CMPT 983

Grounded Natural Language Understanding

February 2, 2022

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]

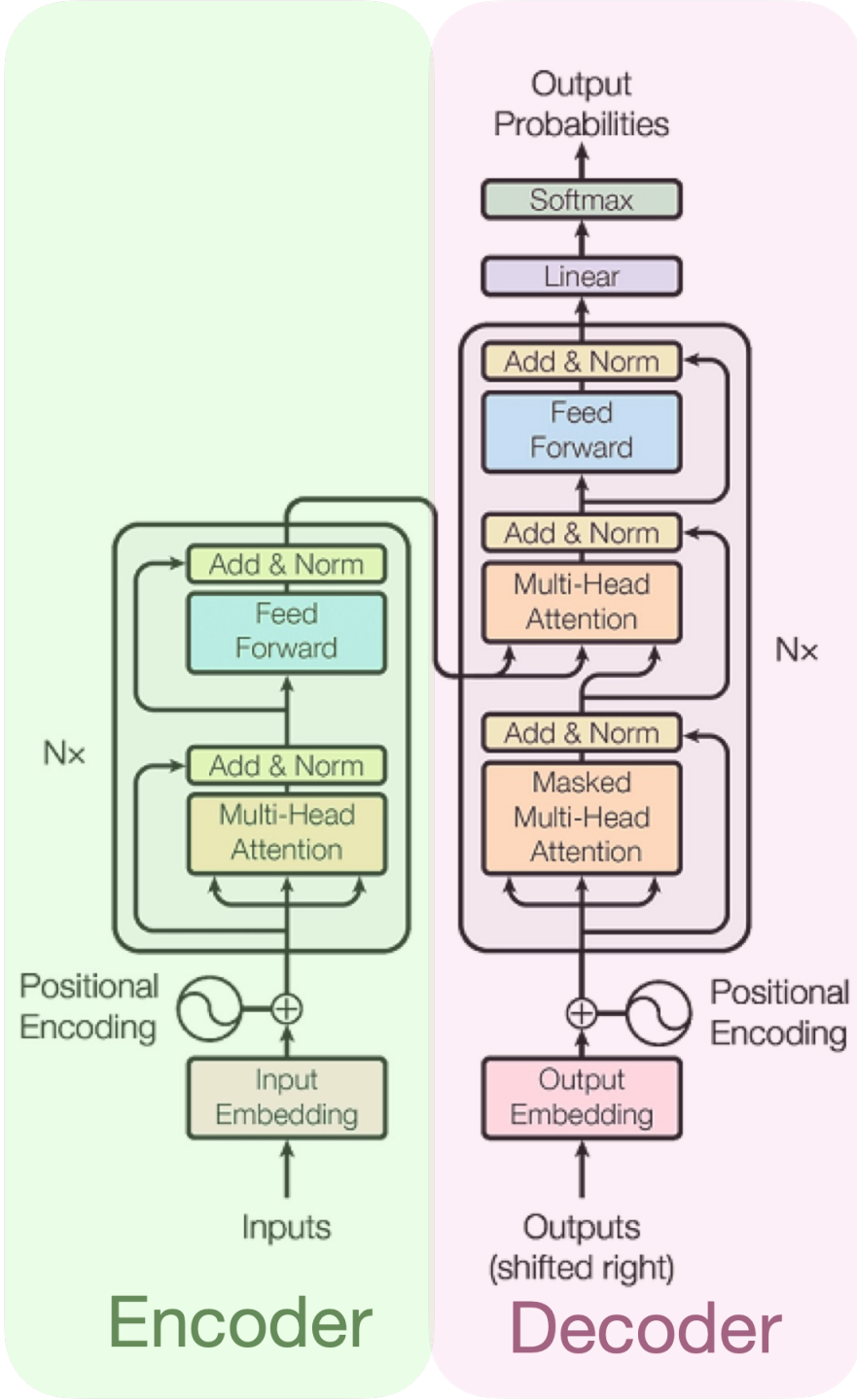
Transformers have had a lot of success in training neural language models. In the past few weeks, we've seen several trending papers with code applying Transformers to new types of task:

🏞️ **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)

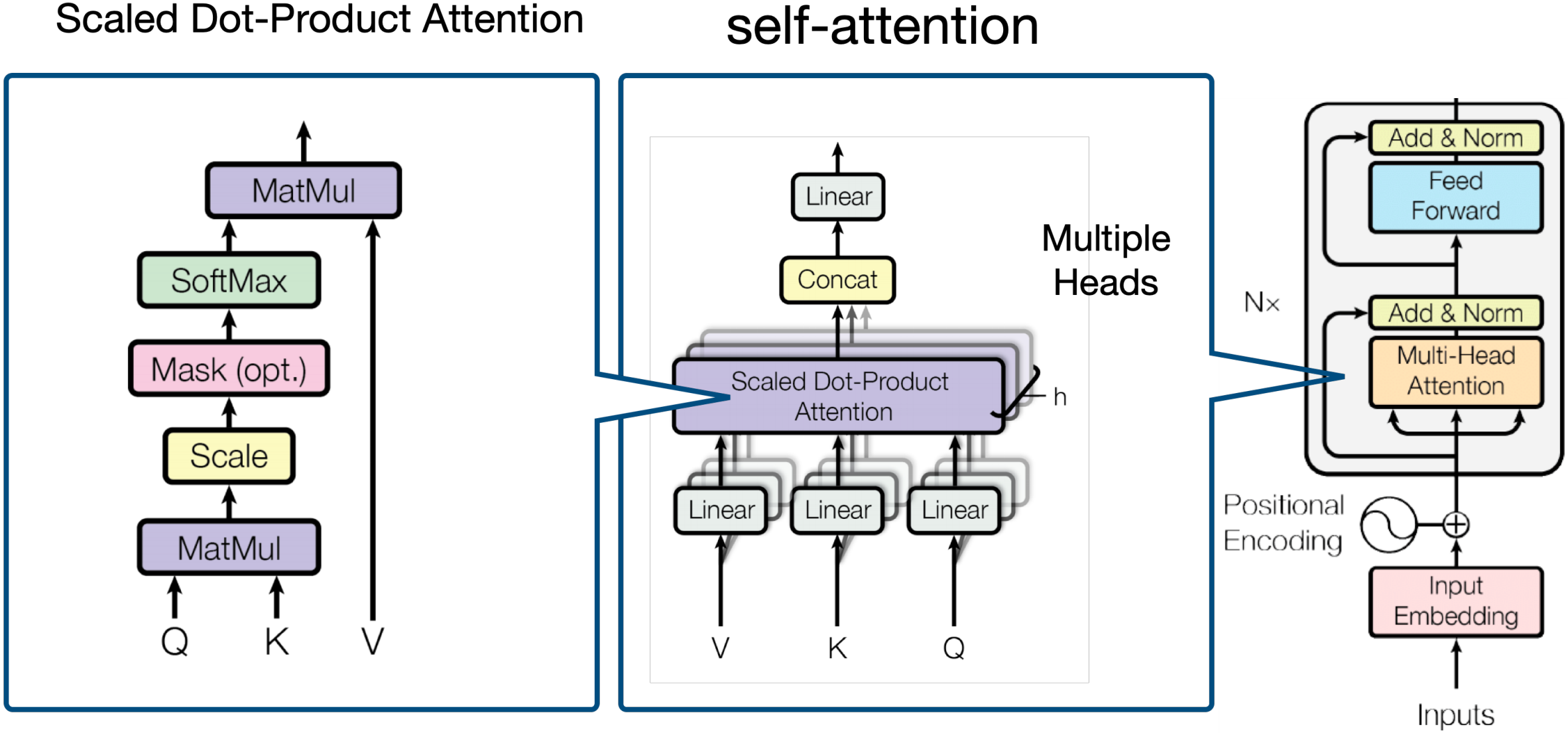
PapersWithCode newsletter (1/20/2021)
https://paperswithcode.com/newsletter/3

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

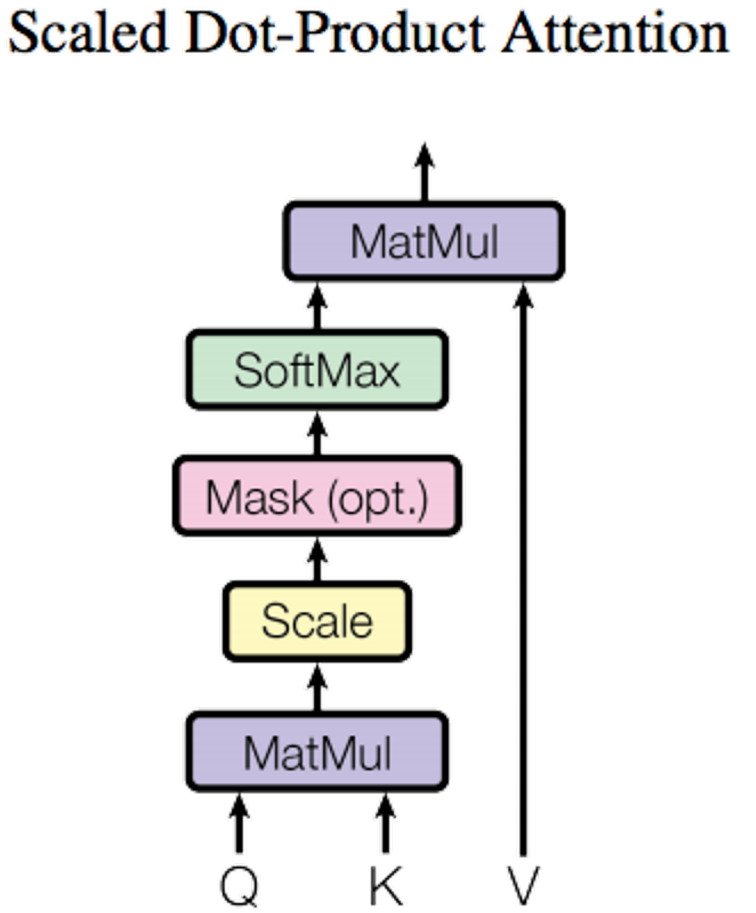


Modelling Sequences -- Transformers



Scaled Dot Product Attention

Efficient, stable training



Let $Z \in \mathbb{R}^{M \times d_z}$ be a matrix of task context vectors to attend to

Let $C \in \mathbb{R}^{N \times d_c}$ be a matrix of input vectors to attend over

$\boldsymbol{SDPAttention(Z, C)}$:

$Q = W_Q Z^T$ $\quad W_Q \in \mathbb{R}^{d_q \times d_z}$ $\quad d_q = d_k$

$K = W_K C^T$ $\quad W_K \in \mathbb{R}^{d_k \times d_c}$

$V = W_V C^T$ $\quad W_V \in \mathbb{R}^{d_v \times d_c}$

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

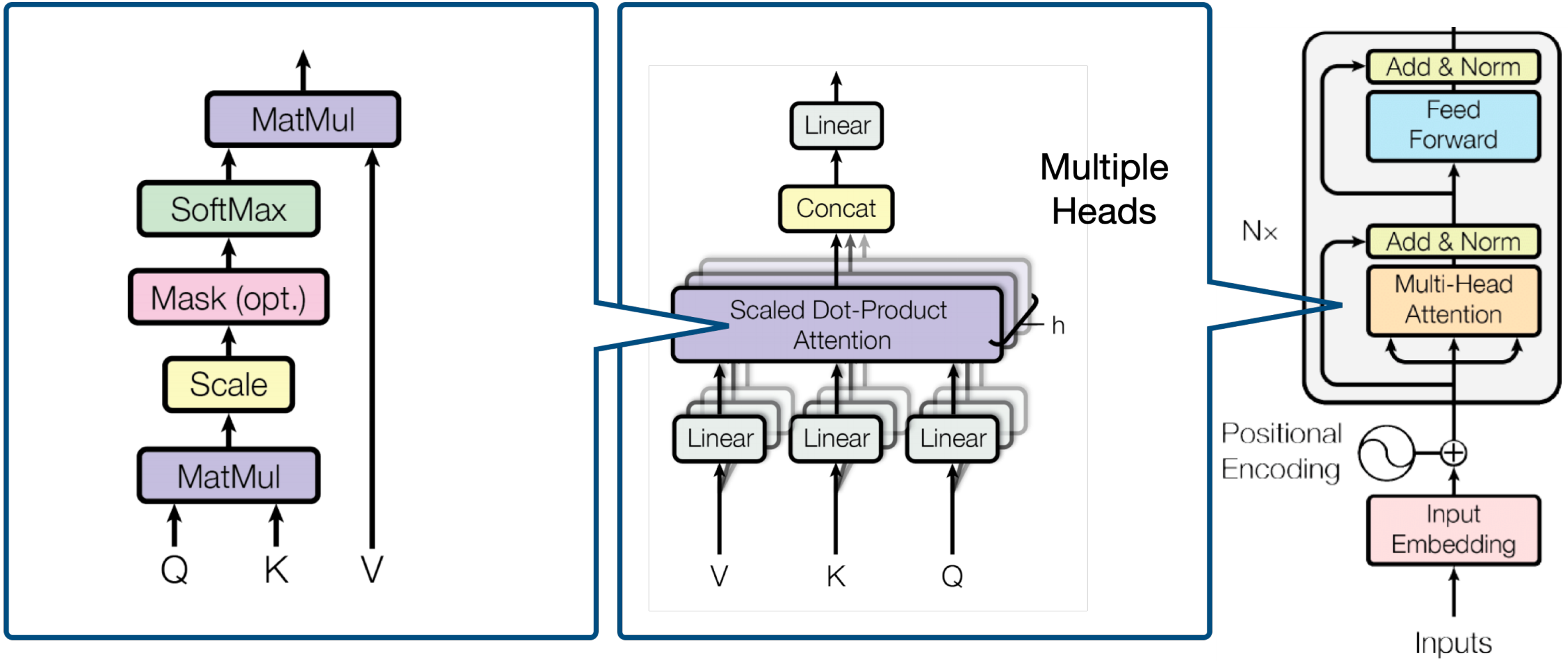
$\hat{V} \in \mathbb{R}^{M \times d_v}$ be a matrix of attended values

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

Modelling Sequences -- Transformers

Scaled Dot-Product Attention

self-attention $\boldsymbol{SDPAttention(C, C)}$:



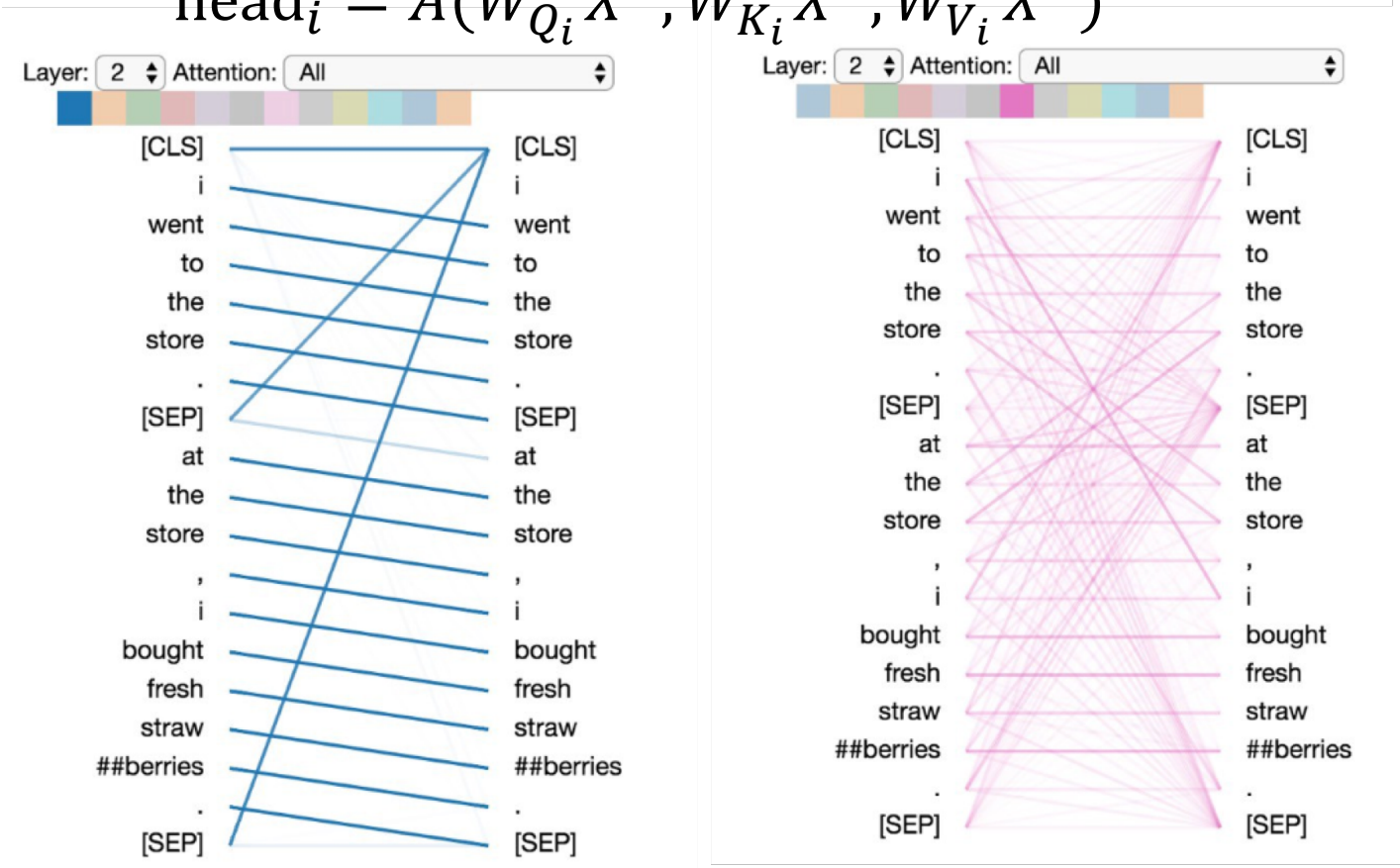
Multi-head attention

One head is not expressive enough. Let's have multiple heads!

$$A(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W_O$$

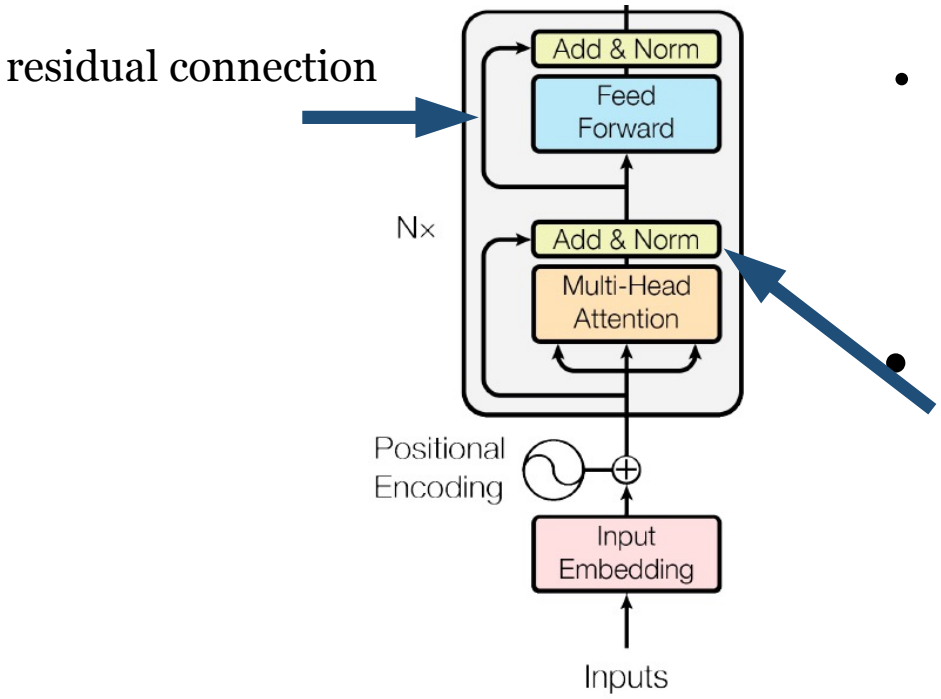
$$\text{head}_i = A(W_{Q_i}X^T, W_{K_i}X^T, W_{V_i}X^T)$$

In practice, $h = 8$,
$d = d_{out}/h, W_O \in \mathbb{R}^{d_{out} \times d_{out}}$



https://github.com/jessevig/bertviz

Putting it all together



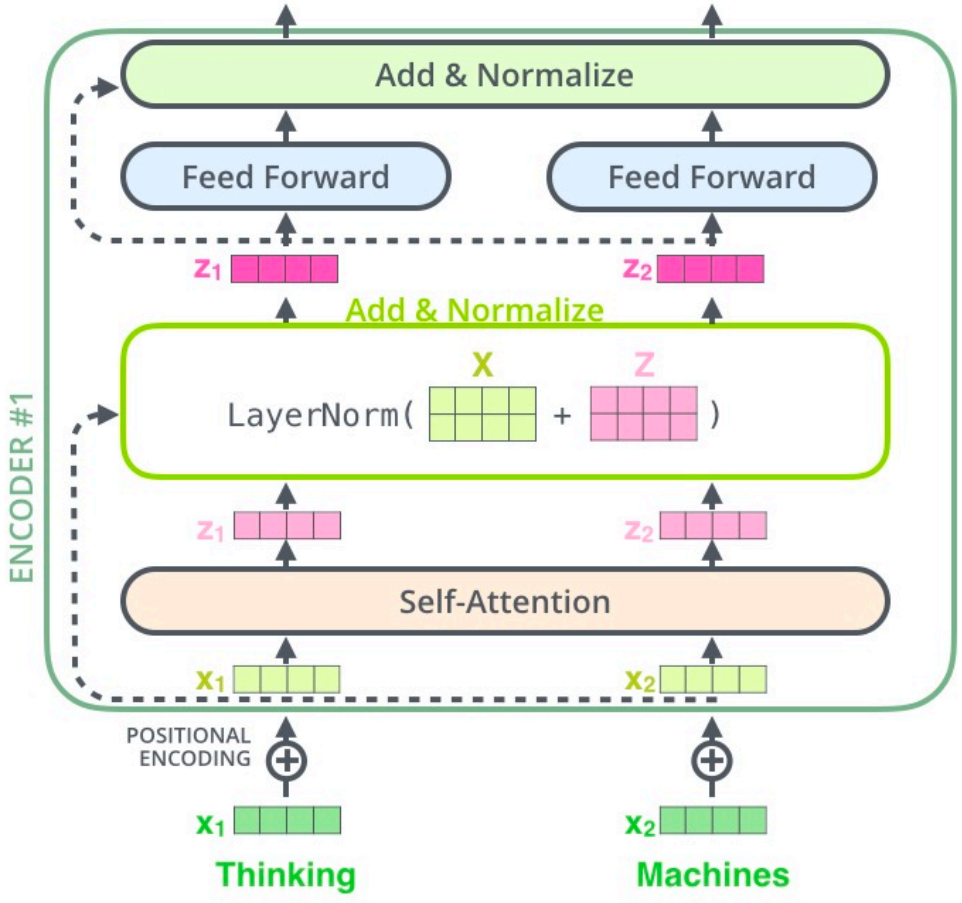
- Each Transformer block has two sub-layers
  - Multi-head attention
  - 2-layer feedforward NN (with ReLU) Provides non-linearity
- Each sublayer has a residual connection and a layer normalization

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

Helps the training process!

(Ba et al, 2016): Layer Normalization

Residual connections and Layer Normalization



(figure credit: [Jay Alammar](http://jalammar.github.io/illustrated-transformer/) http://jalammar.github.io/illustrated-transformer/)

For stable and efficient training

LayerNorm

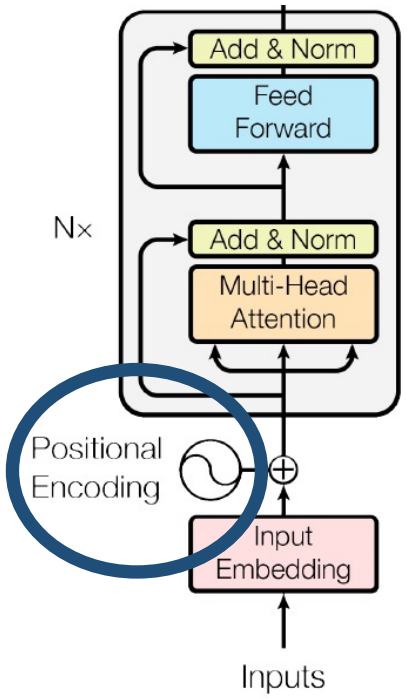
- changes input features to have mean 0 and variance 1 per layer.
- Adds two more parameters

$$\mu^l = \frac{1}{H}\sum_{i=1}^{H} a_i^l \qquad \sigma^l = \sqrt{\frac{1}{H}\sum_{i=1}^{H}\left(a_i^l - \mu^l\right)^2}$$

$$h_i = \frac{g_i}{\sigma_i}\left(a_i - \mu_i\right) + b_i$$

(Ba et al, 2016): Layer Normalization

Putting it all together



- Each Transformer block has two sub-layers
  - Multi-head attention
  - 2-layer feedforward NN (with ReLU)
- Each sublayer has a residual connection and a layer normalization

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

- Input layer has a positional encoding

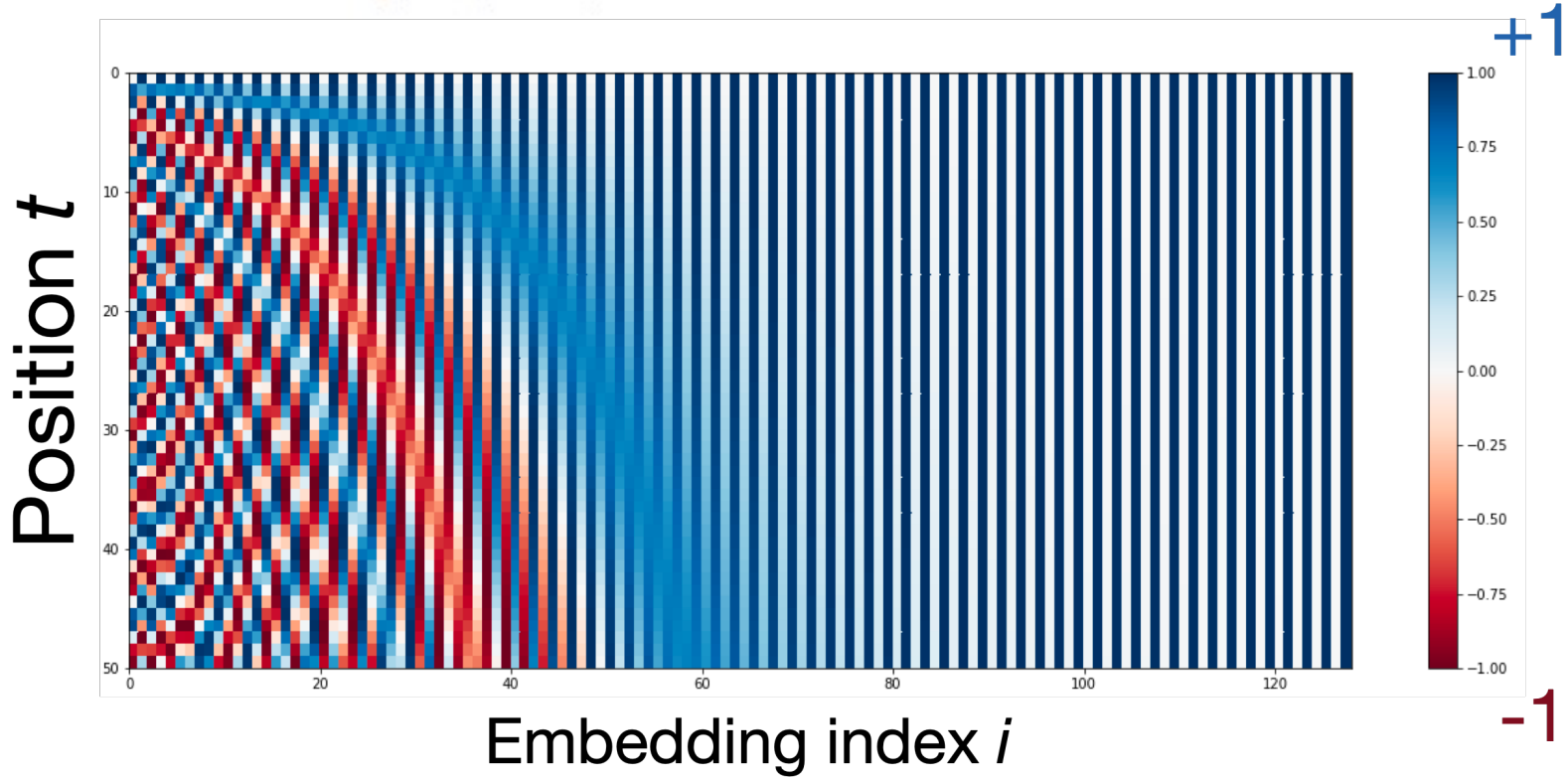
So the model knows where in the sequence the token is

(Ba et al, 2016): Layer Normalization

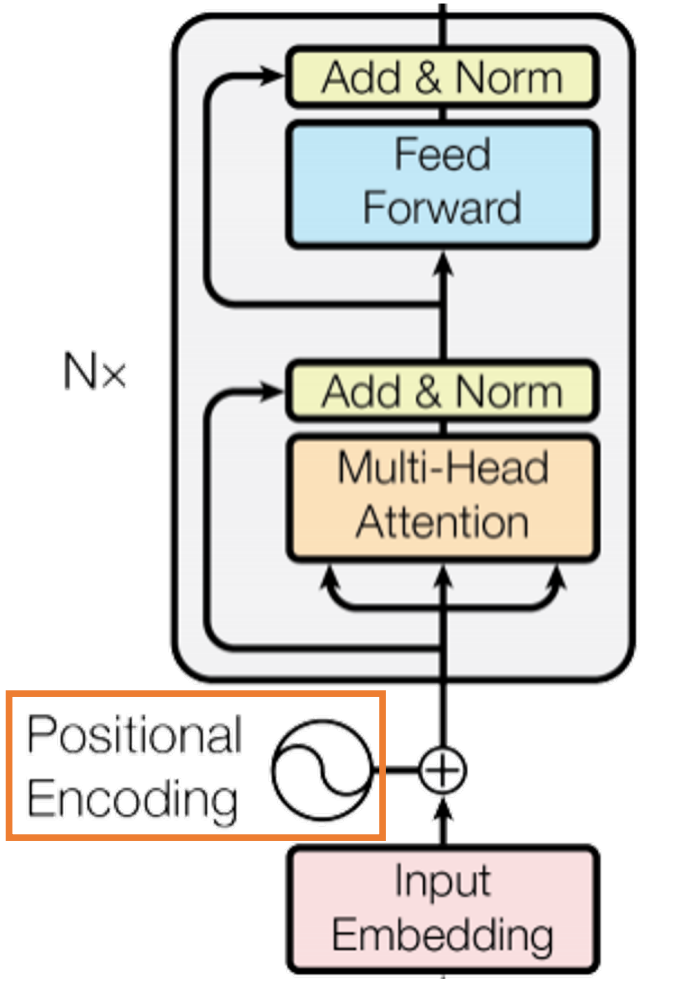
Transformers: Encoding position

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

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



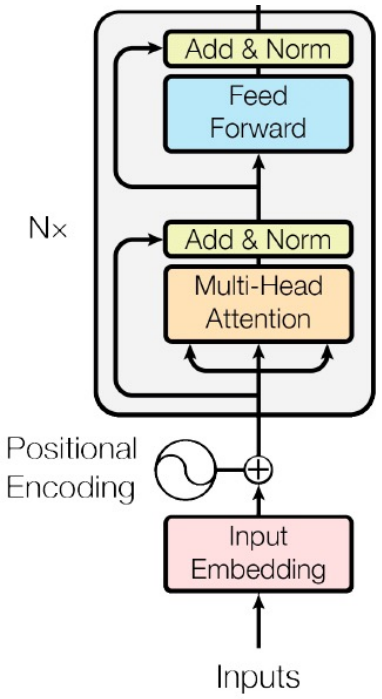
$\boldsymbol{SDPAttention(Y, Y)}$:



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

Putting it all together

Transformer
Non-recurrent, deep model with attention

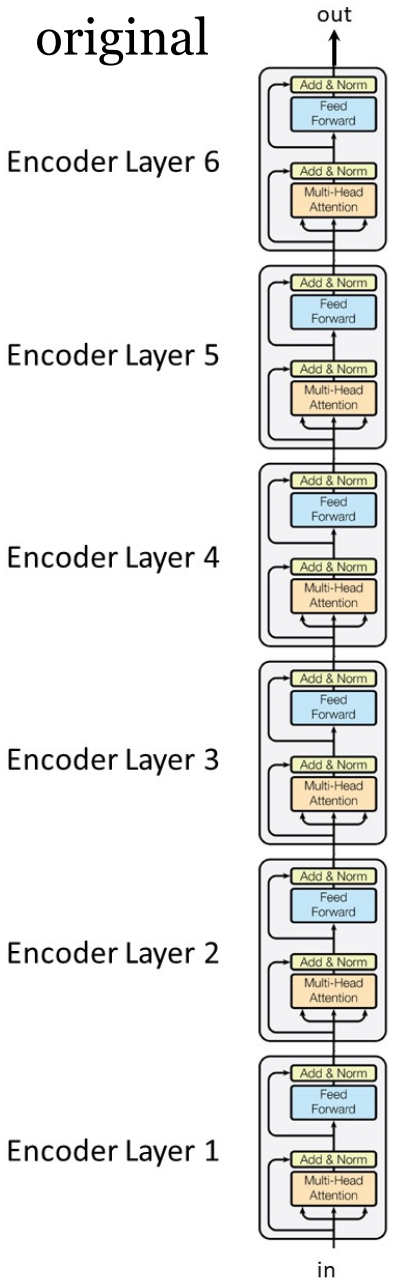


- Each Transformer block has two sub-layers
  - Multi-head attention
  - 2-layer feedforward NN (with ReLU)
- Each sublayer has a residual connection and a layer normalization

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

- Input layer has a positional encoding
- Input embedding is byte pair encoding (BPE)

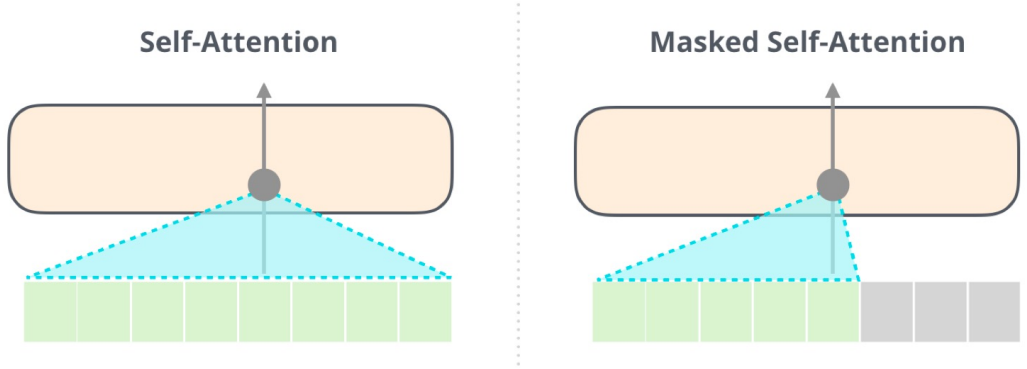
- BERT_base: 12 layers, 12 heads, hidden size = 768, 110M parameters
- BERT_large: 24 layers, 16 heads, hidden size = 1024, 340M parameters



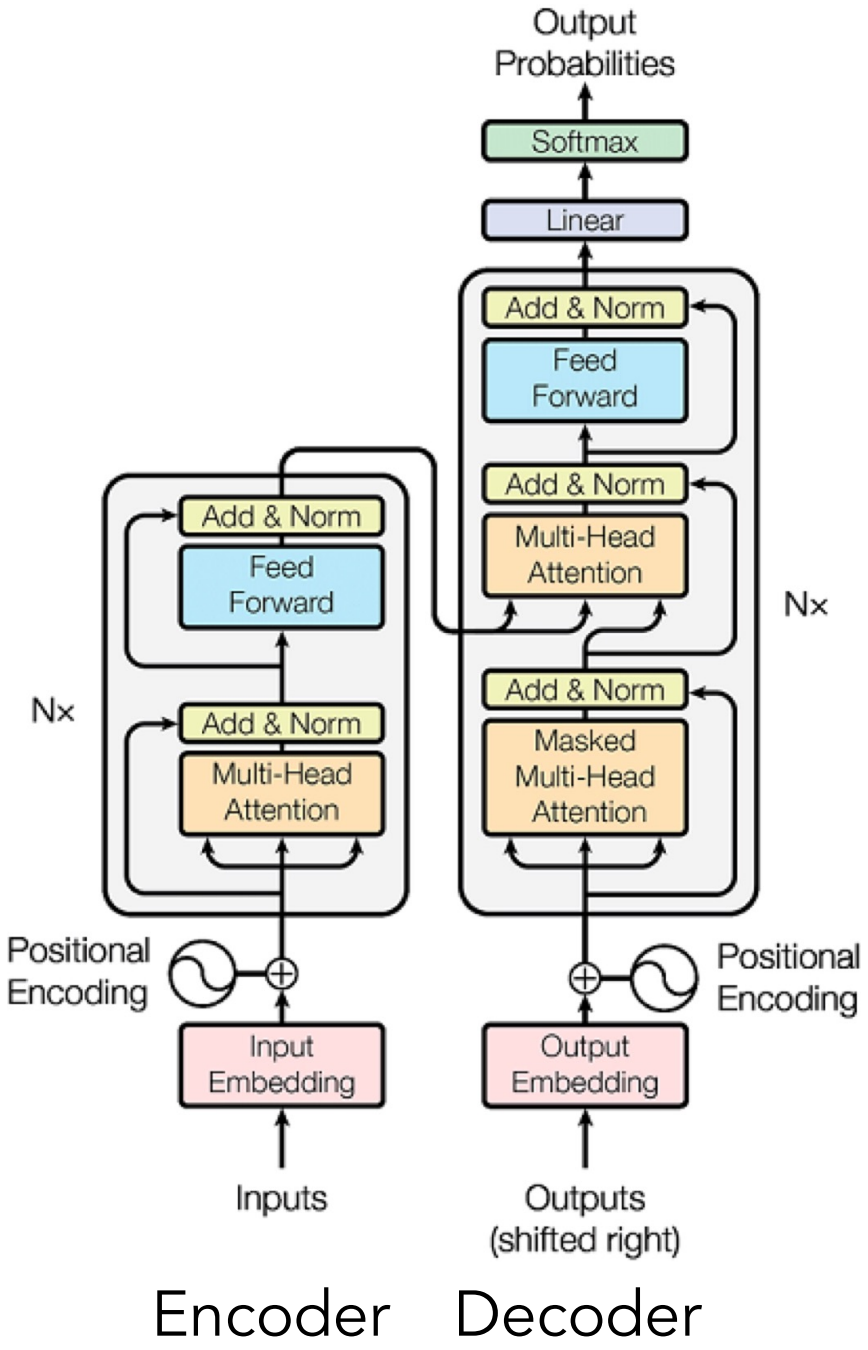
(Ba et al, 2016): Layer Normalization

Transformers

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

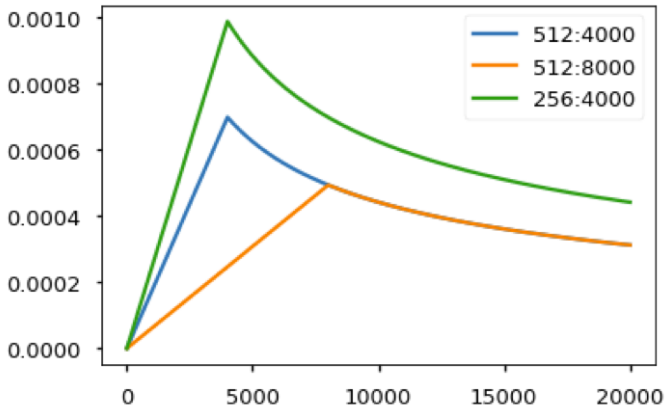


  - Cross attention
    - queries: previous decoder layer
    - keys/values: output of encoder
  - Autoregressive decoding

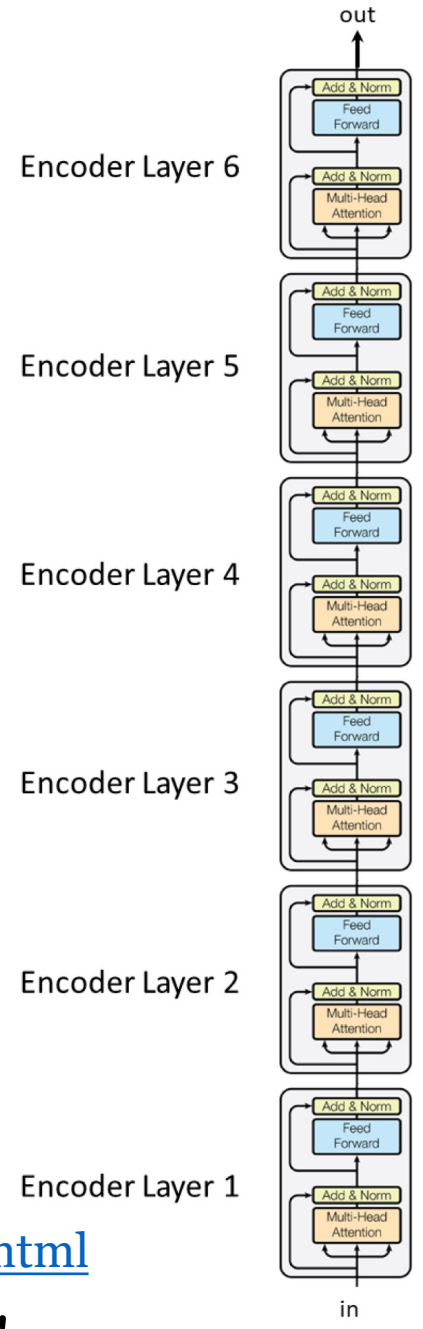


Transformers

- Stacked into multi-layers
- Subwords
  - Byte-pair encoding (BPE) / Word pieces
- 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:

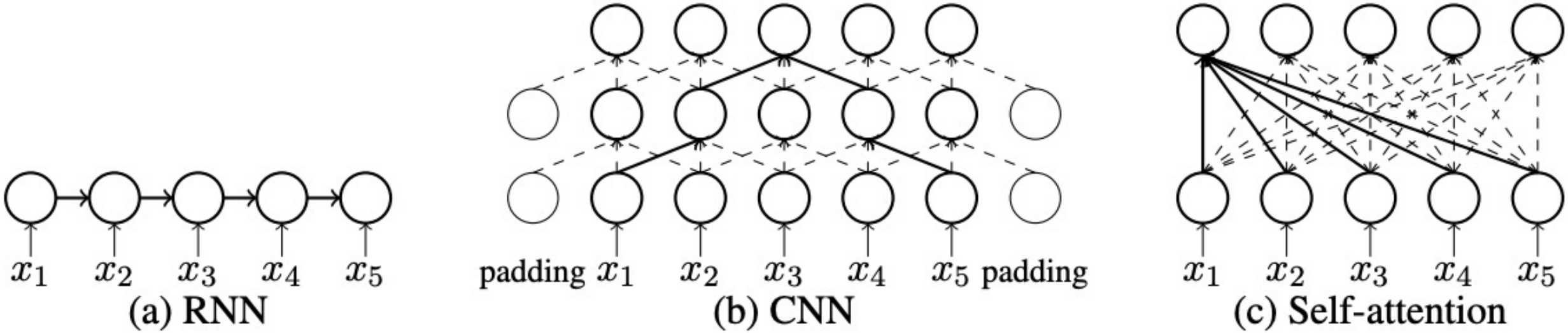
```
>>> encoder_layer = nn.TransformerEncoderLayer(d_model=512, nhead=8)
>>> transformer_encoder = nn.TransformerEncoder(encoder_layer, num_layers=6)
>>> src = torch.rand(10, 32, 512)
>>> out = transformer_encoder(src)
```

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

Other useful resources

- Deeper dive with Borealis AI Tutorials
  - Introduction (https://www.borealisai.com/en/blog/tutorial-14-transformers-i-introduction/)
    - Core of Transformers (Multi-Head Self-Attention, Positional Encoding)
    - Use in NLP (Sub-word tokenizations, BERT, GPT)
  - Extensions (https://www.borealisai.com/en/blog/tutorial-16-transformers-ii-extensions/)
    - Different choices for positional embedding
    - How to have more efficient attention for long sequences
    - Relation to other deep learning architectures
  - Training (https://www.borealisai.com/en/blog/tutorial-17-transformers-iii-training/)
    - Residual connections + LayerNorm
    - Learning rate warmup
    - Details of optimizer (Adam) and how it affects training

RNNs vs CNNs vs Transformers



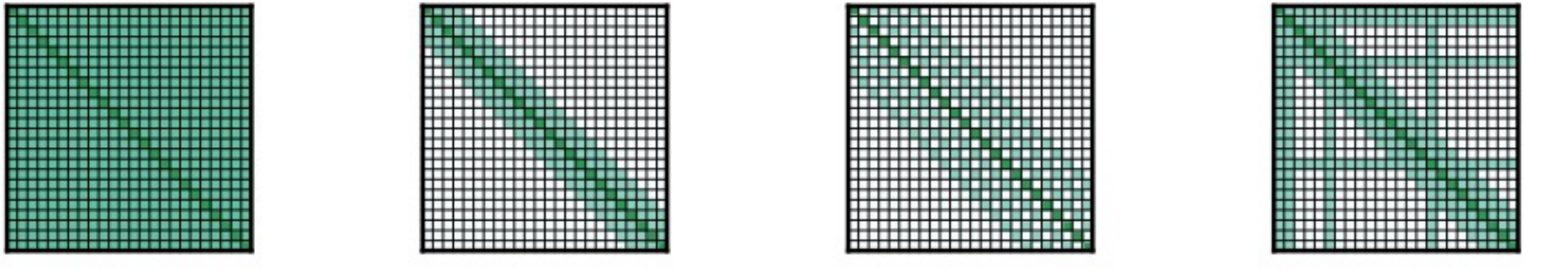
Why Self-Attention? A Targeted Evaluation of Neural Machine Translation Architectures [Tang et al, EMNLP 2018]

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)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

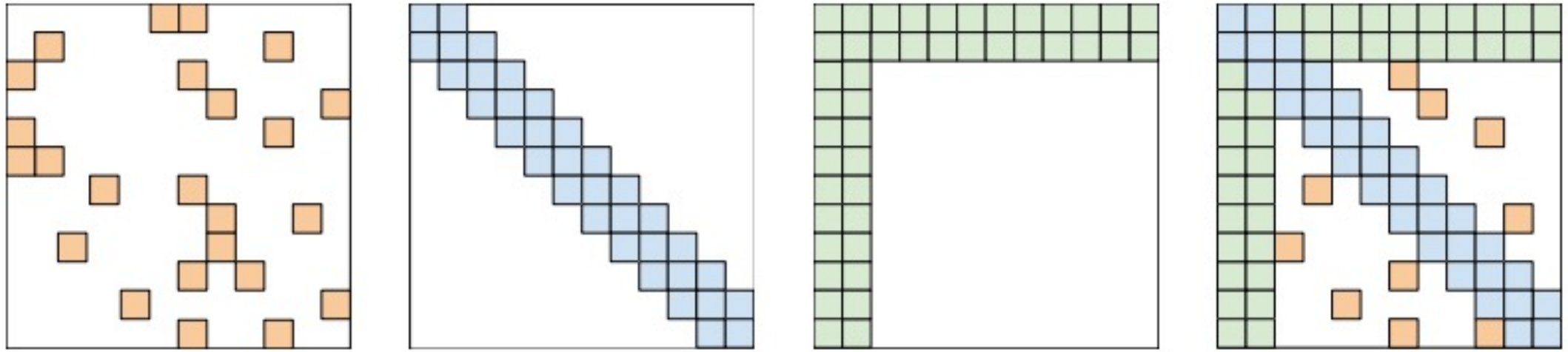
n: sequence length, *d*: representation dimensionality, *k*: kernel width, *r*: neighborhood size

Improving attention for long sequences



(a) Full n^2 attention (b) Sliding window attention (c) Dilated sliding window (d) Global+sliding window

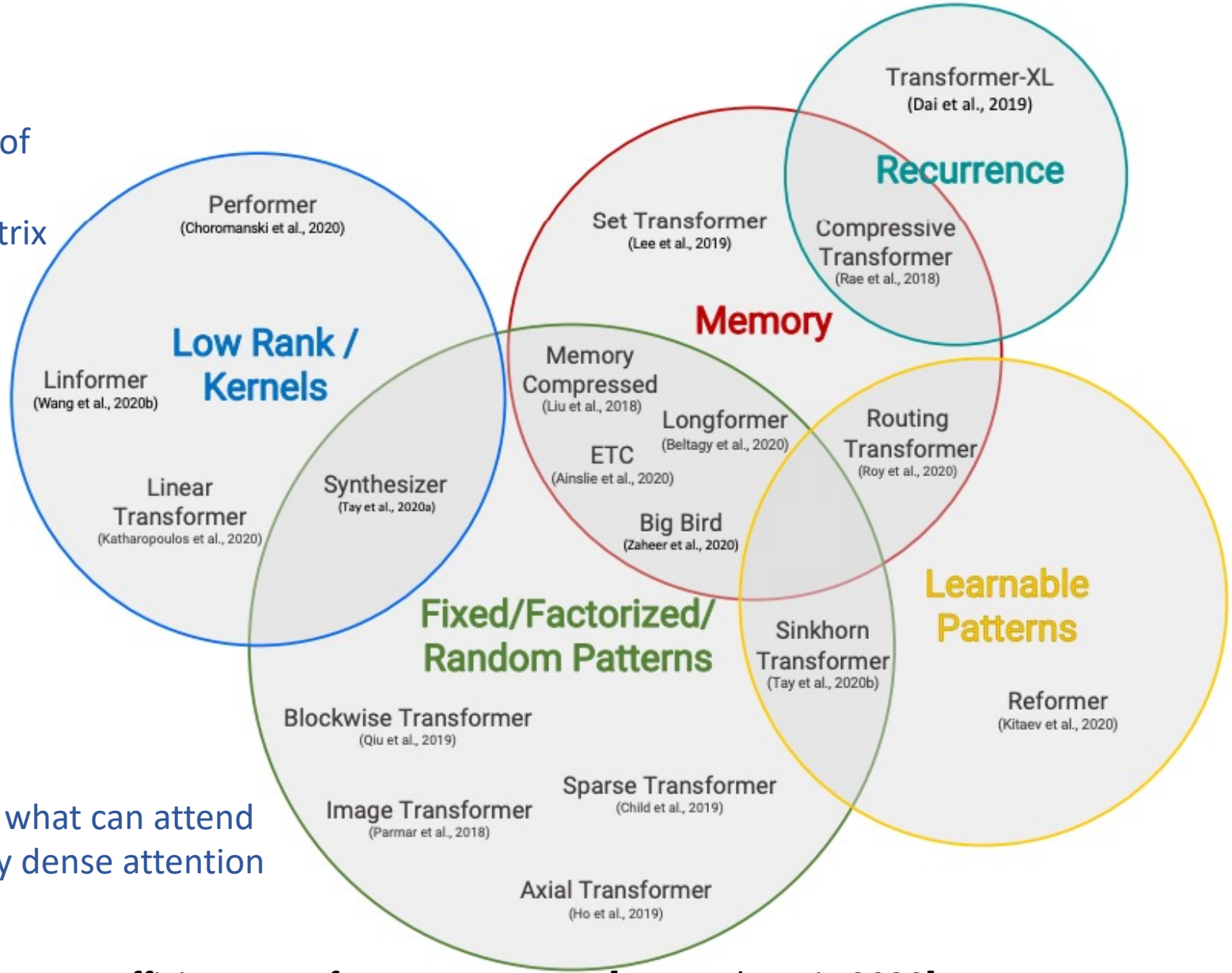
Longformer: The Long-Document Transformer [Beltagy et al, arXiv 2021], AI2



(a) Random attention (b) Window attention (c) Global Attention (d) BIGBIRD

Big Bird: Transformers for Longer Sequences [Zaheer et al, arXiv 2021], Google Research

Decomposition and use of kernels to avoid explicit computation of NxN matrix



Ways to limited what can attend to what: sparsify dense attention

Efficient Transformers: A survey [Tay et al, arXiv 2020]

Ongoing research on transformers

- Ways to make transformers more efficient
  - How to improve the attention mechanism?
  - How to train transformers more efficiently?
  - How to reduce the size of transformer models?

- What does transformer models learn?

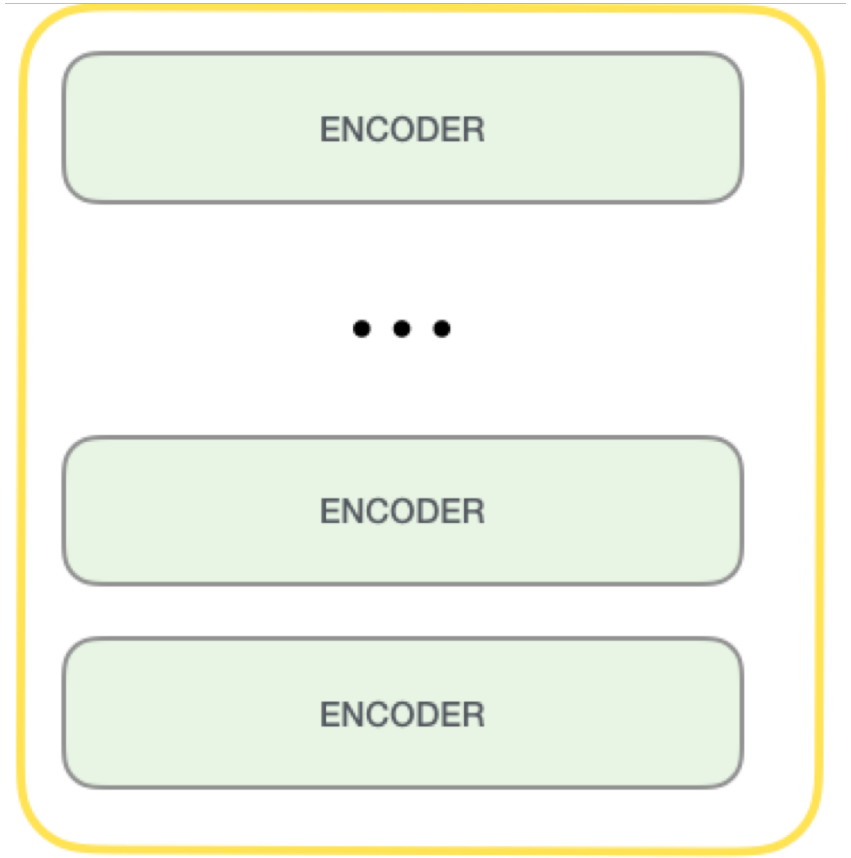
- Applying transformers to other domains:
  - Images, 3D data, audio, …

Pretraining Word Embeddings

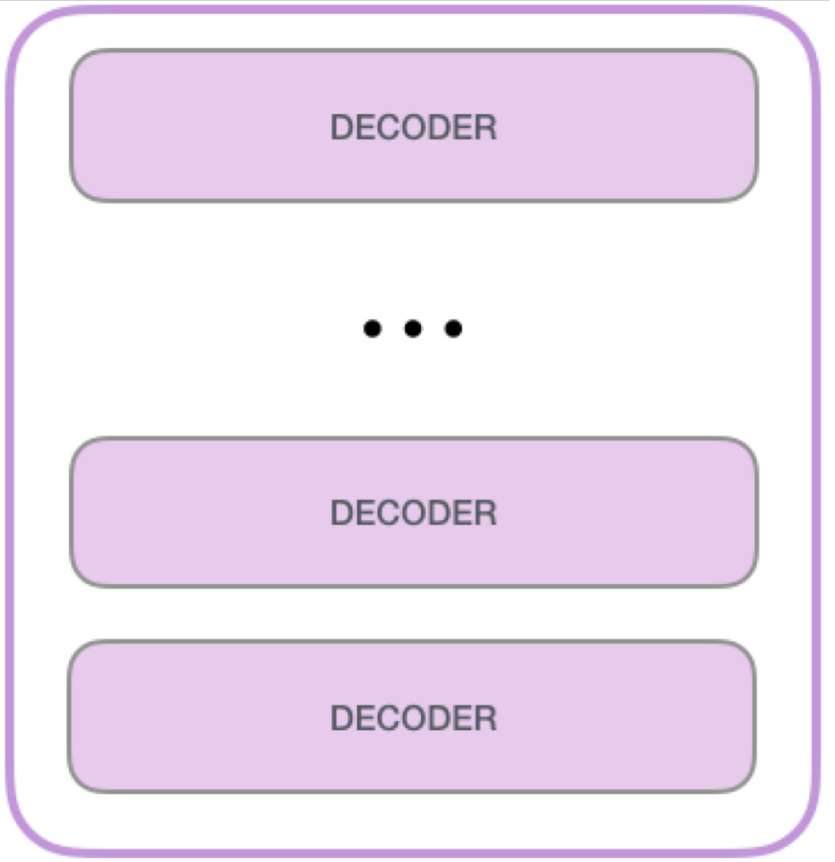
- Self-supervised Transformer based models shattered language understanding benchmarks in NLP in 2018.

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

BERT: uses encoders



GPT: uses decoders



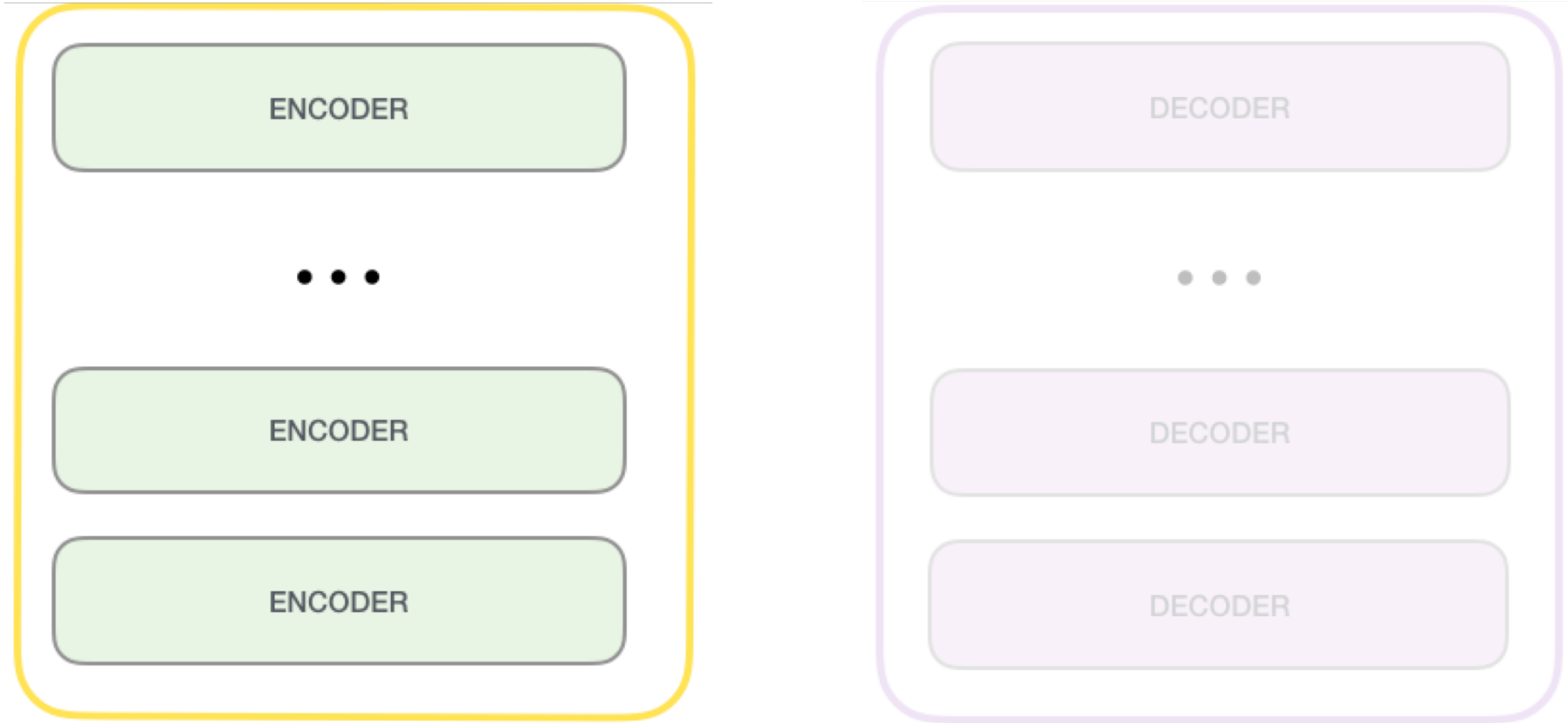
(figure credit: Jay Alammar http://jalammar.github.io/illustrated-gpt2/)

- 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: uses encoders

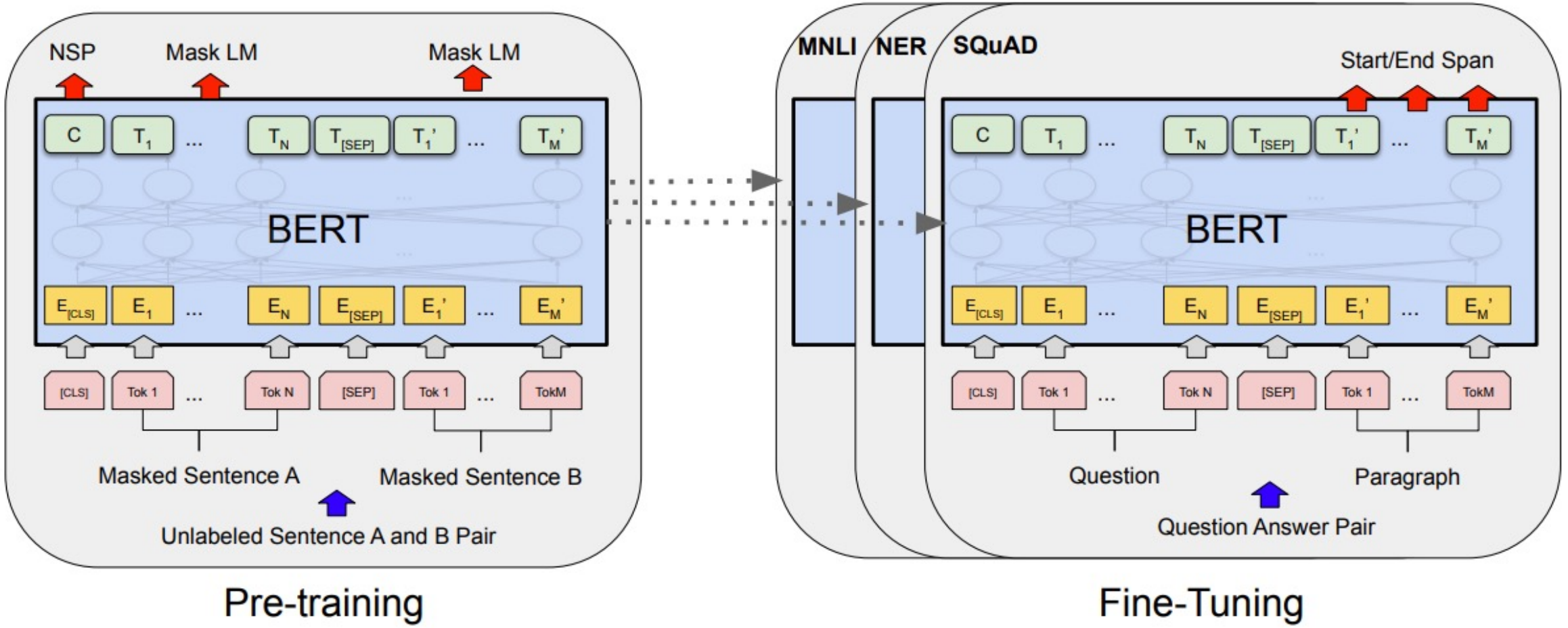
GPT: uses decoders



(figure credit: Jay Alammar http://jalammar.github.io/illustrated-gpt2/)

Pretraining

Task-specific fine-tuning

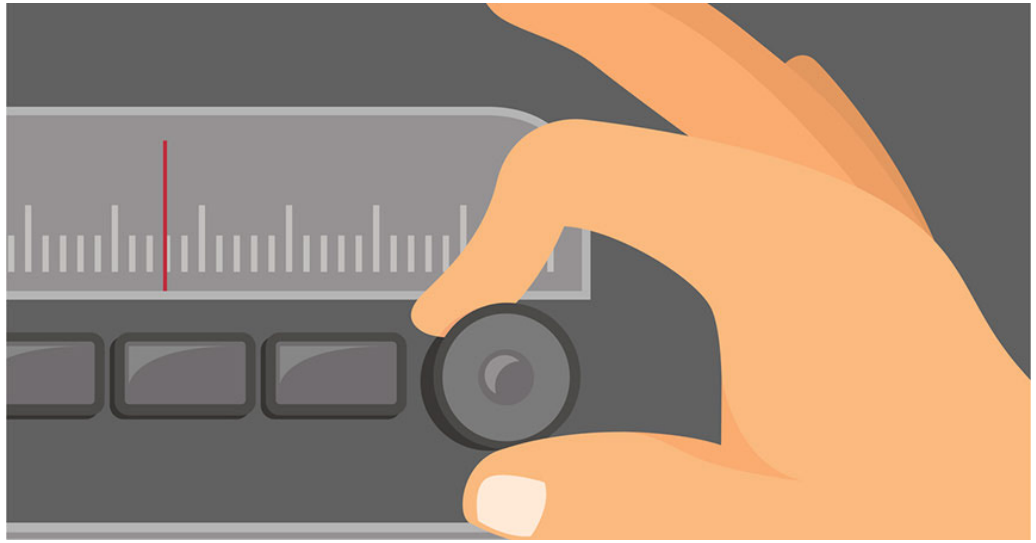


Pretraining

- 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
  - Example: Vision backbones on ImageNet
  - Great but requires labeled images!
- Pretraining with autoencoders
  - Just find lots of data online
- Pretraining by language modeling
  - Only need lots of text data

Traditional

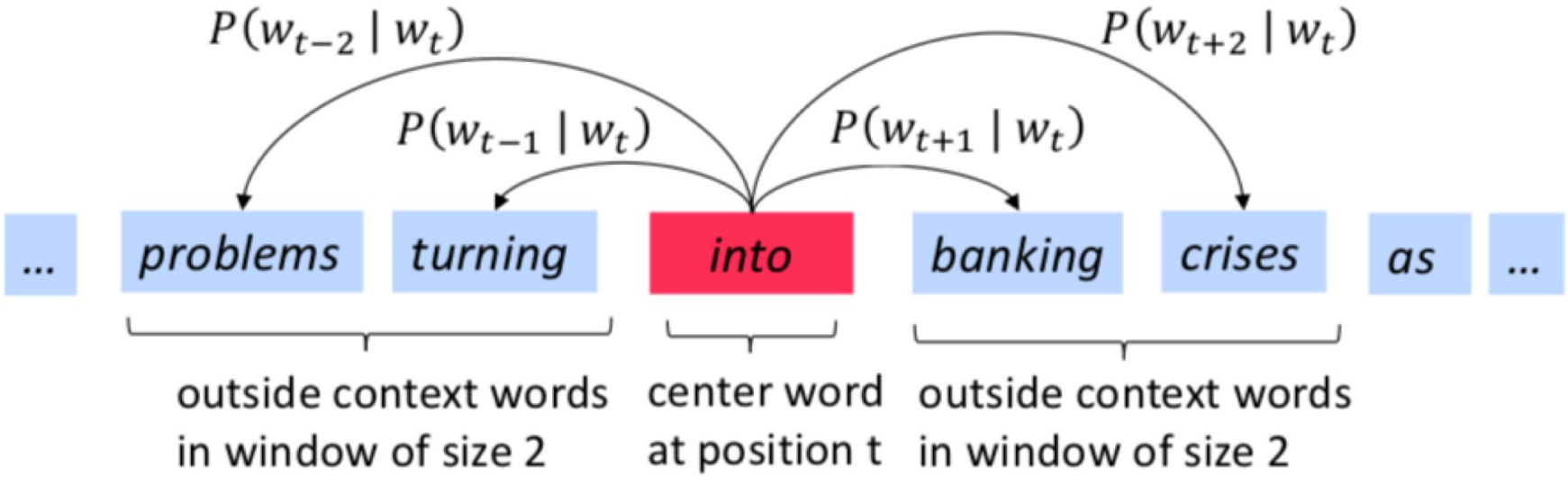
$$P(w_t|w_{<t})$$

Masked

$$P(w_t|w_{<t}, w_{>t})$$

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)

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

- Traditionally used statistical n-grams

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

- Now with neural models

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

- Can mask out any word

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

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

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

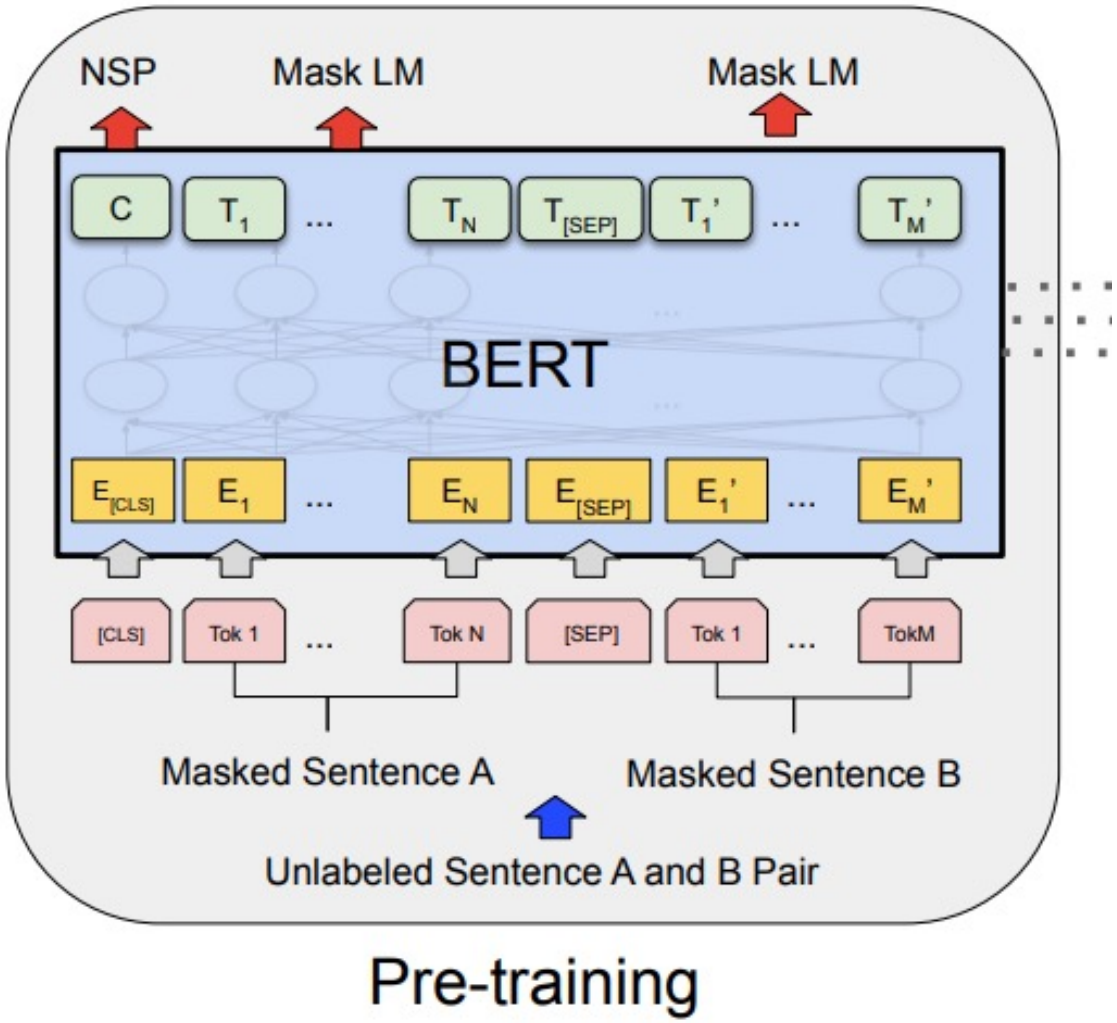
  `my dog is apple`

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

  `my dog is hairy`

Modelling Sequences -- Transformers

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



Pre-training

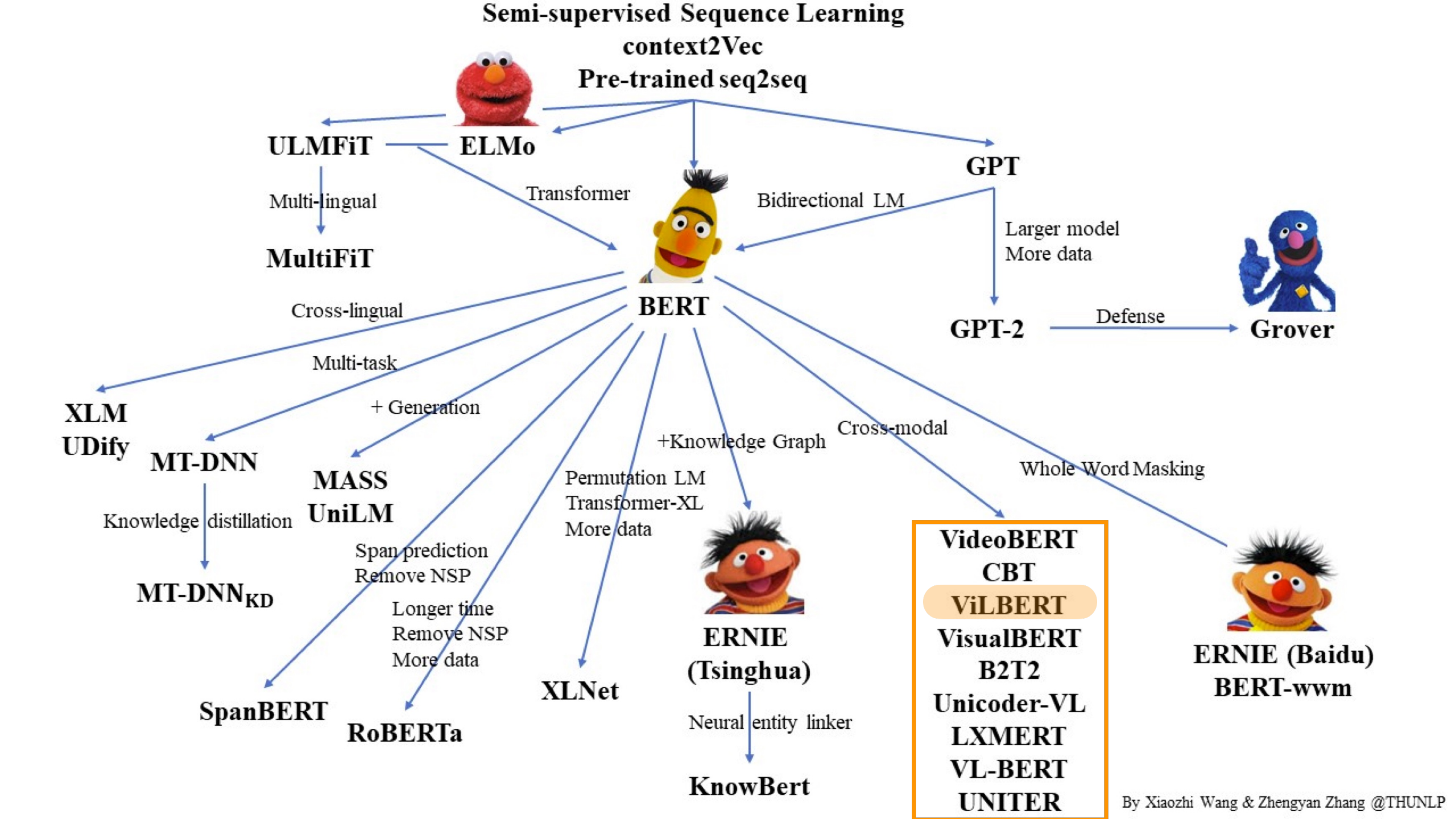
BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding https://arxiv.org/pdf/1810.04805.pdf

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) 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	79.8	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**

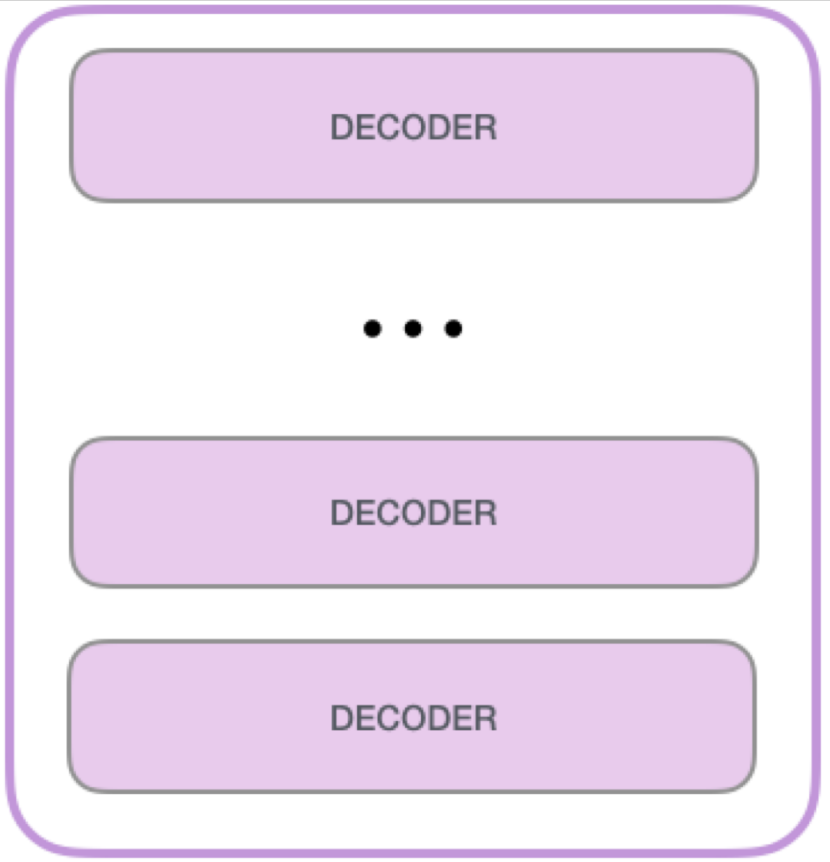

Semi-supervised Sequence Learning
context2Vec
Pre-trained seq2seq
ELMo
ULMFiT
Multi-lingual
MultiFiT
Transformer
BERT
Bidirectional LM
GPT
Larger model
More data
GPT-2
Defense
Grover
Cross-lingual
XLM
UDify
Multi-task
MT-DNN
Knowledge distillation
MT-DNN$_{KD}$
+ Generation
MASS
UniLM
Span prediction
Remove NSP
SpanBERT
Longer time
Remove NSP
More data
RoBERTa
Permutation LM
Transformer-XL
More data
XLNet
+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

- 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: uses encoders

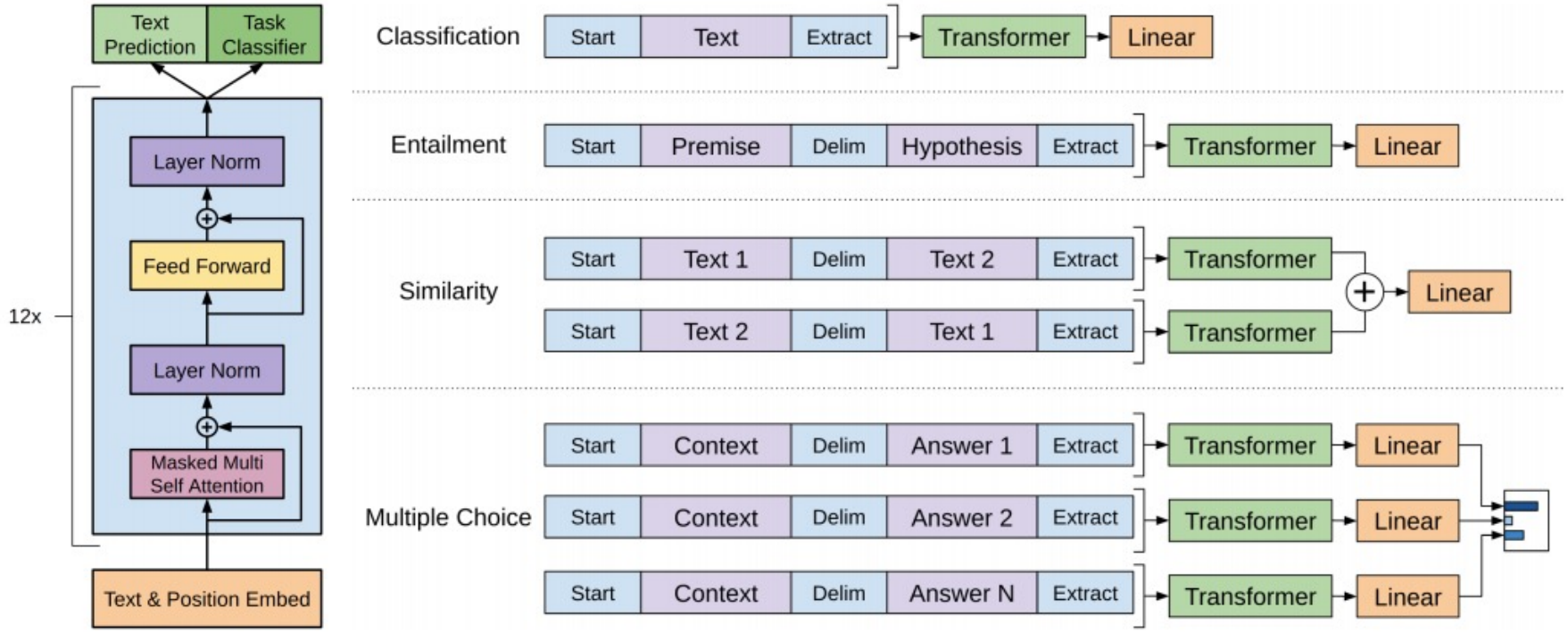
GPT: uses decoders



(figure credit: Jay Alammar http://jalammar.github.io/illustrated-gpt2/)

GTP (Generative pretrained transformer)

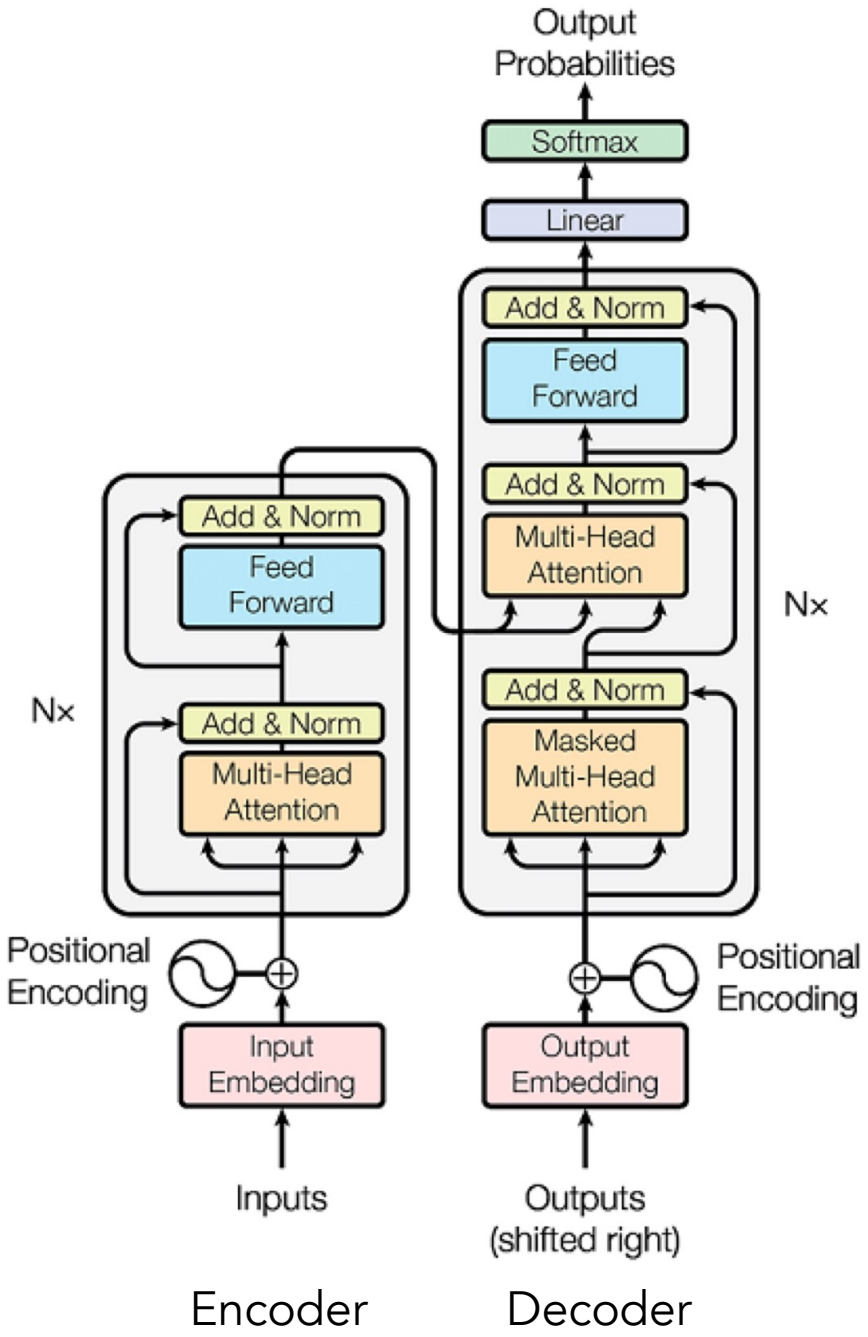
- Unsupervised pretraining: Standard language model loss
- Supervised fine-tuning: Use simple classifier (linear layer + softmax) trained to predict correct class (use combined loss)



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

Transformer

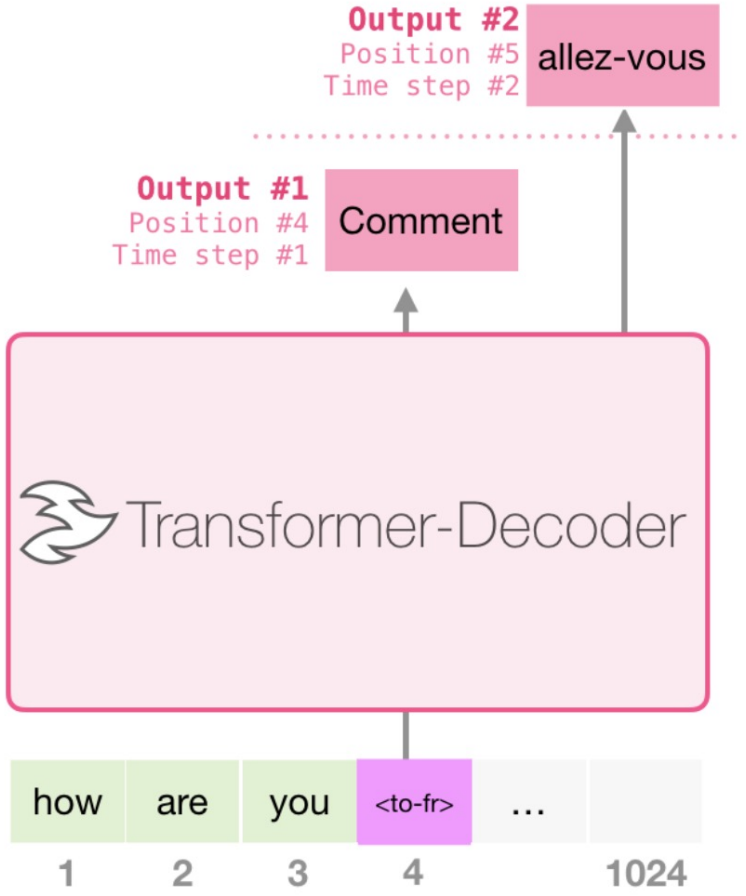
- Encoder and Decoder blocks are not that different!



GPT-2

- Express all tasks as a language modeling task
- Training improvements
  - Improve initialization / additional layer normalization
  - Increased vocabulary / context / batch size
- 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				



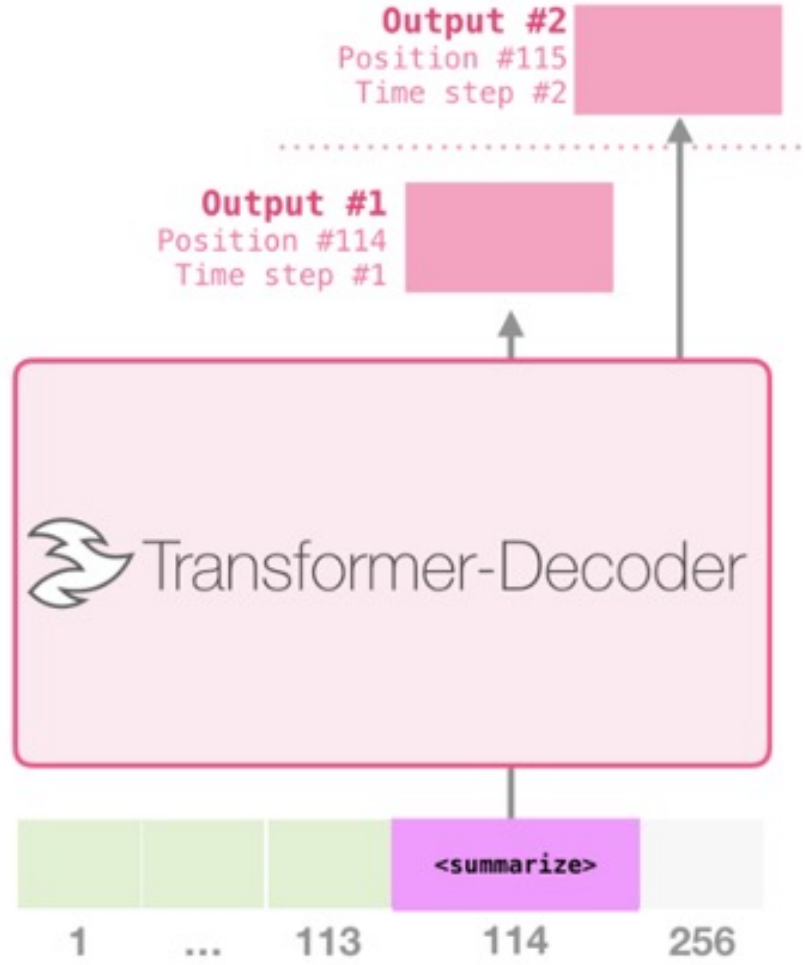
(figure credit: [Jay Alammar](http://jalammar.github.io/illustrated-gpt2/) http://jalammar.github.io/illustrated-gpt2/)

GPT-2

- Express all tasks as a language modeling task

- Summarization

Article #1 tokens		<summarize>	Article #1 Summary
Article #2 tokens	<summarize>	Article #2 Summary	padding
Article #3 tokens		<summarize>	Article #3 Summary



(figure credit: Jay Alammar http://jalammar.github.io/illustrated-gpt2/)

GPT models

- GPT
  - 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)
- GPT-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
- GPT-3
  - Language Models are Few-Shot Learners (Brown et al, 2020)
  - Uses Sparse Transformers
  - Trained on Web+Books+Wikipedia (300B words), 175B parameters (96 layers)

What goes into a pretrained model?

Papers will investigate differences in:

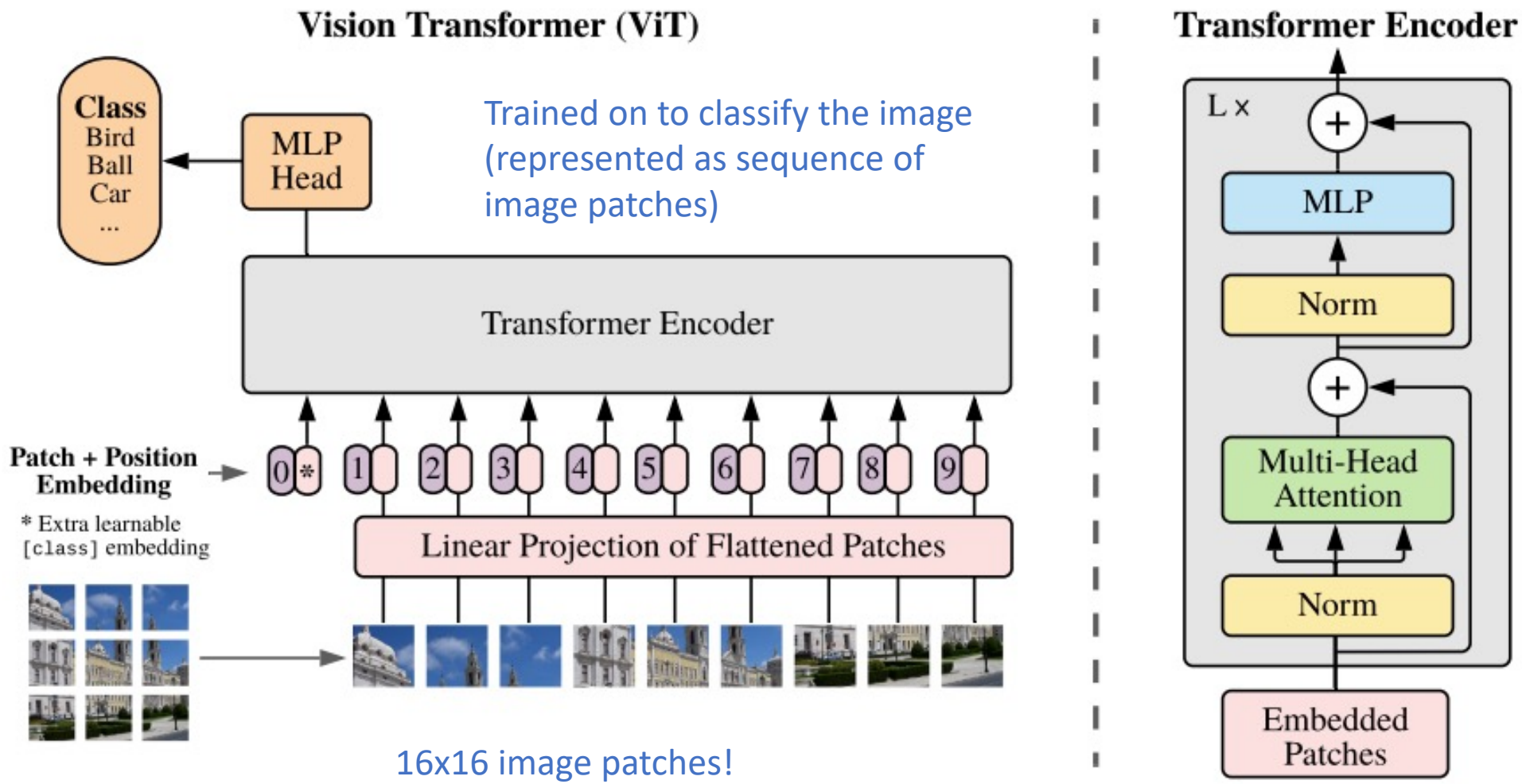
- Model
  - Architecture
  - Size (number of layers, embedding size, etc)
- Training Objective
  - Traditional LM
  - Masked LM
  - Contrastive loss
- Pretraining Dataset
  - Larger is better

Other things that matter:

- Tokenization
- Implementation and training details

Pretraining with transformers for images and 3D

Supervised pre-training using image labels



An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale, Dosovitskiy et al, ICLR 2021

Visual transformer performance

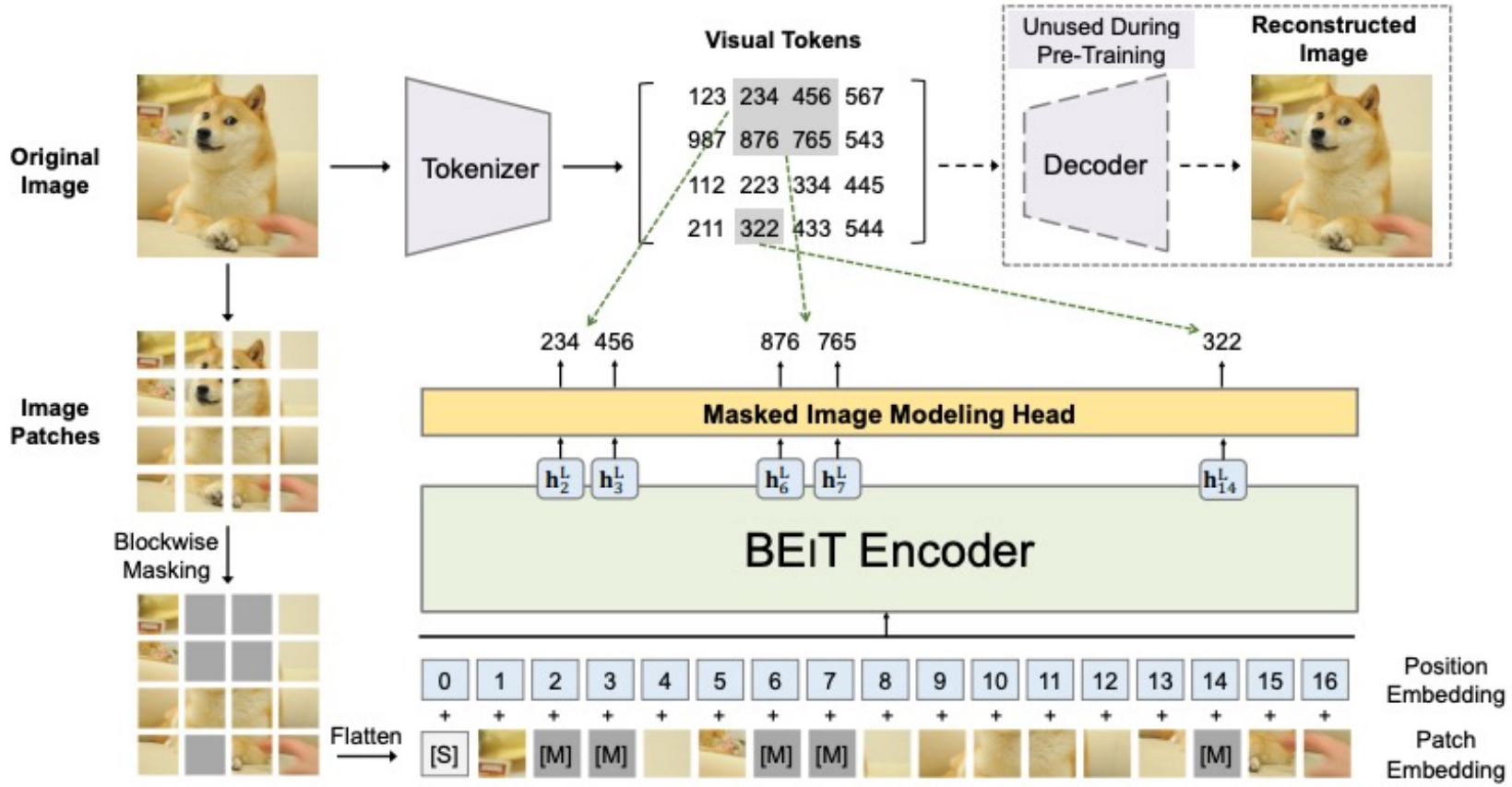
Can outperformance Big Transfer (BiT) (Kolesnikov et al., 2020), supervised pre-training with ResNet

	Ours-JFT (ViT-H/14)	Ours-JFT (ViT-L/16)	Ours-I21K (ViT-L/16)	BiT-L (ResNet152x4)	Noisy Student (EfficientNet-L2)
ImageNet	**88.55** ± 0.04	87.76 ± 0.03	85.30 ± 0.02	87.54 ± 0.02	88.4/88.5*
ImageNet ReaL	**90.72** ± 0.05	90.54 ± 0.03	88.62 ± 0.05	90.54	90.55
CIFAR-10	**99.50** ± 0.06	99.42 ± 0.03	99.15 ± 0.03	99.37 ± 0.06	–
CIFAR-100	**94.55** ± 0.04	93.90 ± 0.05	93.25 ± 0.05	93.51 ± 0.08	–
Oxford-IIIT Pets	**97.56** ± 0.03	97.32 ± 0.11	94.67 ± 0.15	96.62 ± 0.23	–
Oxford Flowers-102	99.68 ± 0.02	**99.74** ± 0.00	99.61 ± 0.02	99.63 ± 0.03	–
VTAB (19 tasks)	**77.63** ± 0.23	76.28 ± 0.46	72.72 ± 0.21	76.29 ± 1.70	–
TPUv3-core-days	2.5k	0.68k	0.23k	9.9k	12.3k

An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale, Dosovitskiy et al, ICLR 2021

Self-supervised training: BERT for images

- Masked Image Modeling (MIM): Predict visual tokens in a masked fashion



BEIT: BERT Pre-training of image transformers, Bao et al, ICLR 2022

Self-supervised training: sequence modeling

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

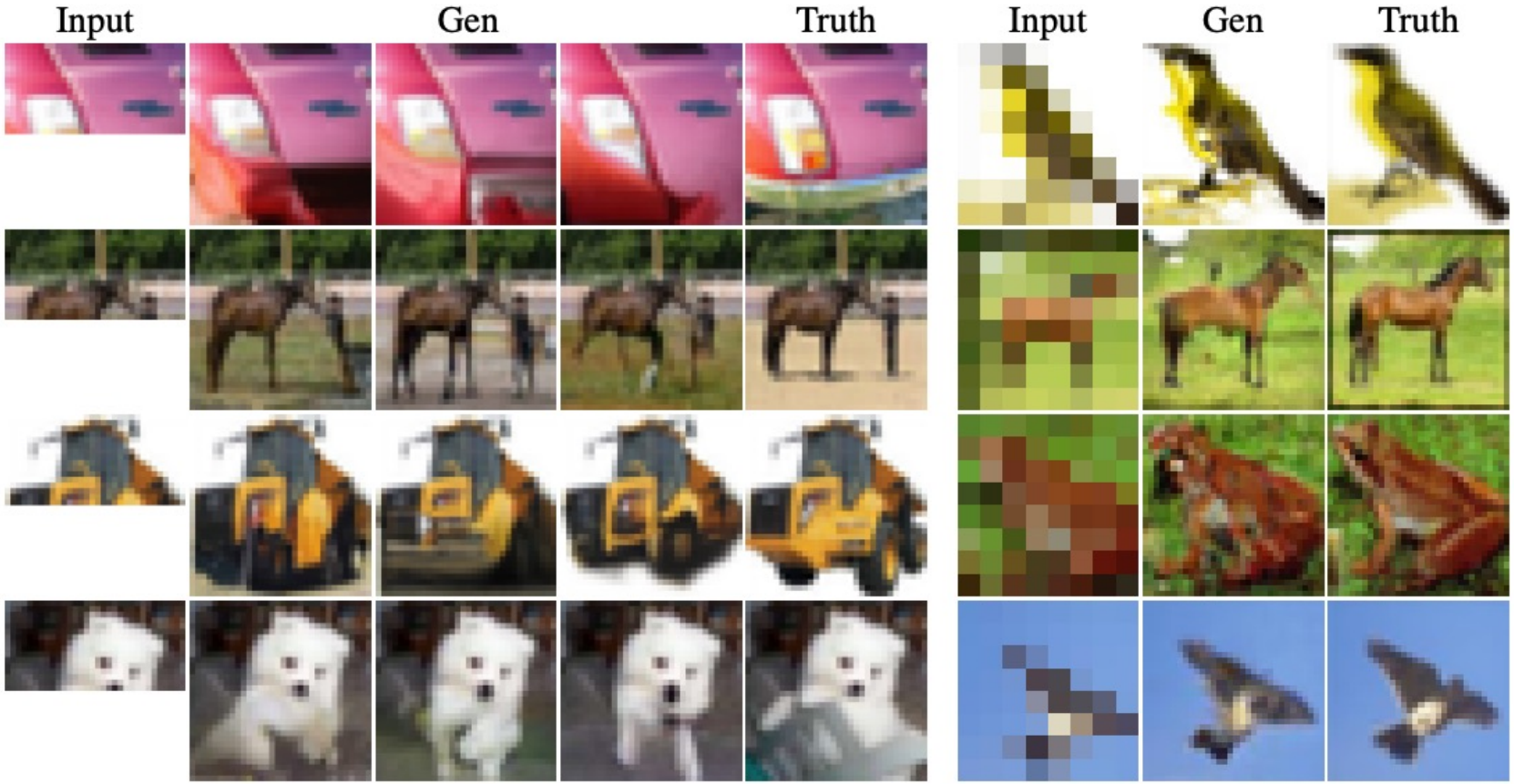


Image Transformer, Parmar et al, ICML 2018

Self-supervised training: sequence modeling

- 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

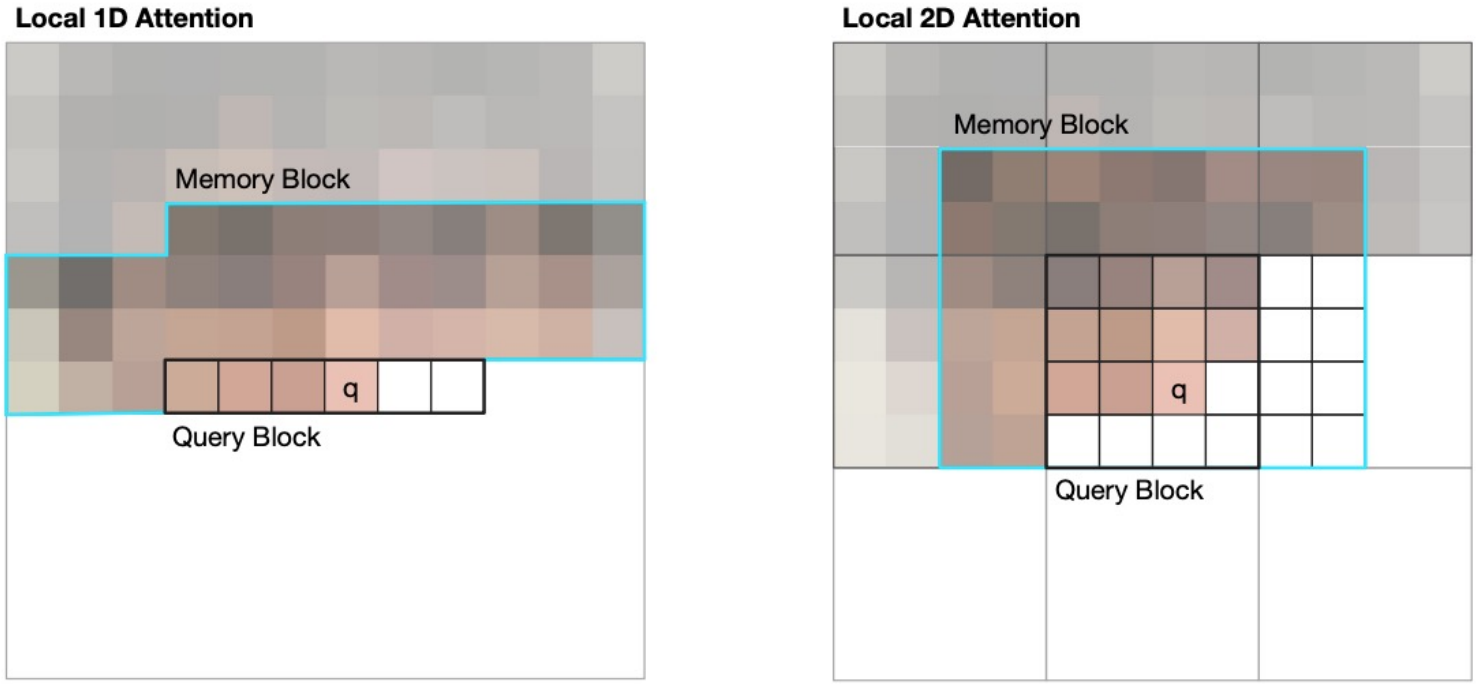


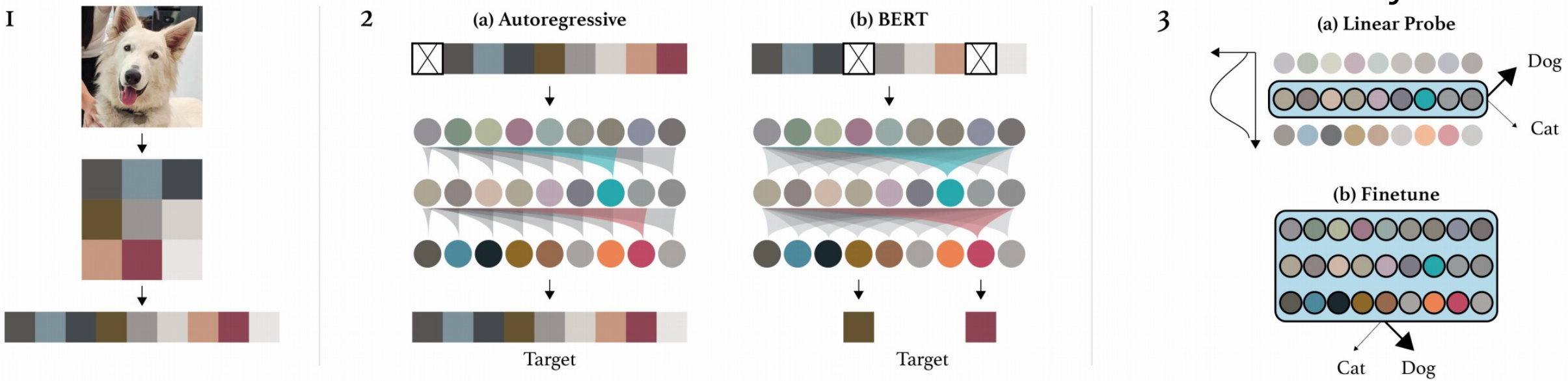
Image Transformer, Parmar et al, ICML 2018

Image GPT

Autoregressive

Masked

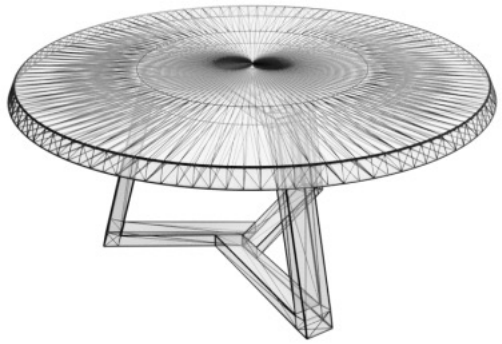
Classify using features from an intermediate or final layer



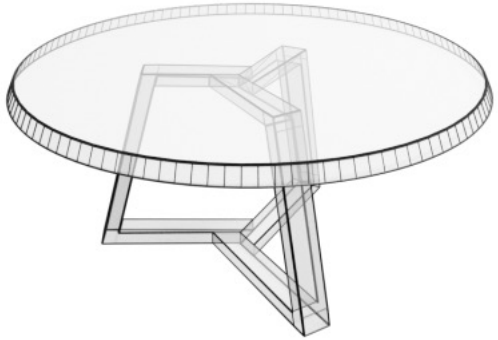
Generative Pretraining from Pixels, Chen et al, ICML 2020
https://openai.com/blog/image-gpt/

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



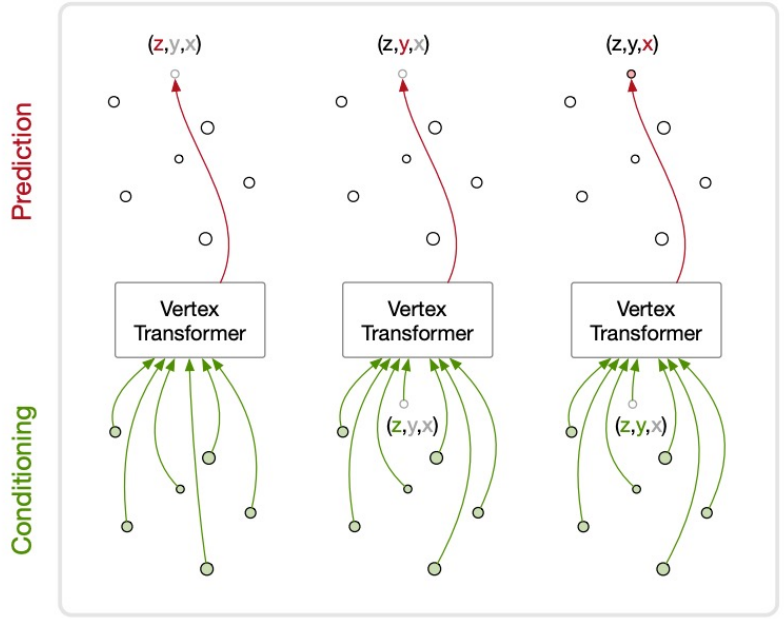
(b) n-gon mesh

$$p(\mathcal{M}) = p(\mathcal{V}, \mathcal{F})$$
$$= p(\mathcal{F}|\mathcal{V})p(\mathcal{V})$$

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

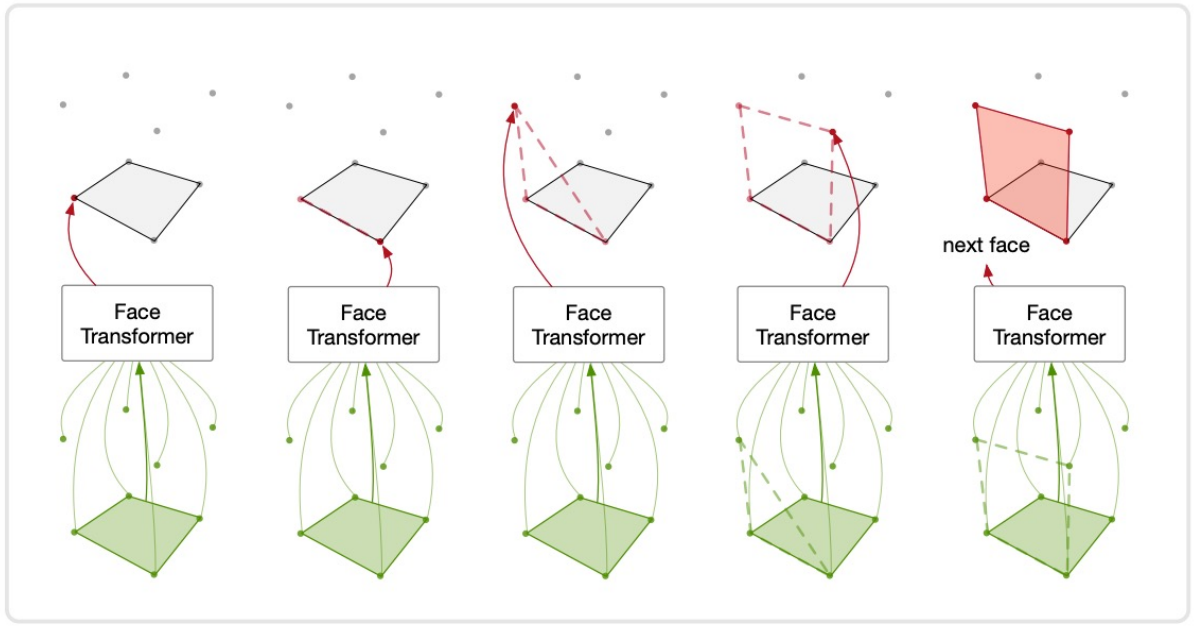
Pretraining for shapes

Vertex Model



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

Face Model



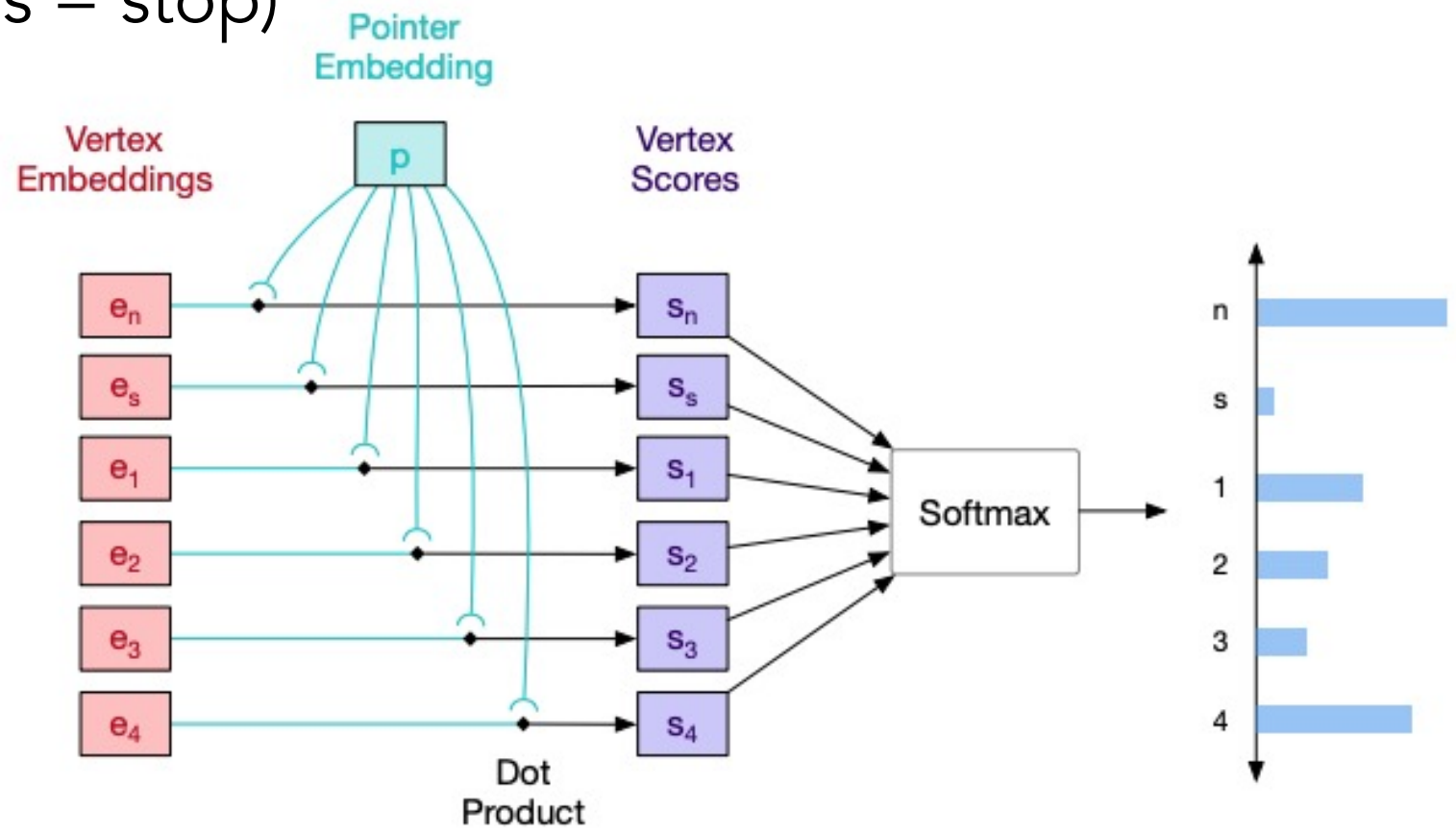
$$p(\mathcal{F}^{\text{seq}}|\mathcal{V};\theta) = \prod_{n=1}^{N_F} p(f_n|f_{<n},\mathcal{V};\theta)$$

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)

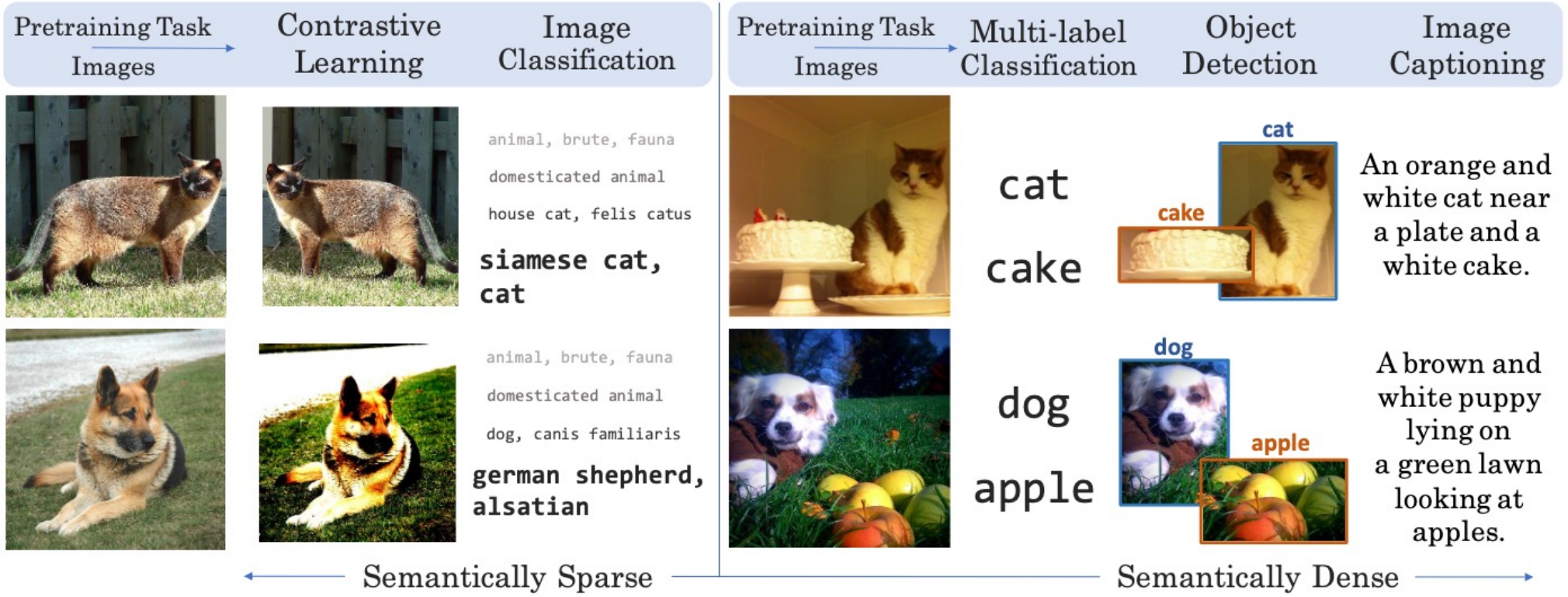


PolyGen: An Autoregressive Generative Model of 3D Meshes, Nash et al, ICML 2020

Pretraining with vision and language

Pretraining for vision with language

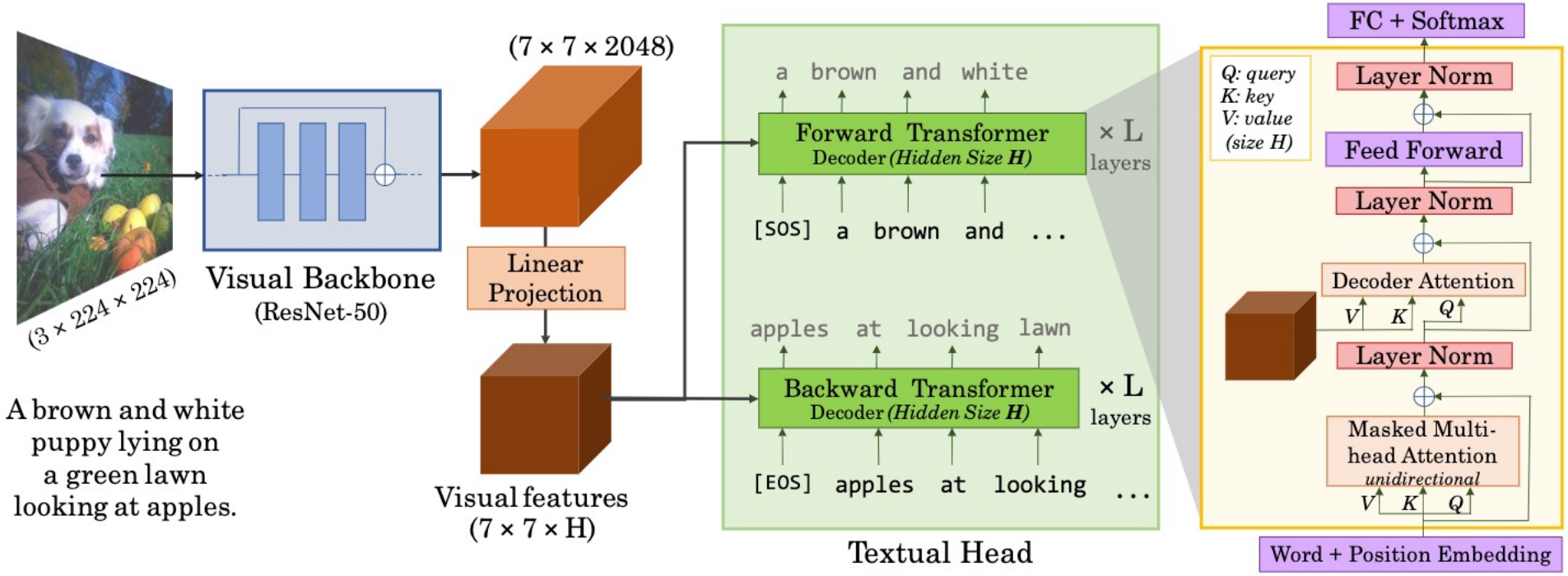
Can language data be used to improve vision models?



VirTex: Learning Visual Representations from Textual Annotations, Desai and Johnson, 2020

VirTex (Desai and Johnson, 2020)

- Pre-train to caption on COCO captions (118K images, 5 captions each)

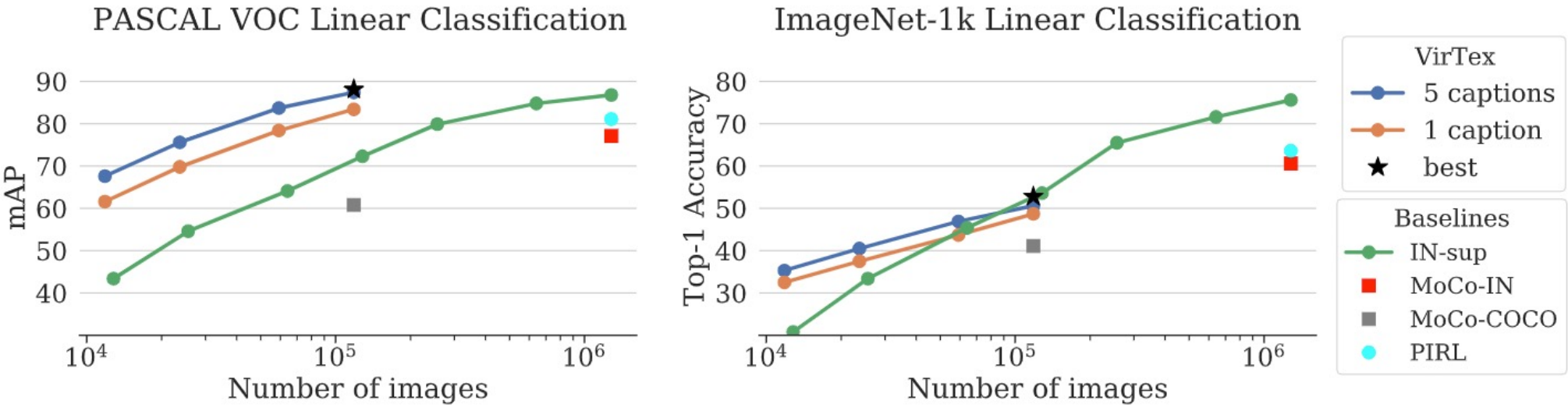


- Use visual features on downstream vision tasks

VirTex: Learning Visual Representations from Textual Annotations, Desai and Johnson, 2020

VirTex (Desai and Johnson, 2020)

- Classification performance (with ResNet-50 backbone)
  - Compare against training on ImageNet (1.28M images total)
  - Methods for self-supervised pre-training with images only
    - MoCo (predict visual features from context), PIRL (contrastive predictions)



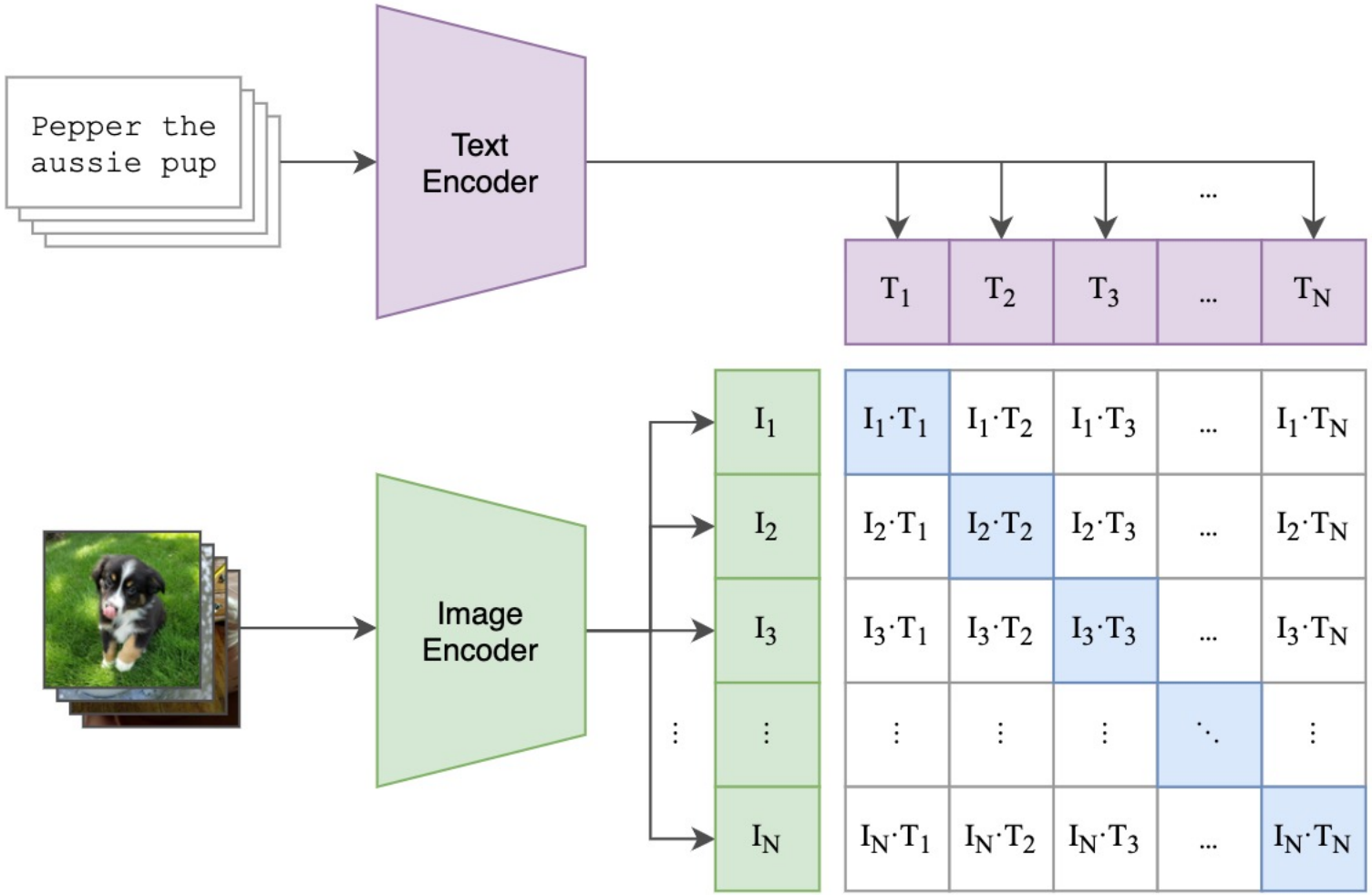
- Better performance for the same number of images

VirTex: Learning Visual Representations from Textual Annotations, Desai and Johnson, 2020

Contrastive pretraining

CLIP (Contrastive Language-Image Pre-Training)

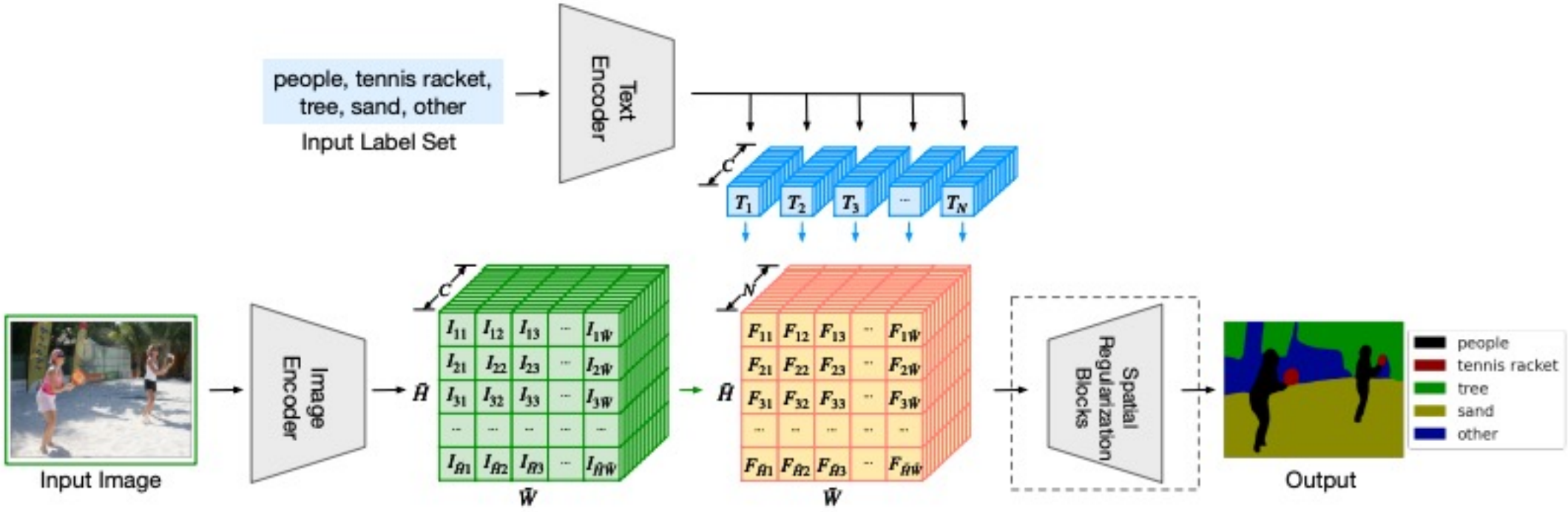
Does the image and text pair match?



Learning Transferable Visual Models From Natural Language Supervision, Radford et al, 2021

Language Driven Semantic Segmentation

Use CLIP as Text Encoder



Use dense prediction transformers architecture

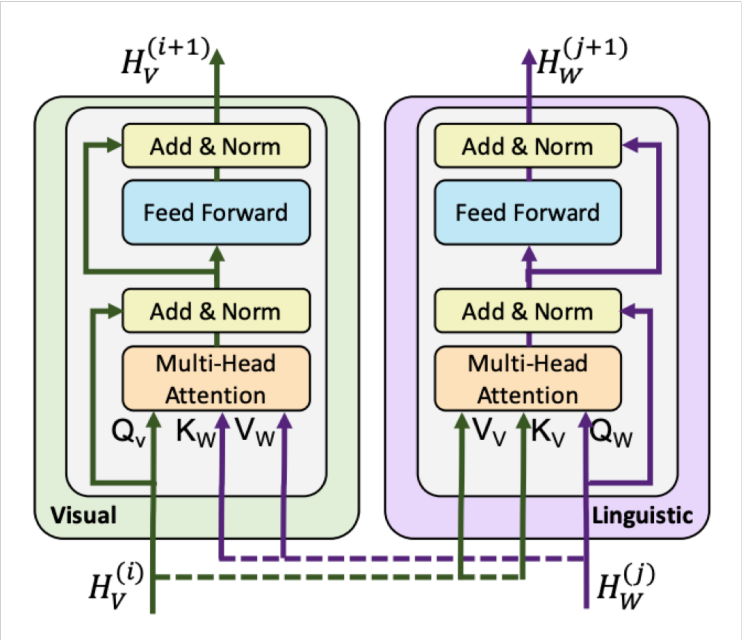
Upsamples to obtain final output

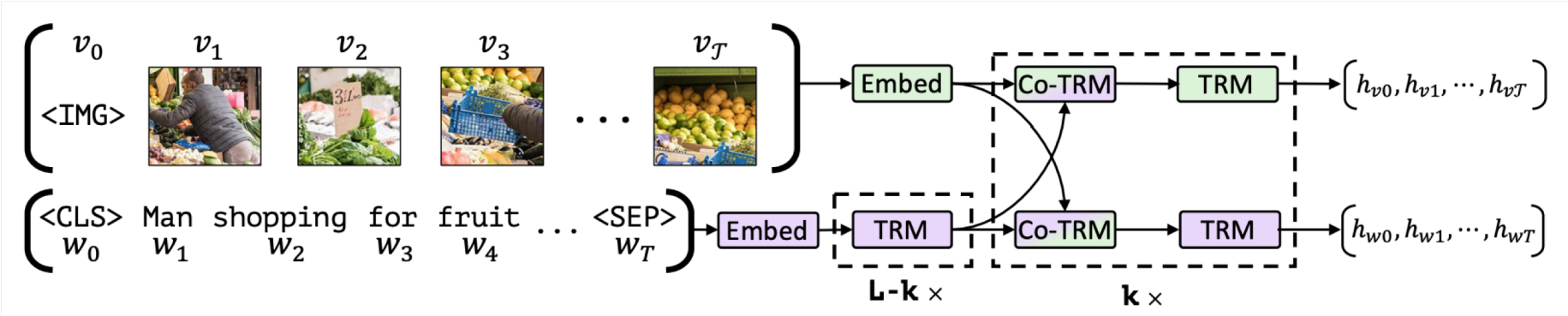
Language Driven Semantic Segmentation, Li et al, ICLR 2022

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





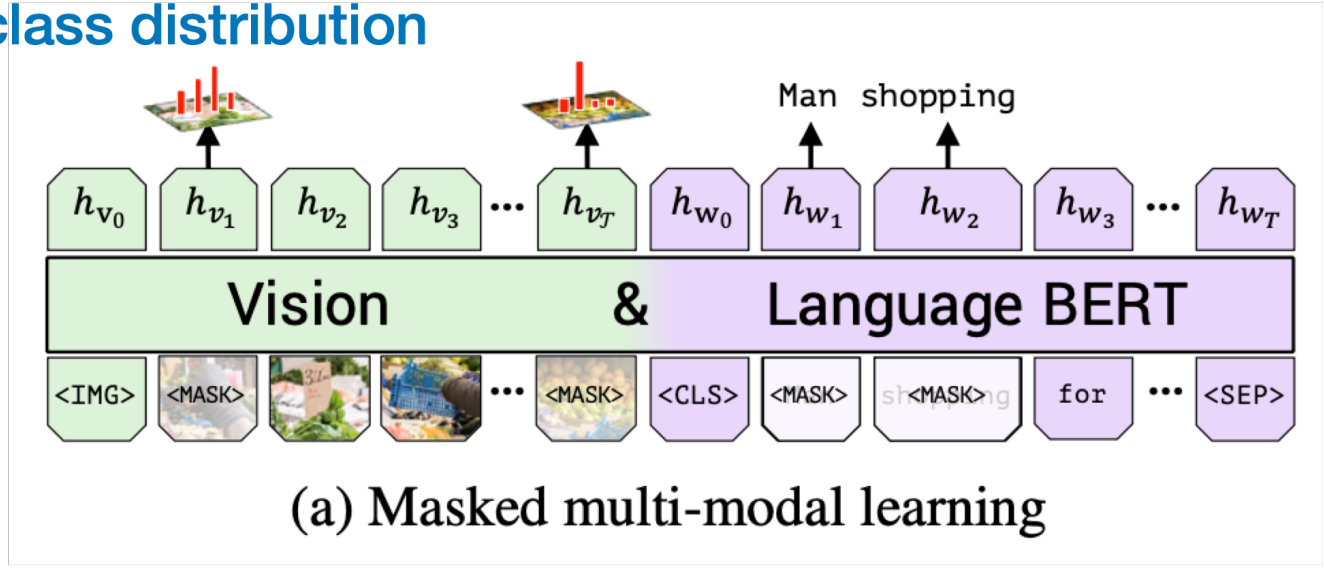
ViLBERT: Pretraining Task-Agnostic Visiolinguistic Representations for Vision-and-Language Tasks
[Lu et al 2019, https://arxiv.org/pdf/1908.02265.pdf]

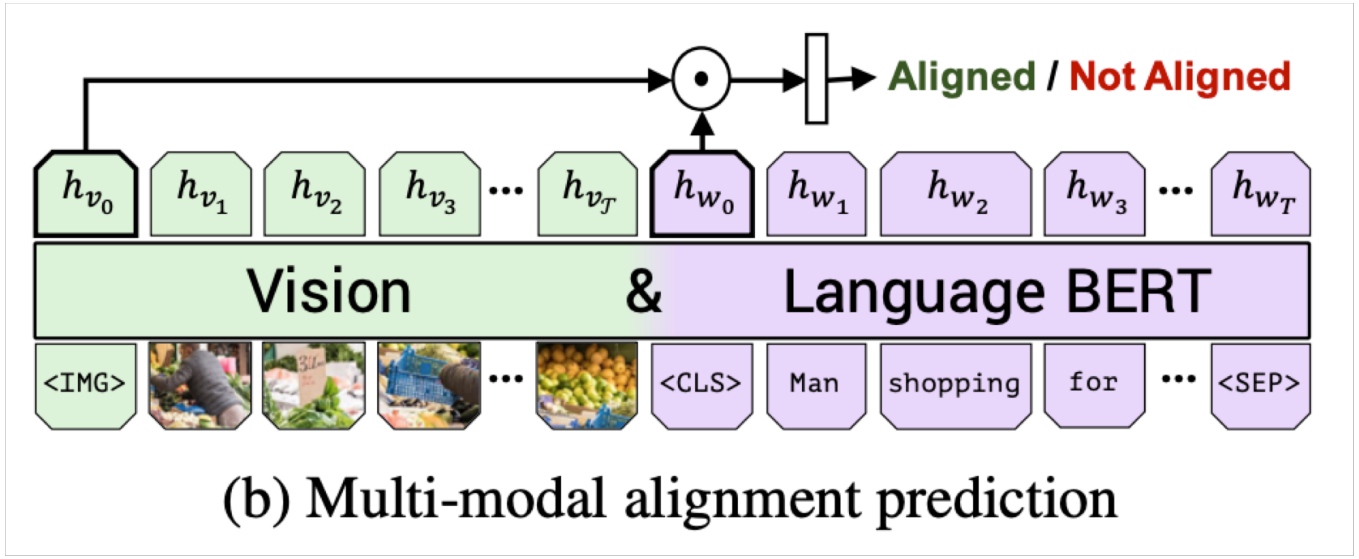
Pretrained representations for vision and language

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?





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

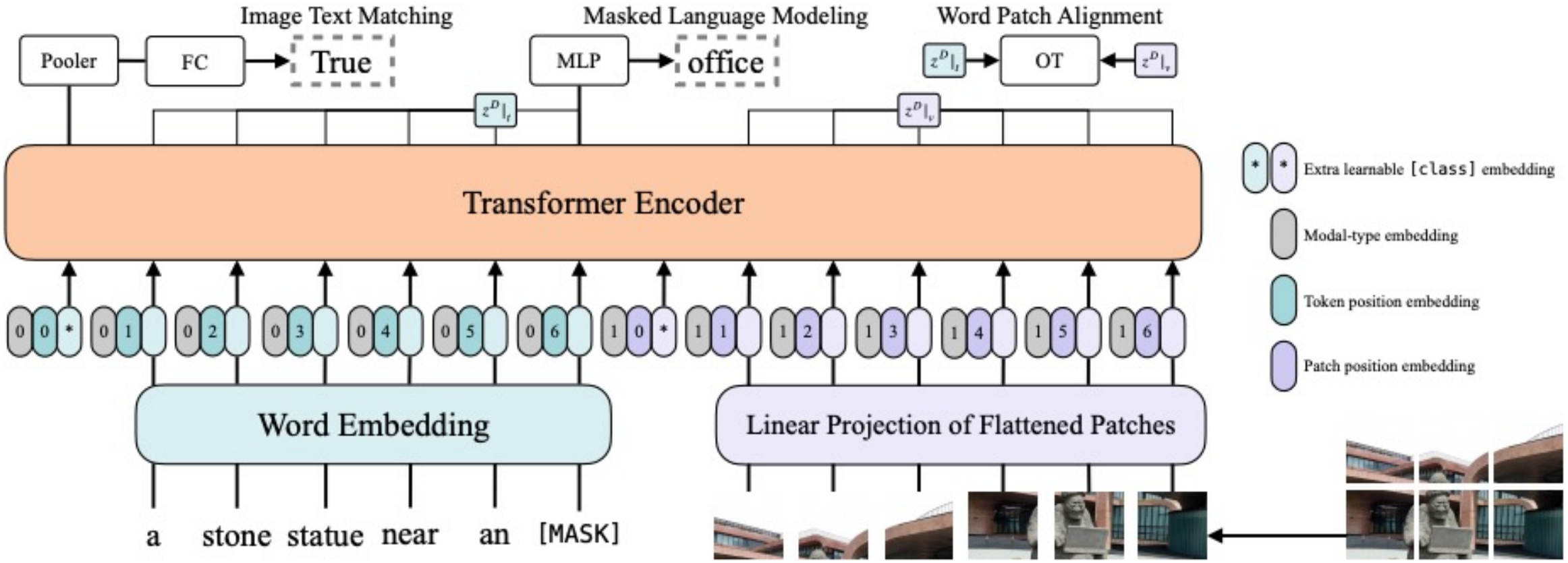
		VQA [3]	VCR [25]			RefCOCO+ [32]			Image Retrieval [26]			ZS Image Retrieval		
	Method	test-dev (test-std)	Q→A	QA→R	Q→AR	val	testA	testB	R1	R5	R10	R1	R5	R10
SOTA	DFAF [36]	70.22 (70.34)	-	-	-	-	-	-	-	-	-	-	-	-
SOTA	R2C [25]	-	63.8 (65.1)	67.2 (67.3)	43.1 (44.0)	-	-	-	-	-	-	-	-	-
SOTA	MAttNet [33]	-	-	-	-	65.33	71.62	56.02	-	-	-	-	-	-
SOTA	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	-	-	-	-	-	-
Ours	Single-Stream	68.85	71.09	73.93	52.73	69.21	75.32	61.02	-	-	-	-	-	-
Ours	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
Ours	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]

Pretrained representation for vision and language

- Use image patches, no need for object detectors



ViLT: Vision-and-Language Transformer Without Convolution or Region Supervision, Kim et al, ICML 2021

Pretrained representation for vision and language

Table 4. Comparison of ViLT-B/32 with other models on downstream retrieval tasks. We use SCAN for w/o VLP SOTA results. † additionally used GQA, VQAv2, VG-QA for pre-training. ‡ additionally used the Open Images dataset. ⓐ indicates RandAugment is applied during fine-tuning. ⊕ indicates model trained for a longer 200K pre-training steps.

Visual Embed	Model	Time (ms)	Text Retrieval						Image Retrieval					
			Flickr30k (1K)			MSCOCO (5K)			Flickr30k (1K)			MSCOCO (5K)		
			R@1	R@5	R@10	R@1	R@5	R@10	R@1	R@5	R@10	R@1	R@5	R@10
Region	w/o VLP SOTA	~900	67.4	90.3	95.8	50.4	82.2	90.0	48.6	77.7	85.2	38.6	69.3	80.4
	ViLBERT-Base	~920	-	-	-	-	-	-	58.2	84.9	91.5	-	-	-
	Unicoder-VL	~925	86.2	96.3	99.0	62.3	87.1	92.8	71.5	91.2	95.2	48.4	76.7	85.9
	UNITER-Base	~900	85.9	97.1	98.8	64.4	87.4	93.1	72.5	92.4	96.1	50.3	78.5	87.2
	OSCAR-Base†	~900	-	-	-	70.0	91.1	95.5	-	-	-	54.0	80.8	88.5
	VinVL-Base†‡	~650	-	-	-	74.6	92.6	96.3	-	-	-	58.1	83.2	90.1
Grid	Pixel-BERT-X152	~160	87.0	98.9	99.5	63.6	87.5	93.6	71.5	92.1	95.8	50.1	77.6	86.2
	Pixel-BERT-R50	~60	75.7	94.7	97.1	59.8	85.5	91.6	53.4	80.4	88.5	41.1	69.7	80.5
Linear	ViLT-B/32	~15	81.4	95.6	97.6	61.8	86.2	92.6	61.9	86.8	92.8	41.3	72.0	82.5
	ViLT-B/32ⓐ	~15	83.7	97.2	98.1	62.9	87.1	92.7	62.2	87.6	93.2	42.6	72.8	83.4
	ViLT-B/32ⓐ⊕	~15	83.5	96.7	98.6	61.5	86.3	92.7	64.4	88.7	93.8	42.7	72.9	83.1

ViLT: Vision-and-Language Transformer Without Convolution or Region Supervision, Kim et al, ICML 2021

Pretrained representation for vision and language

a display of flowers growing out and over the retaining wall in front of cottages on a cloudy day.

a room with a rug, a chair, a painting, and a plant.

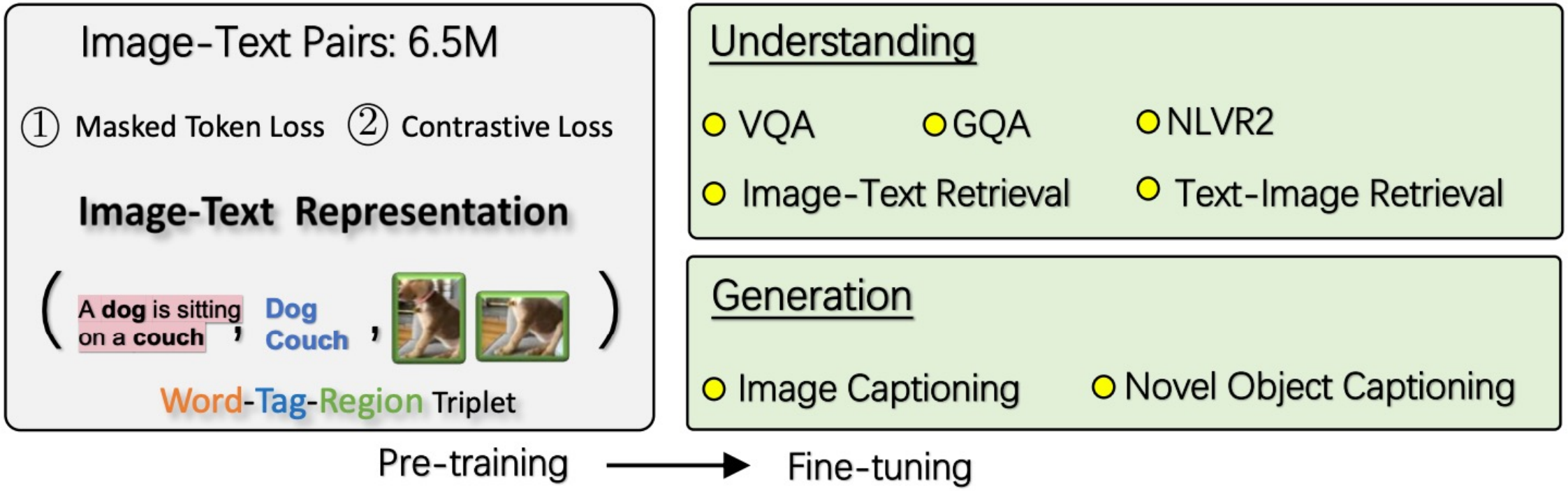
ViLT: Vision-and-Language Transformer Without Convolution or Region Supervision, Kim et al, ICML 2021

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)

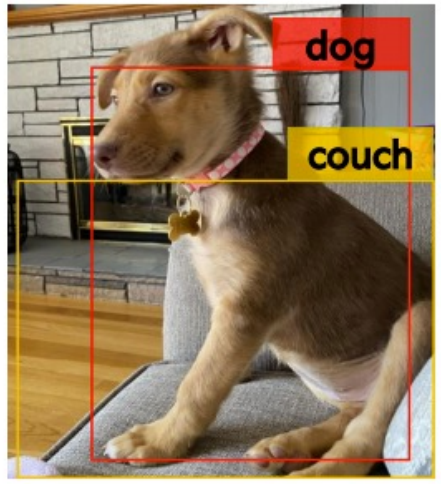
- Fine tuned on 7 tasks



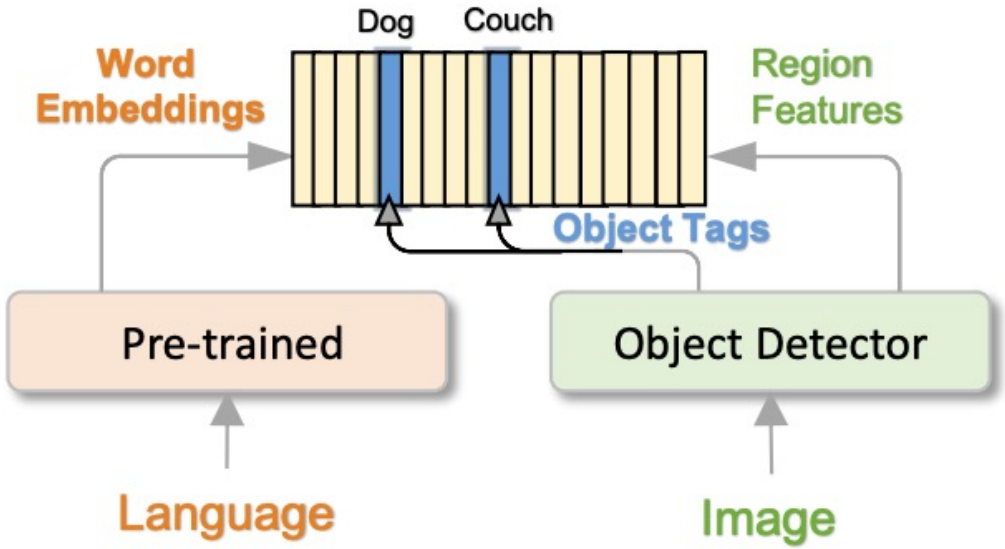
Oscar: Object-Semantics Aligned Pre-training for Vision-Language Tasks, Li et al, ECCV 2020

Oscar: Pre-training with object-semantic alignment

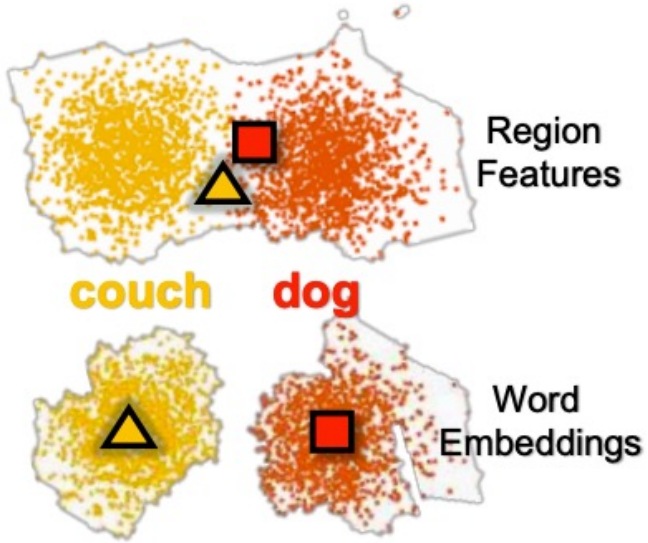
- Use detected object tags as anchors



(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

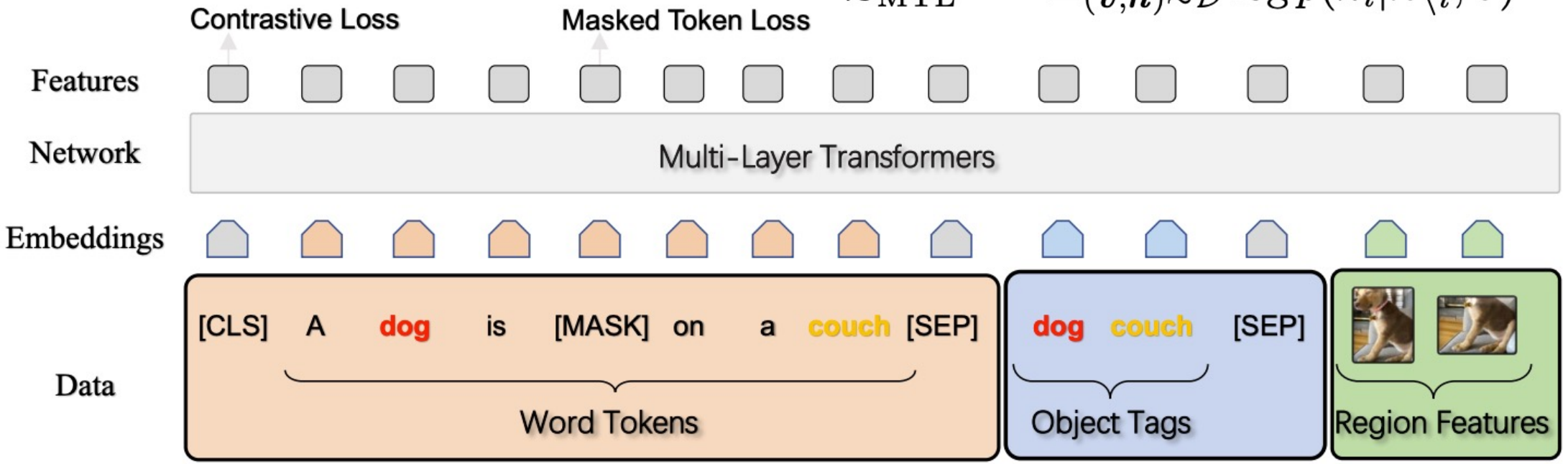
Oscar: Pre-training with object-semantic alignment

Randomly pollute q $\quad h' \triangleq [q, v]$

Randomly mask token h_i $\quad h \triangleq [w, q]$

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

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



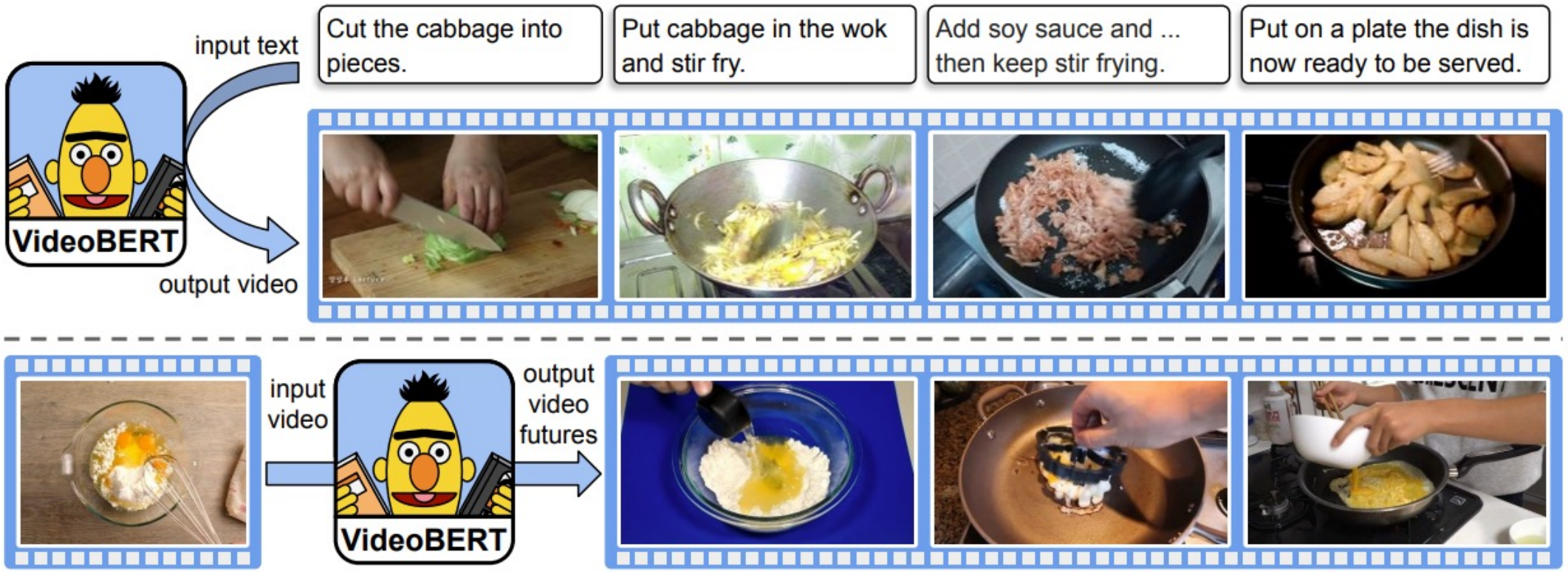
w = text embedding

q = object tags embedding

v = visual features

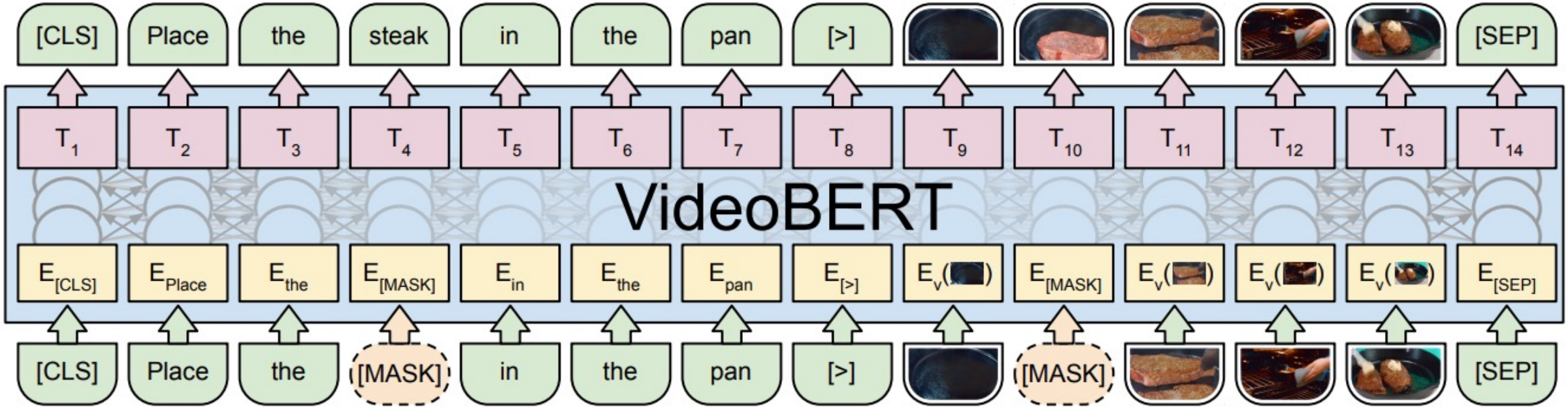
Oscar: Object-Semantics Aligned Pre-training for Vision-Language Tasks, Li et al, ECCV 2020

Pretraining for video and language



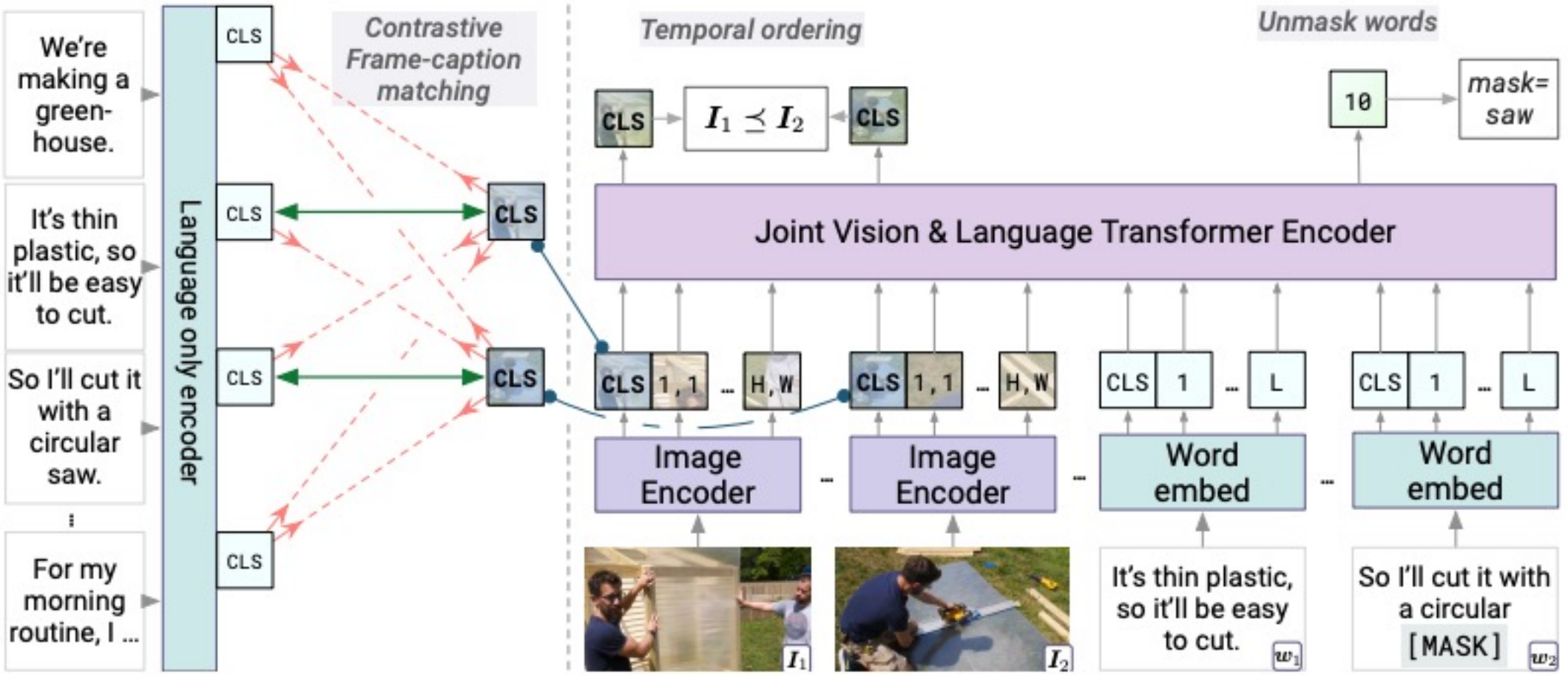
VideoBERT: A Joint Model for Video and Language Representation Learning, Sun et al, ICCV 2019

Pretraining for video and language



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

Pretraining for video and language



MERLOT: Multimodal Neural Script Knowledge Models, Zeller et al, NeurIPS 2021

Next week

- Monday: Paper presentations and discussions
  - ViLBERT (Tristan)
  - Merlot (Dave)
- Wednesday: Text conditioned content generation