Like Word2Vec's CBOW

Masked Language Modeling

Example: `my dog is hairy`, we replace the word hairy

- 80% of time: replace word with `[MASK]` token

  `my dog is [MASK]`

- 10% of time: replace word with random word

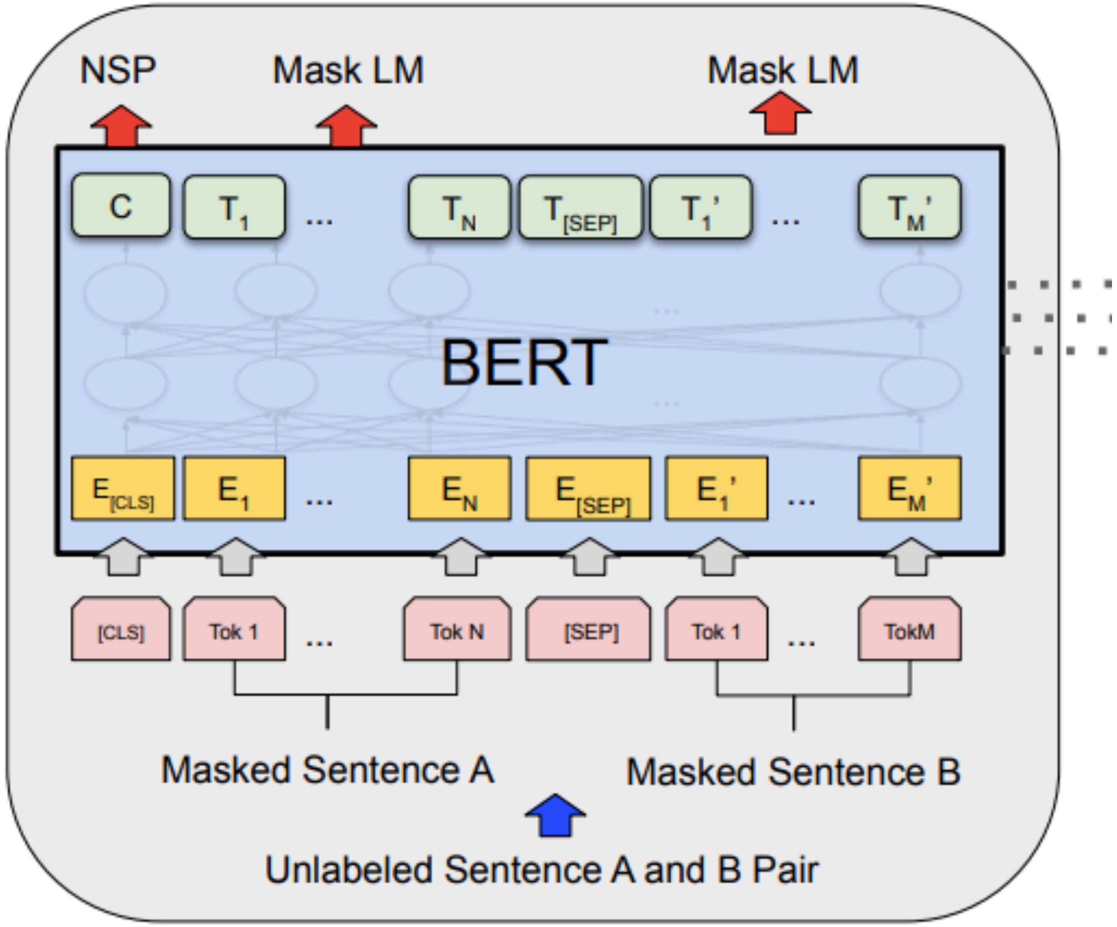
  `my dog is apple`

- 10% of time: keep word unchanged to bias representation toward actual observed word

  `my dog is hairy`

Modelling Sequences -- Transformers

- Two training objectives:
  - Masked Language Modelling
  - Next Sentence Prediction



Pre-training

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding https://arxiv.org/pdf/1810.04805.pdf

BERT performance

Two model sizes

- $\text{BERT}_{\text{BASE}}$ (L=12, H=768, A=12, Total Parameters=110M)
  $\text{BERT}_{\text{LARGE}}$ (L=24, H=1024, A=16, Total Parameters=340M)
- Does well for several tasks!

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**

All of these models are Transformer models

ELMo
Oct 2017
Training:
800M words
42 GPU days

GPT
June 2018
Training
800M words
240 GPU days

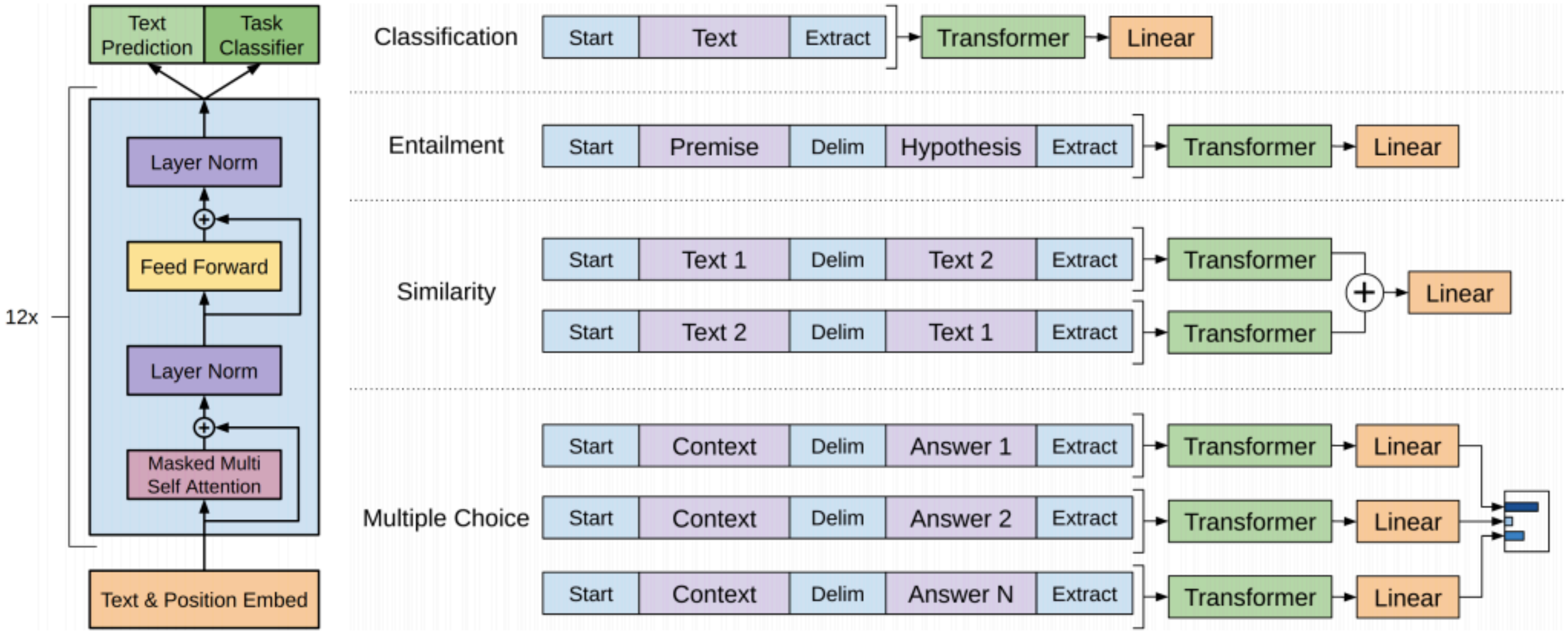
BERT
Oct 2018
Training
3.3B words
256 TPU days
~320–560
GPU days

GPT-2
Feb 2019
Training
40B words
~2048 TPU v3
days according to
a reddit thread

XL-Net,
ERNIE,
Grover
RoBERTa, T5
July 2019—

(slide credit: Stanford CS224N, Chris Manning)

GTP (Generative pretrained transformer)

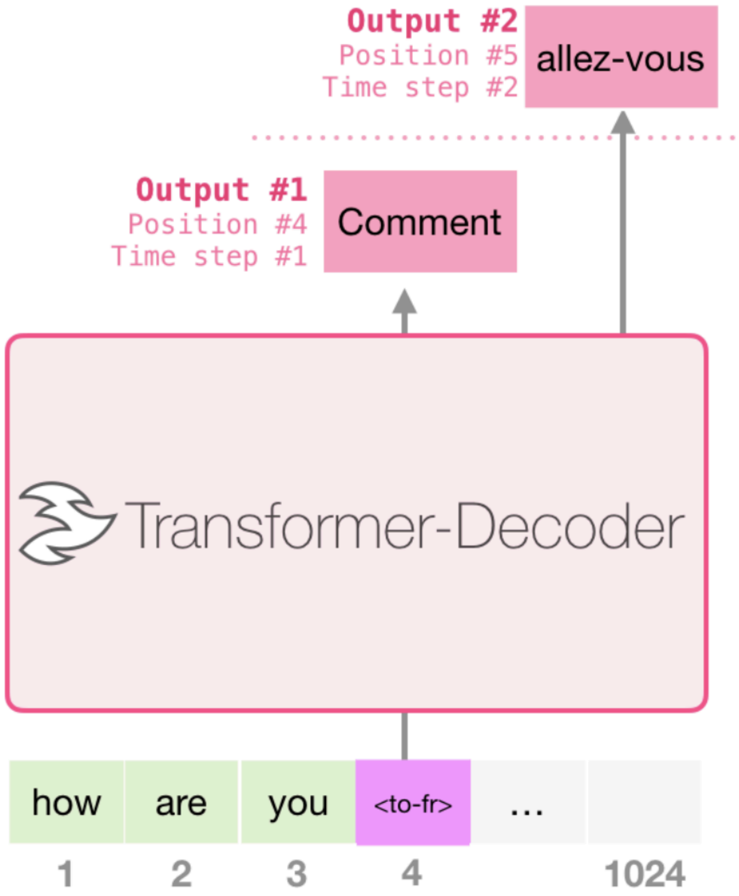


Improving language understanding by generative pre-training (Radford et al, 2018)

GTP

- Machine Translation

I	am	a	student	<to-fr>	je	suis	étudiant
let	them	eat	cake	<to-fr>	Qu'ils	mangent	de
good	morning	<to-fr>	Bonjour				



http://jalammar.github.io/illustrated-gpt2/

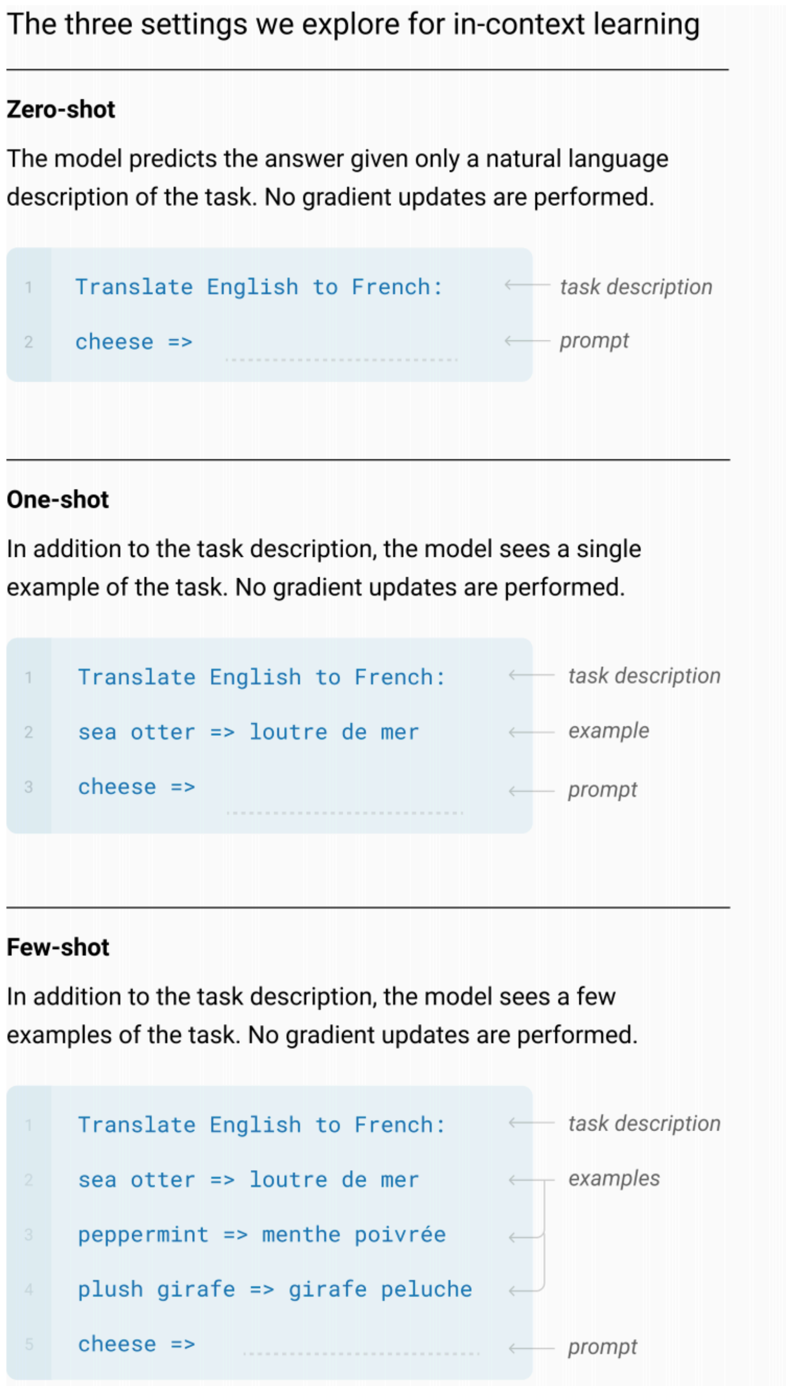
GTP models

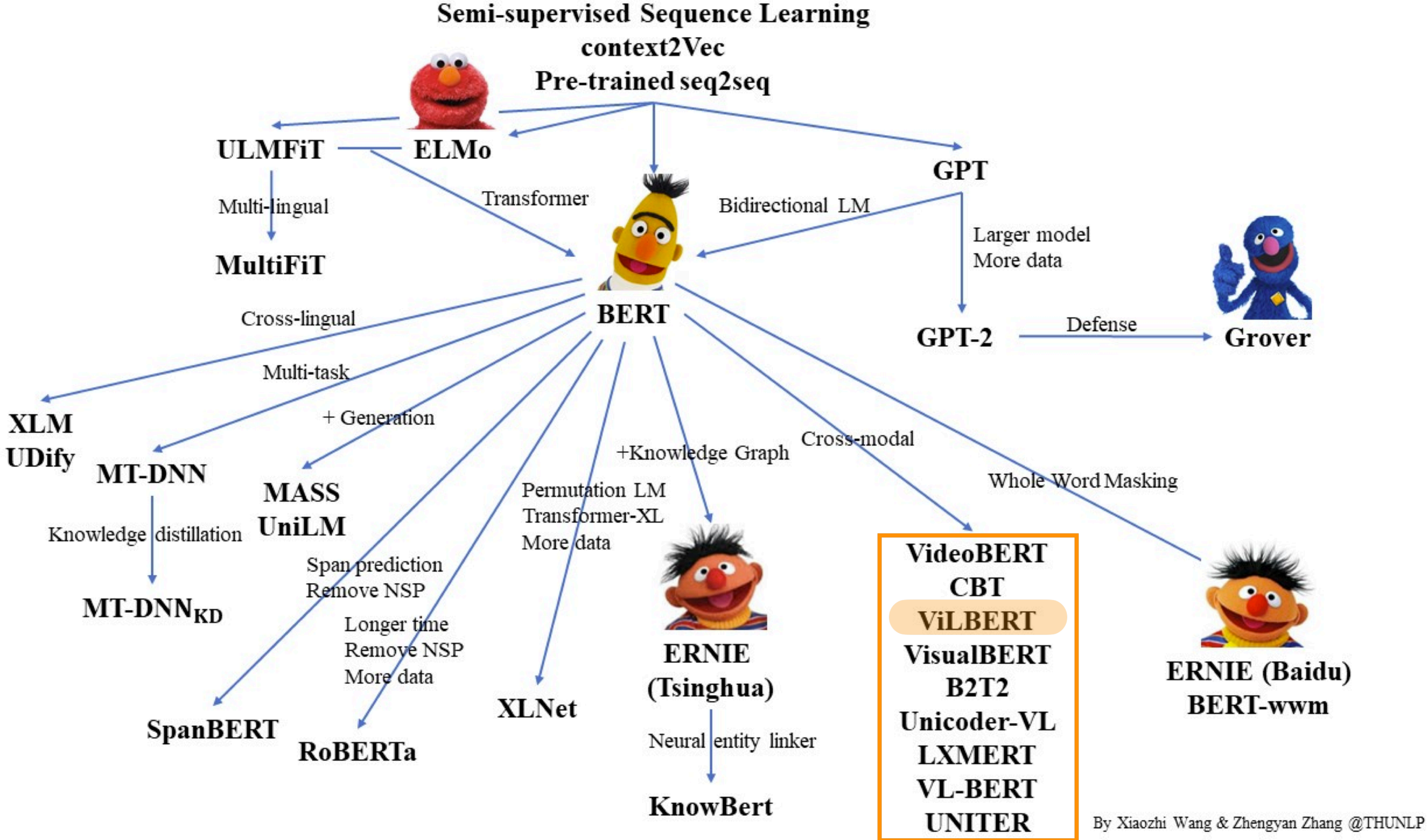
OpenAI

- GTP
  - Improving language understanding by generative pre-training (Radford et al, 2018)
  - Large language model with transformers with fine-tuning!
  - Trained on BooksCorpus (800M words), 117M parameters (12 layers)
- GTP-2
  - Language Models are Unsupervised Multitask Learner (Radford et al, 2019)
  - Trained on WebText (40B words), 1.5B parameters (48 layers)
  - No fine-tuning, few-shot learning
- GTP-3
  - Language Models are Few-Shot Learners (Brown et al, 2020)
  - Trained on Web+Books+Wikipedia (300B words), 175B parameters (96 layers)

Few-shot learning

- Few-shot
  - A few examples are provided at test time
- One-shot (1 training example)
- Zero-shot (0 training examples)




Semi-supervised Sequence Learning
context2Vec
Pre-trained seq2seq
ULMFiT
ELMo
GPT
Multi-lingual
MultiFiT
Transformer
Bidirectional LM
Larger model
More data
BERT
GPT-2
Defense
Grover
Cross-lingual
Multi-task
XLM
UDify
+ Generation
MT-DNN
MASS
UniLM
Knowledge distillation
MT-DNN$_{KD}$
Span prediction
Remove NSP
Longer time
Remove NSP
More data
Permutation LM
Transformer-XL
More data
XLNet
SpanBERT
RoBERTa
+Knowledge Graph
ERNIE
(Tsinghua)
Neural entity linker
KnowBert
Cross-modal
VideoBERT
CBT
ViLBERT
VisualBERT
B2T2
Unicoder-VL
LXMERT
VL-BERT
UNITER
Whole Word Masking
ERNIE (Baidu)
BERT-wwm
By Xiaozhi Wang & Zhengyan Zhang @THUNLP

Pretraining with transformers for vision and language

Pretraining for images

- Image generation as autoregressive sequence modeling
  - Use Transformers! $\log p(x) = \sum_{t=1}^{h \cdot w \cdot 3} \log p(x_t \mid x_{<t})$

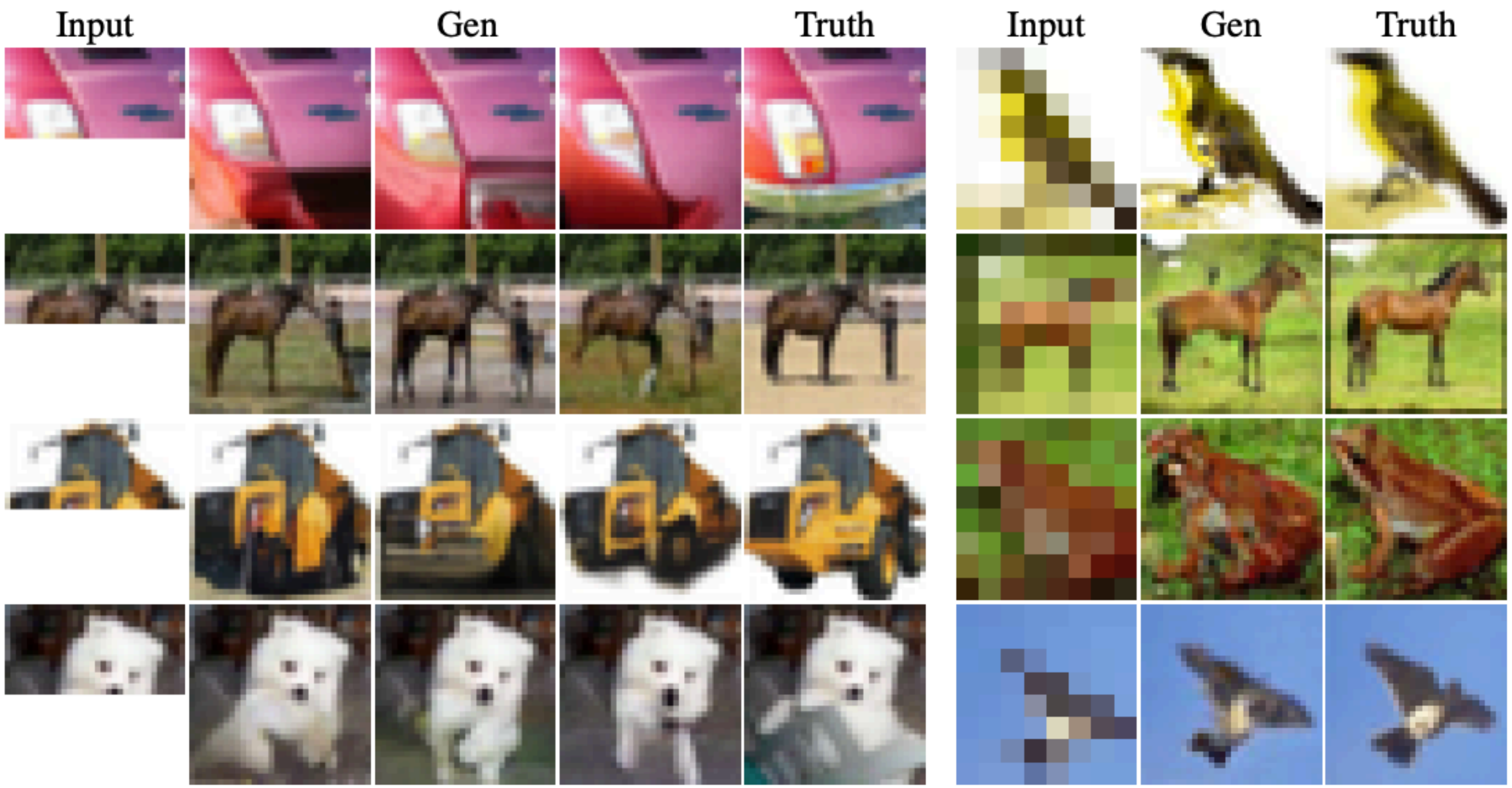


Image Transformer, Parmar et al, ICML 2018

Pretraining for images

- Image generation as autoregressive sequence modeling
  - Use Transformers!
- RGB values modeled as categorical or ordinal values
  - Each channel is embedded
  - Position is embedded
- Local attention

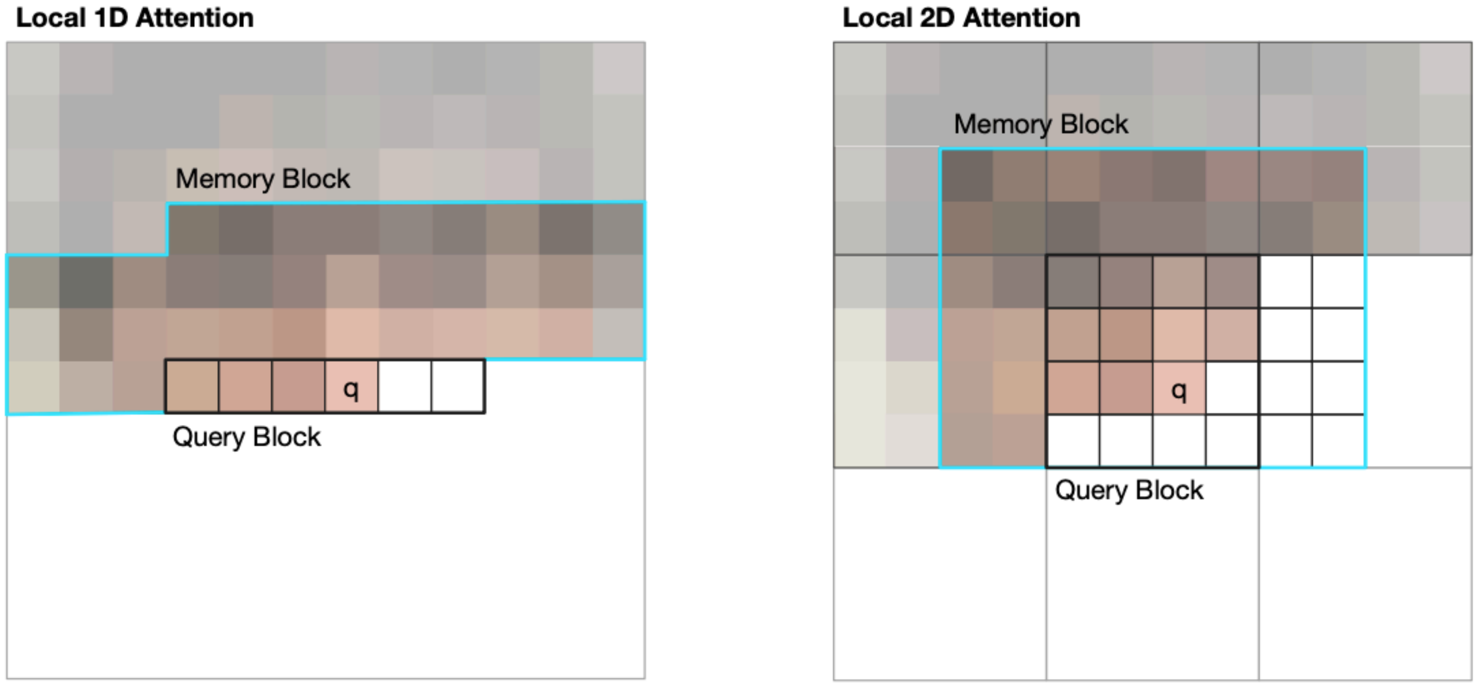


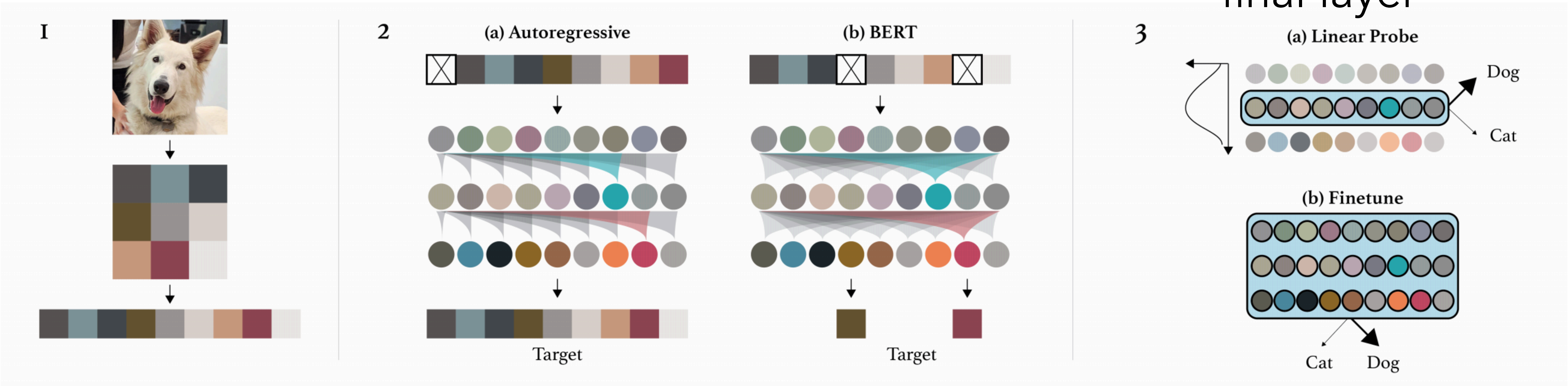
Image Transformer, Parmar et al, ICML 2018

Image GPT

Classify using features from an intermediate or final layer

Autoregressive

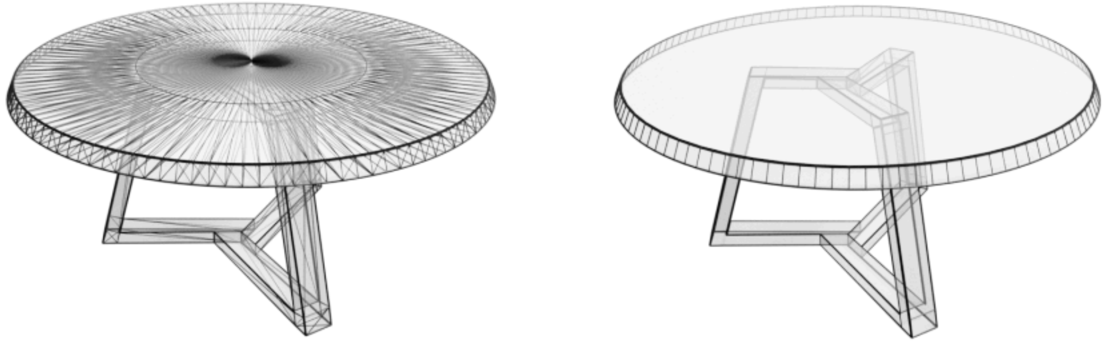
Masked



Generative Pretraining from Pixels, Chen et al, ICML 2020
https://openai.com/blog/image-gpt/

Pretraining for 3D shapes

- Mesh generation as autoregressive sequence modeling
  - Use Transformers!
- Model 3D shapes as n-gons (polygons)
- Decompose mesh-generation into generating vertices and then faces



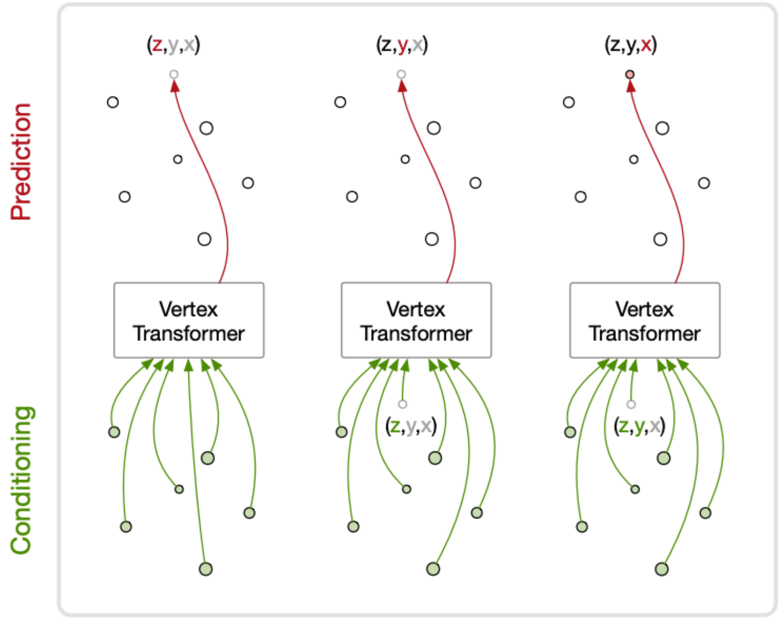
(a) Triangle mesh (b) n-gon mesh

$$\begin{aligned} p(\mathcal{M}) &= p(\mathcal{V}, \mathcal{F}) \\ &= p(\mathcal{F}|\mathcal{V})p(\mathcal{V}) \end{aligned}$$

PolyGen: An Autoregressive Generative Model of 3D Meshes, Nash et al, ICML 2020

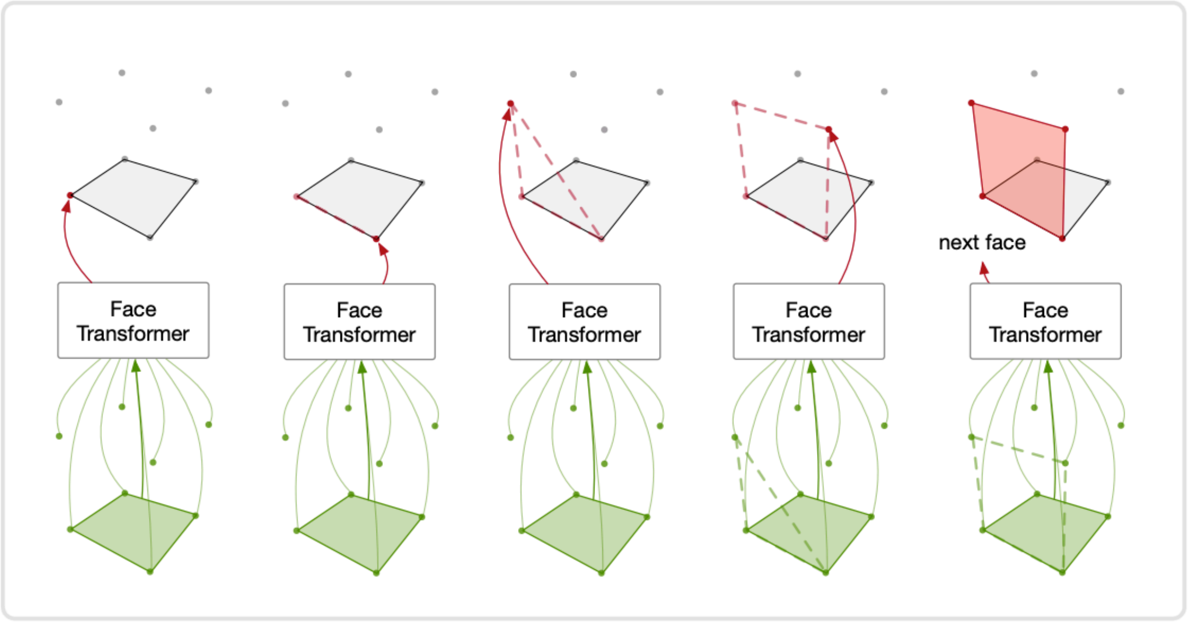
Pretraining for shapes

Vertex Model



$$p(\mathcal{V}^{\text{seq}};\theta) = \prod_{n=1}^{N_V} p(v_n|v_{<n};\theta)$$

Face Model



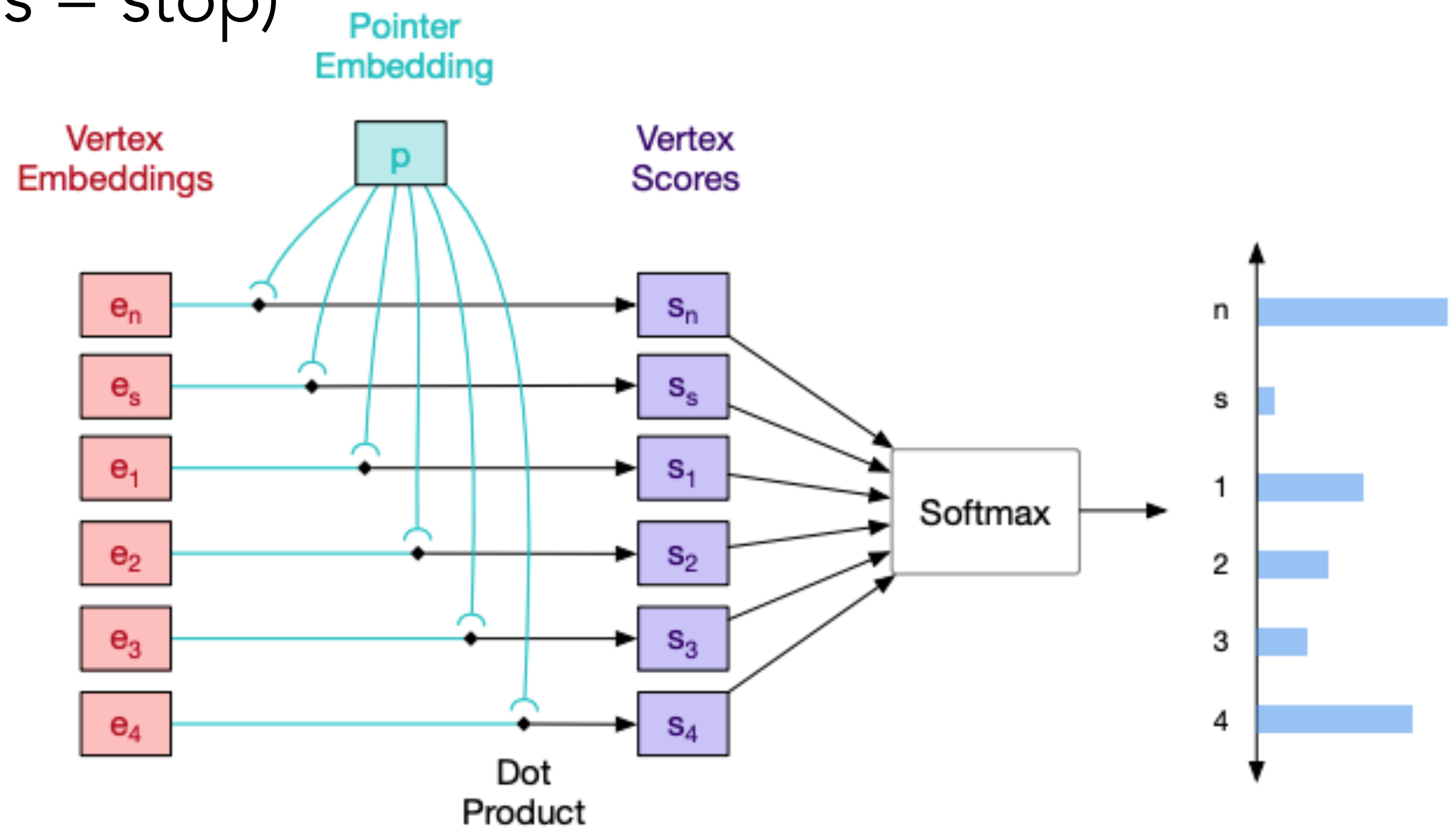
$$p(\mathcal{F}^{\text{seq}}|\mathcal{V};\theta) = \prod_{n=1}^{N_F} p(f_n|f_{<n},\mathcal{V};\theta)$$

PolyGen: An Autoregressive Generative Model of 3D Meshes, Nash et al, ICML 2020

Pretraining for shapes

- Mesh pointer network for predicting vertex for a face

(n = end of face, s = stop)

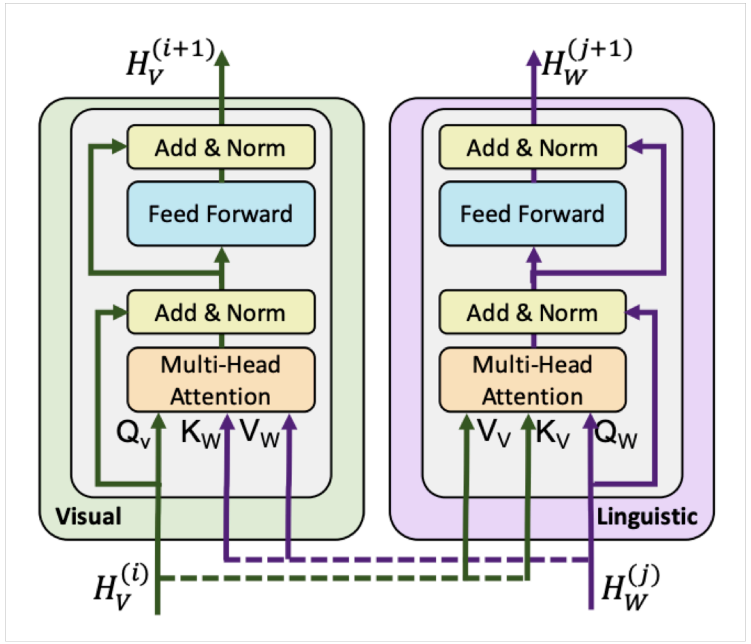


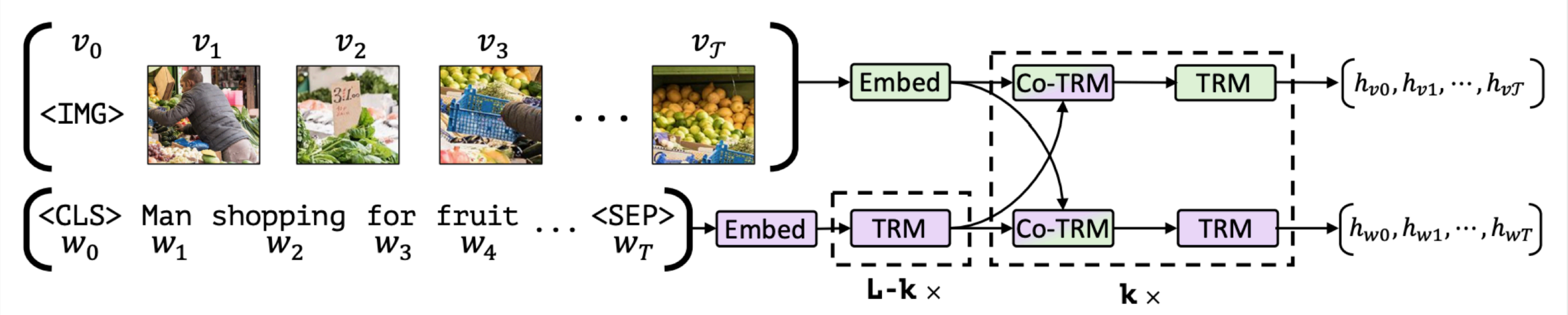
PolyGen: An Autoregressive Generative Model of 3D Meshes, Nash et al, ICML 2020

Pretrained representations for vision and language

Image represented as

- series of **image region features** (extracted from pre-trained object detection network)
- **Region position** encoded as $5d$ vector





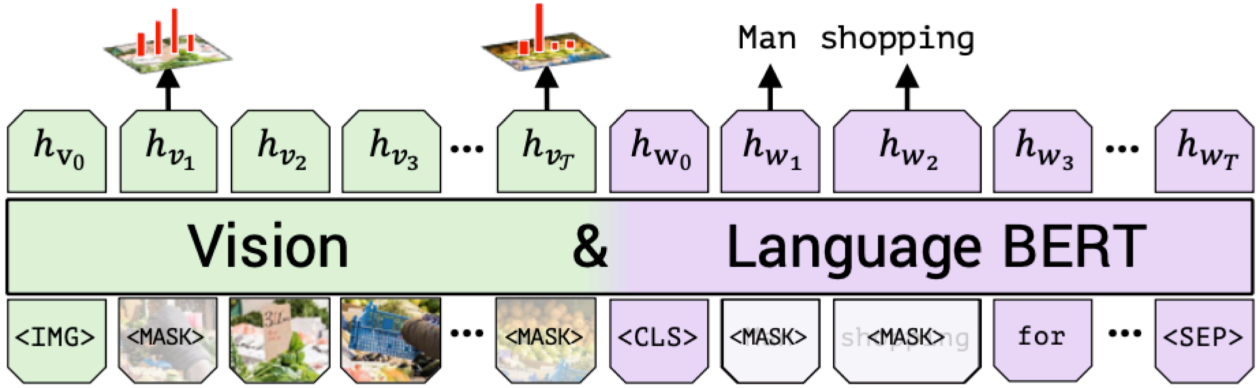
ViLBERT: Pretraining Task-Agnostic Visiolinguistic Representations for Vision-and-Language Tasks
[Lu et al 2019, https://arxiv.org/pdf/1908.02265.pdf]

Pretrained representations for vision and language

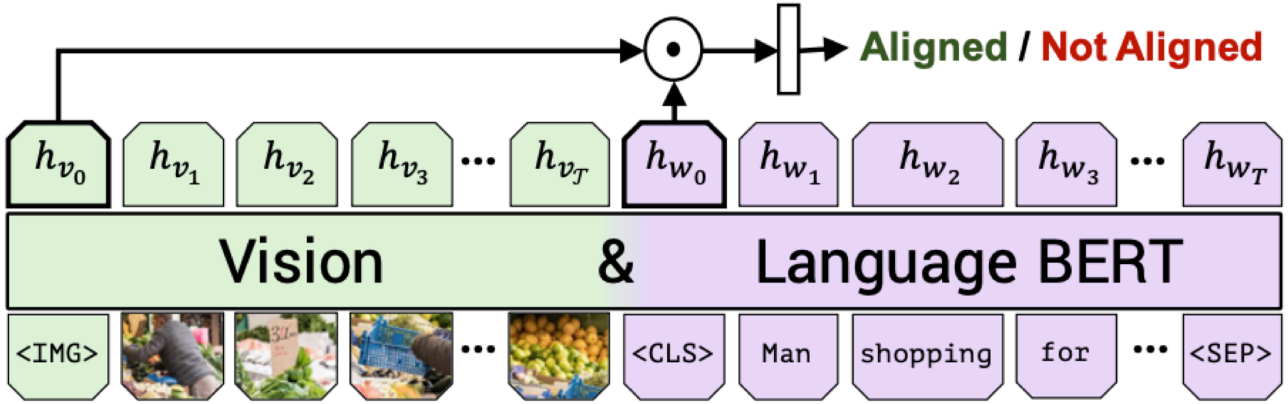
Predict semantic class distribution

Trained on

- Conceptual captions (~3.3M images with captions cleaned from alt-text labels)
- Two tasks to predict:
  - masked out words and semantic class distribution for masked out image regions
  - Is the image/description aligned?



(a) Masked multi-modal learning



(b) Multi-modal alignment prediction

ViLBERT: Pretraining Task-Agnostic Visiolinguistic Representations for Vision-and-Language Tasks [Lu et al 2019, https://arxiv.org/pdf/1908.02265.pdf]

Pretrained representations for vision and language

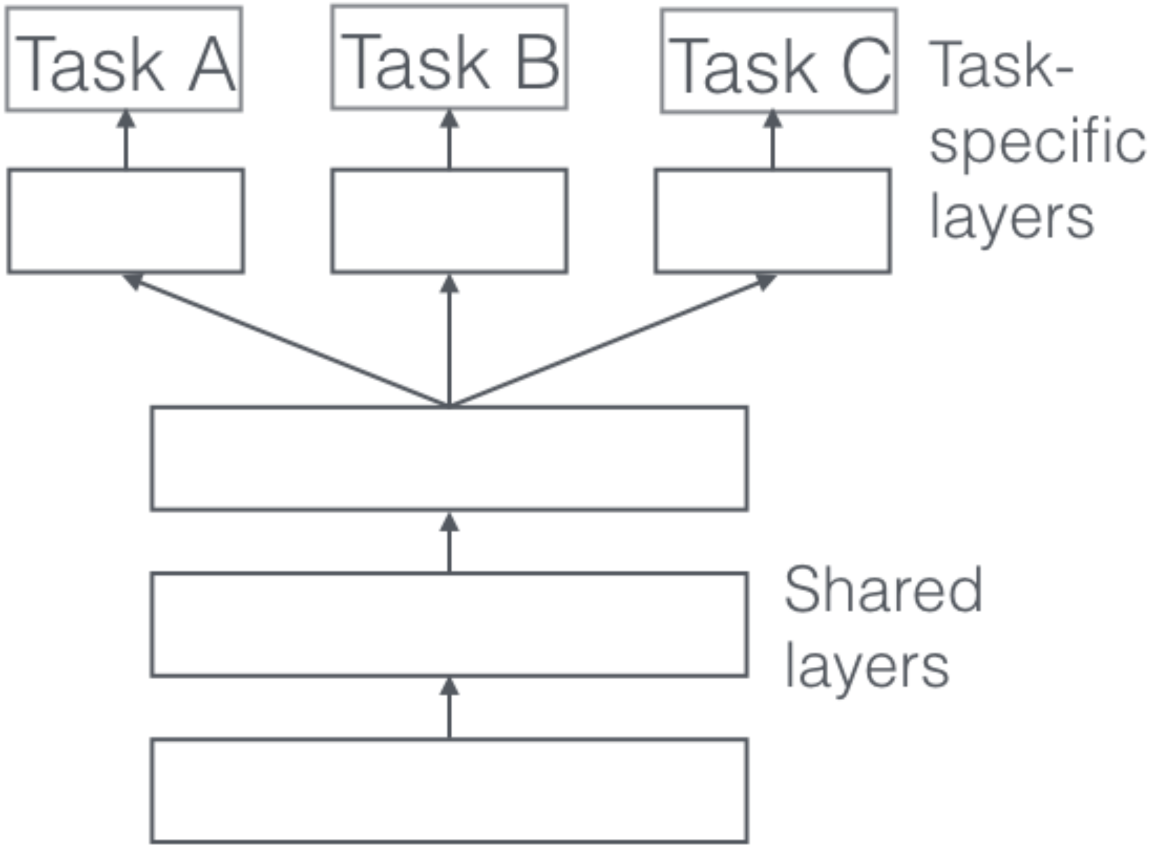
	Method	VQA [3]	VCR [25]			RefCOCO+ [32]			Image Retrieval [26]			ZS Image Retrieval		
		test-dev (test-std)	Q→A	QA→R	Q→AR	val	testA	testB	R1	R5	R10	R1	R5	R10
SOTA	DFAF [36]	70.22 (70.34)	-	-	-	-	-	-	-	-	-	-	-	-
	R2C [25]	-	63.8 (65.1)	67.2 (67.3)	43.1 (44.0)	-	-	-	-	-	-	-	-	-
	MAttNet [33]	-	-	-	-	65.33	71.62	56.02	-	-	-	-	-	-
	SCAN [35]	-	-	-	-	-	-	-	48.60	77.70	85.20	-	-	-
Ours	Single-Stream†	65.90	68.15	68.89	47.27	65.64	72.02	56.04	-	-	-	-	-	-
	Single-Stream	68.85	71.09	73.93	52.73	69.21	75.32	61.02	-	-	-	-	-	-
	ViLBERT†	68.93	69.26	71.01	49.48	68.61	75.97	58.44	45.50	76.78	85.02	0.00	0.00	0.00
	ViLBERT	**70.55** (70.92)	**72.42** (73.3)	**74.47** (74.6)	**54.04** (54.8)	**72.34**	**78.52**	**62.61**	**58.20**	**84.90**	**91.52**	**31.86**	**61.12**	**72.80**

Pretraining improves performance on variety of vision+language tasks!

ViLBERT: Pretraining Task-Agnostic Visiolinguistic Representations for Vision-and-Language Tasks
[Lu et al 2019, https://arxiv.org/pdf/1908.02265.pdf]

Multi-task learning

- One model, several tasks
- Task conditioning
  - Predict output given input + task
- Common parameters and task-specific parameters
- Two extremes:
  - Single model with shared parameters
  - Independent models with gating



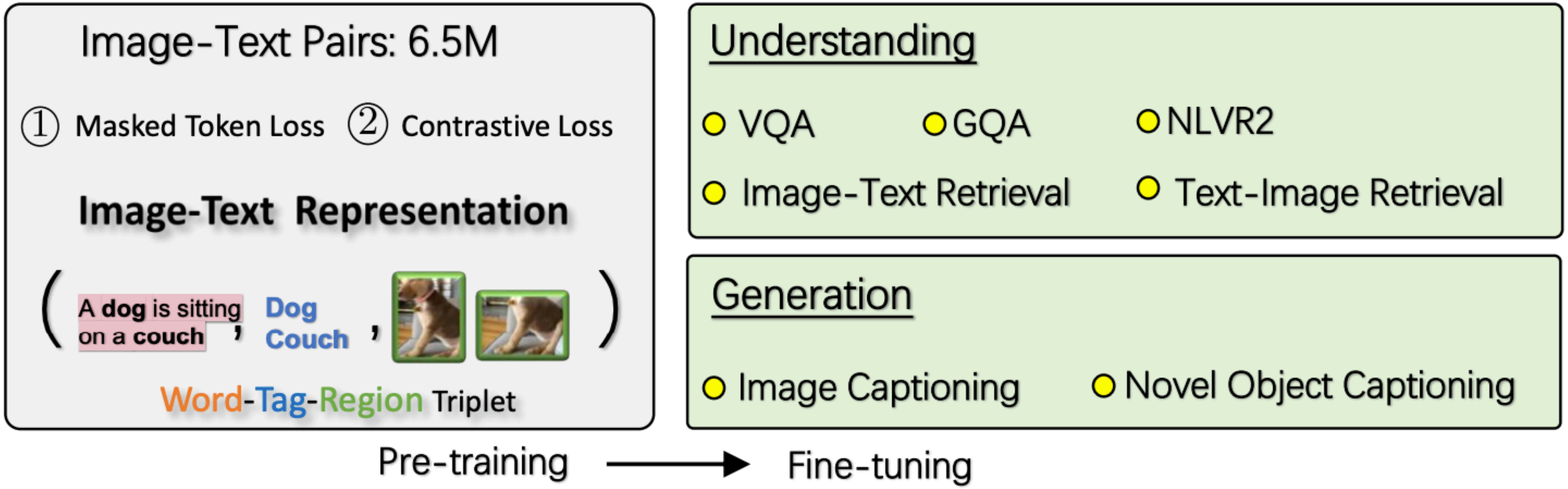
Multi-task learning with vision+language

Tasks		SOTA	ViLBERT	VLBERT	Unicoder-VL	VisualBERT	LXMERT	UNITER BASE	UNITER LARGE	Ours$_{ST}$	Ours$_{AT->ST}$
	Pretraining Data		CC	CC + Wiki Corpus	CC	CC + COCO	COCO + VG	CC+SUB+COCO+VG		CC	CC
VQA	test-dev	70.63	70.55	70.50	-	70.80	72.42	72.27	**73.24**	71.82	73.15
VG QA	val	-	-	-	-	-	-	-	-	34.38	**36.64**
GQA	test-dev	-	-	-	-	-	60.00	-	-	58.19	**60.65**
IR COCO	R1	61.60	-	-	**68.50**	-	-	-	-	65.28	68.00
	R5	89.6	-	-	**92.70**	-	-	-	-	91.02	92.38
	R10	95.2	-	-	**96.90**	-	-	-	-	96.18	96.52
IR Flickr	R1	48.60	58.20	-	68.30	-	-	71.50	**73.66**	61.14	67.90
	R5	77.70	84.90	-	90.30	-	-	91.16	**93.06**	87.16	89.60
	R10	85.20	91.52	-	94.60	-	-	95.20	**95.98**	92.30	94.18
Visual 7W	test	72.53	-	-	-	-	-	-	-	80.51	**83.35**
Ref-COCO	test	77.12	-	-	-	-	-	80.48	80.88	78.63	**81.20**
Ref-COCO+	test	67.17	70.93	69.47	-	-	-	73.26	73.73	71.11	**74.22**
Ref-COCOg	test	69.46	-	-	-	-	-	74.51	75.77	72.24	**76.35**
GuessWhat	test	61.30	-	-	-	-	-	-	-	62.81	**65.69**
NLVR^2	test-P	53.50	-	-	-	67.00	74.50	77.87	**79.50**	74.25	78.87
SNLI-VE	test	71.16	-	-	-	-	-	78.02	**78.98**	76.72	76.95

12-in-1: Multi-Task Vision and Language Representation Learning, Lu et al, CVPR 2020

Oscar: Pre-training with object-semantic alignment

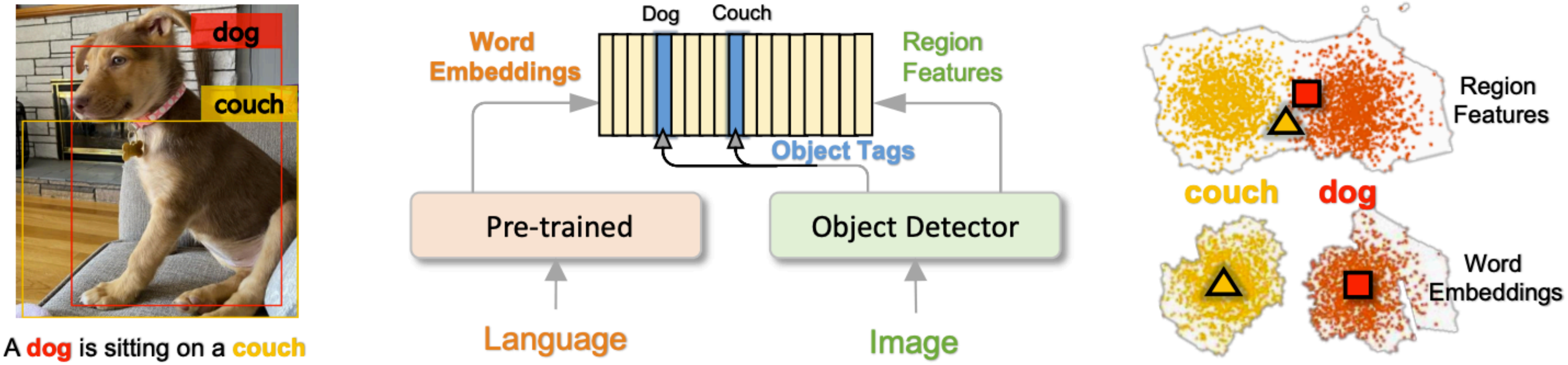
- Pretrained on 6.5 million pairs of vision+language data (MSCOCO, Conceptual Captions (CC), SBU captions, flicker30k, GQA)
- Fine tuned on 7 tasks



Oscar: Object-Semantics Aligned Pre-training for Vision-Language Tasks, Li et al, ECCV 2020

Oscar: Pre-training with object-semantic alignment

- Use detected object tags as anchors



(a) Image-text pair (b) Objects as anchor points (c) Semantics spaces

Oscar: Object-Semantics Aligned Pre-training for Vision-Language Tasks, Li et al, ECCV 2020

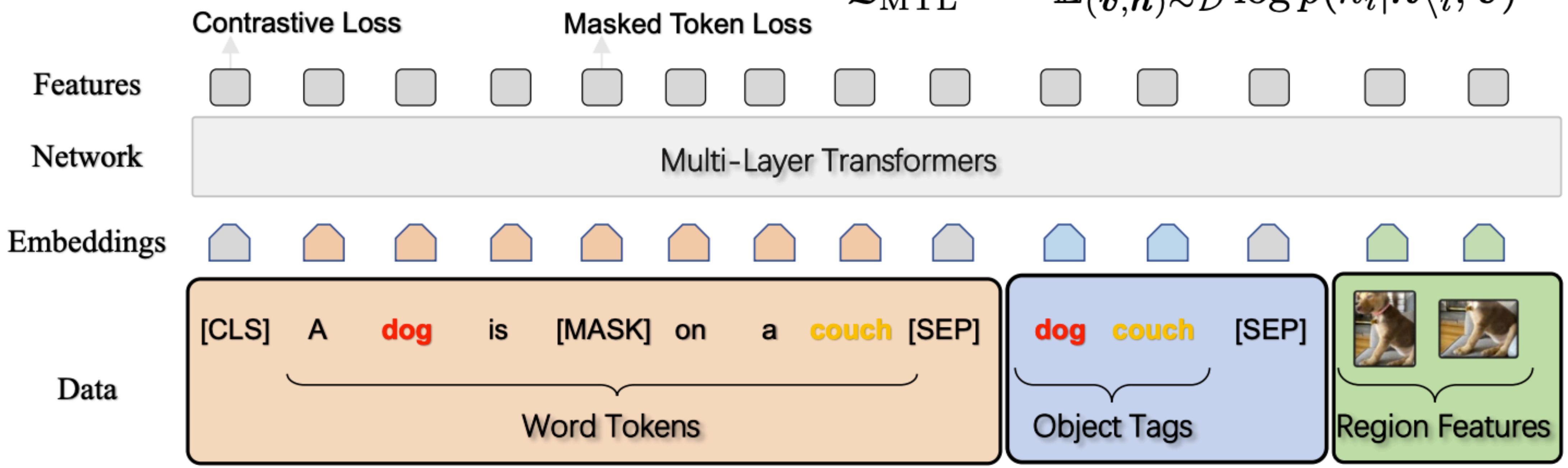
Oscar: Pre-training with object-semantic alignment

Randomly pollute ***q*** $\boldsymbol{h}' \triangleq [\boldsymbol{q}, \boldsymbol{v}]$

Randomly mask token h_i $\boldsymbol{h} \triangleq [\boldsymbol{w}, \boldsymbol{q}]$

$$\mathcal{L}_{\mathrm{C}} = -\mathbb{E}_{(\boldsymbol{h}', \boldsymbol{w}) \sim \mathcal{D}} \log p(y | f(\boldsymbol{h}', \boldsymbol{w}))$$

$$\mathcal{L}_{\mathrm{MTL}} = -\mathbb{E}_{(\boldsymbol{v}, \boldsymbol{h}) \sim \mathcal{D}} \log p(h_i | \boldsymbol{h}_{\setminus i}, \boldsymbol{v})$$



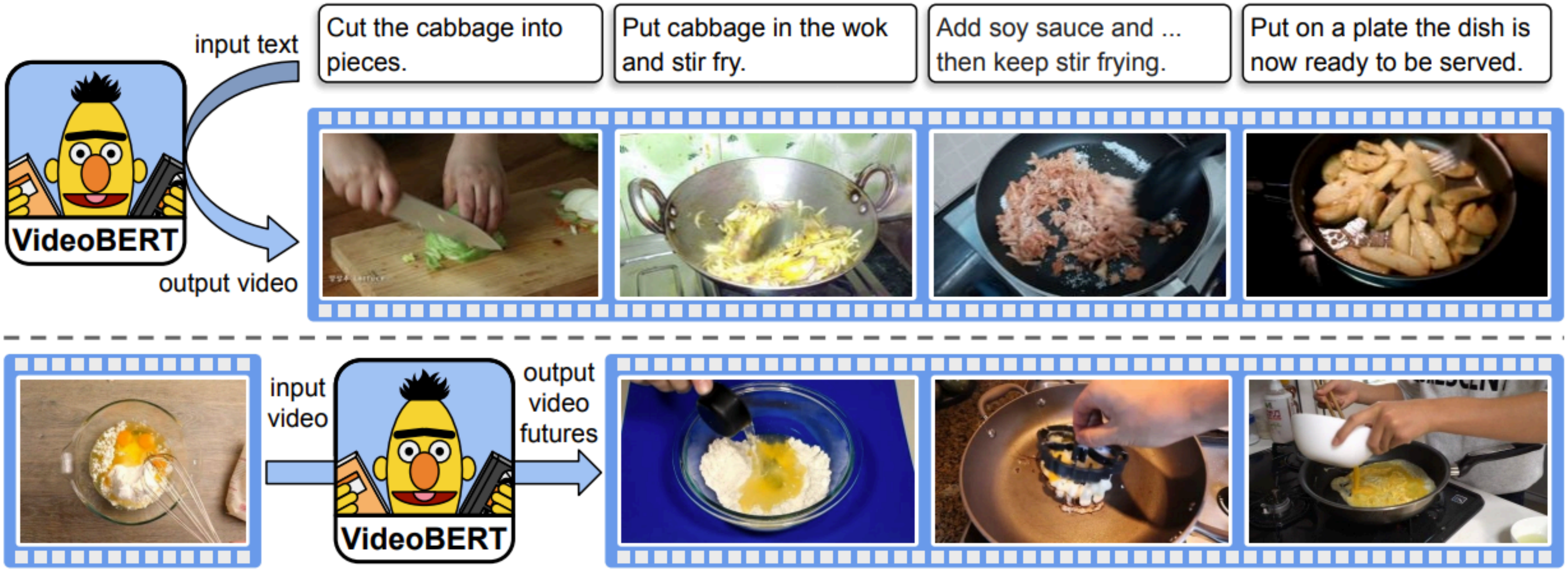
w = text embedding

q = object tags embedding

v = visual features

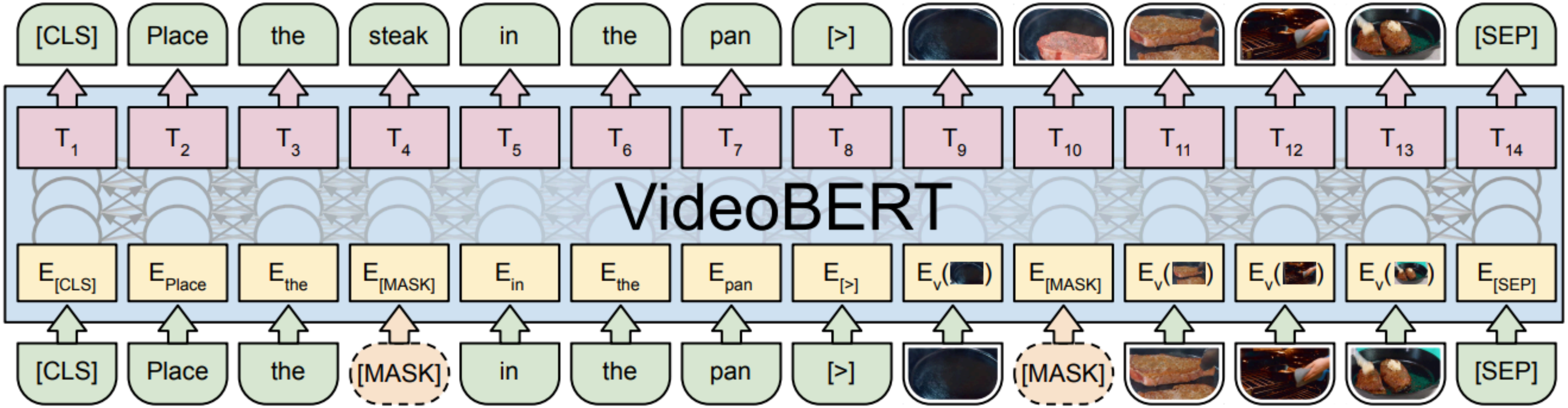
Oscar: Object-Semantics Aligned Pre-training for Vision-Language Tasks, Li et al, ECCV 2020

Pretraining for videos



VideoBERT: A Joint Model for Video and Language Representation Learning, Sun et al, ICCV 2019

Pretraining for videos



VideoBERT: A Joint Model for Video and Language Representation Learning, Sun et al, ICCV 2019

Next week

- Monday: Paper presentations and discussions
    - ViLBERT (Qirui)
    - CLIP (open discussion)
- Thursday: Compositionality and structure