COSE474 Deep Learning

Lecture 14: Attention, Transformers, and Vision Transformers

Seungryong Kim

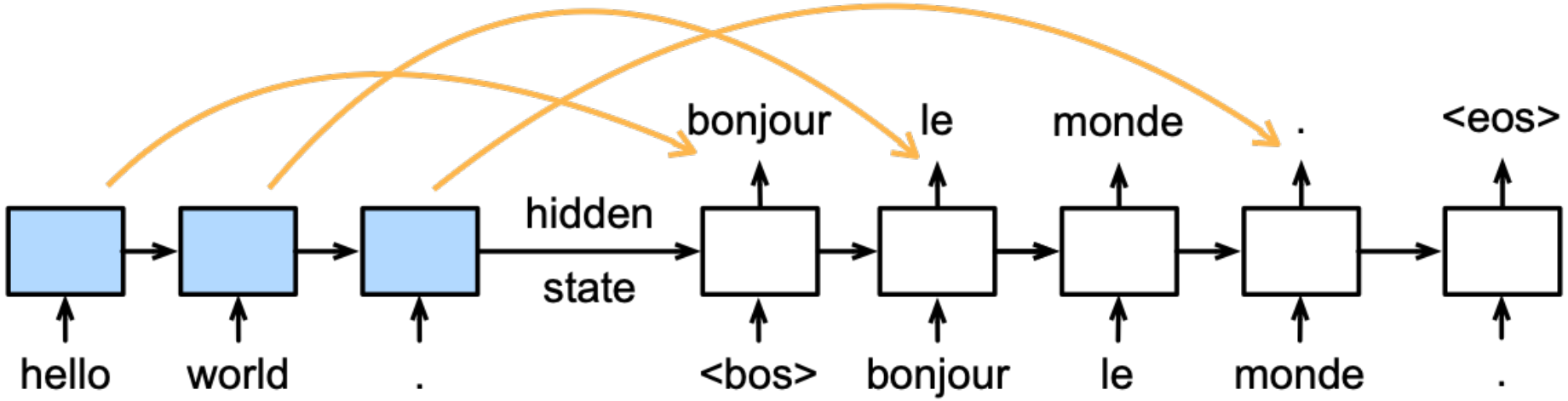
Computer Vision Lab. (CVLAB)

Department of Computer Science and Engineering

Korea University

Attention is All You Need!

- Each generated token might be related to different source tokens

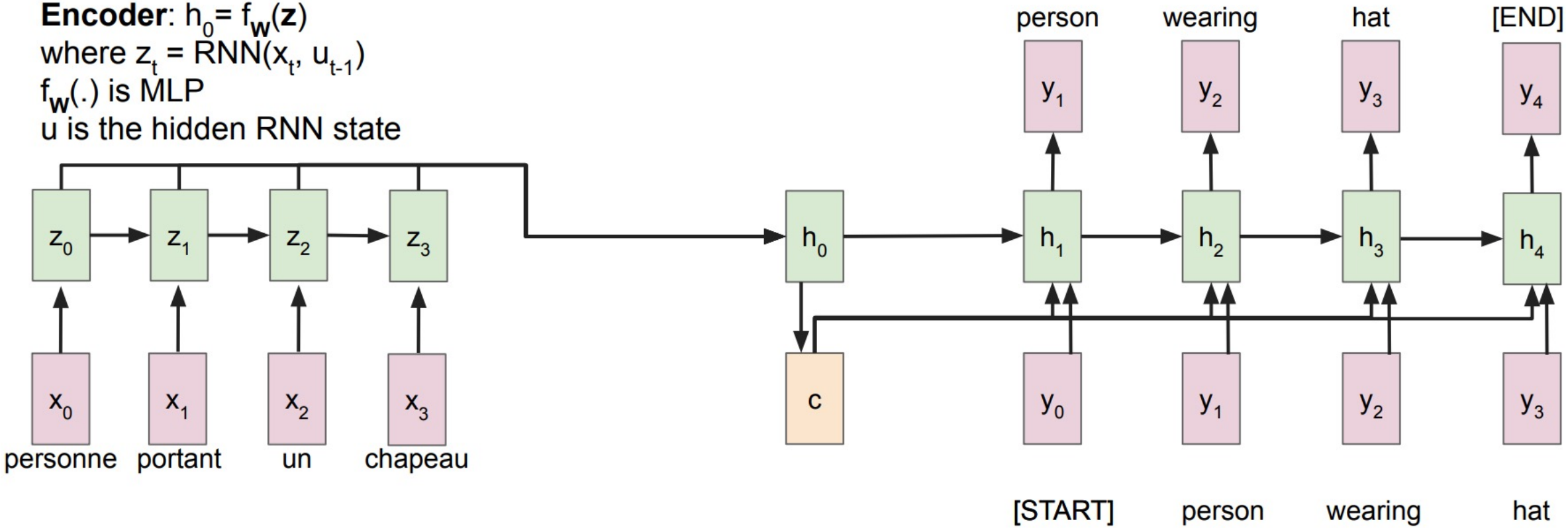


Language Translation Example

Input: Sequence $\mathbf{x} = x_1, x_2, \ldots, x_T$
Output: Sequence $\mathbf{y} = y_1, y_2, \ldots, y_T$

Decoder: $y_t = g_{\mathbf{V}}(y_{t-1}, h_{t-1}, c)$
where context vector c is often $c = h_0$

Encoder: $h_0 = f_{\mathbf{W}}(\mathbf{z})$
where $z_t = \text{RNN}(x_t, u_{t-1})$
$f_{\mathbf{W}}(.)$ is MLP
u is the hidden RNN state

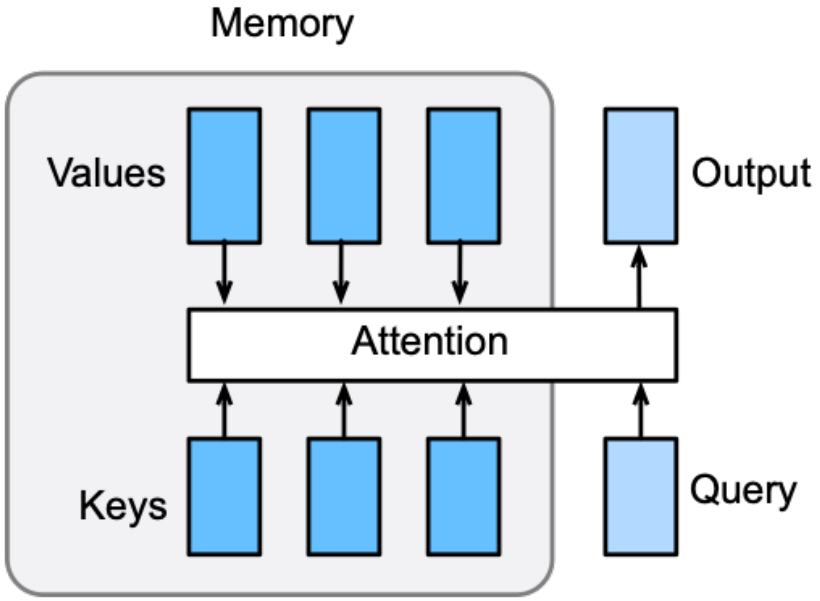


courses.d2l.ai/berkeley-stat-157/index.html

Attention in NLP

Attention Layer

- An attention layer explicitly select related information
  - Its memory consists of key-value pairs
  - The output will be close to the values whose keys are similar to the query



Attention in NLP

Attention Layer

- Assume query $\mathbf{q} \in \mathbb{R}^{d_q}$ and the memory $(\mathbf{k}_1, \mathbf{v}_1), \ldots, (\mathbf{k}_n, \mathbf{v}_n)$ with $\mathbf{k}_i \in \mathbb{R}^{d_k}$ $\mathbf{v}_i \in \mathbb{R}^{d_v}$
- Compute n scores $a_1, \ldots, a_n$ with $a_i = \alpha(\mathbf{q}, \mathbf{k}_i)$
- Use softmax to obtain the attention weights

$$b_1, \ldots, b_n = \text{softmax}(a_1, \ldots, a_n)$$

- The output is a weighted sum of values

$$\mathbf{o} = \sum_{i=1}^{n} b_i \mathbf{v}_i$$

Vary α to get different attention layers

Dot Product Attention

- Assume the query has the same length as values $\mathbf{q}, \mathbf{k}_i \in \mathbb{R}^d$

$$\alpha(\mathbf{q}, \mathbf{k}) = \langle \mathbf{q}, \mathbf{k} \rangle / \sqrt{d}$$

- Vectorized version
  - *m* queries $\mathbf{Q} \in \mathbb{R}^{m \times d}$ and *n* keys $\mathbf{K} \in \mathbb{R}^{n \times d}$

$$\alpha(\mathbf{Q}, \mathbf{K}) = \mathbf{Q}\mathbf{K}^T / \sqrt{d}$$

Multilayer Perception Attention

- Learnable parameters $\mathbf{W}_k \in \mathbb{R}^{h \times d_k}, \mathbf{W}_q \in \mathbb{R}^{h \times d_q}$, and $\mathbf{v} \in \mathbb{R}^h$

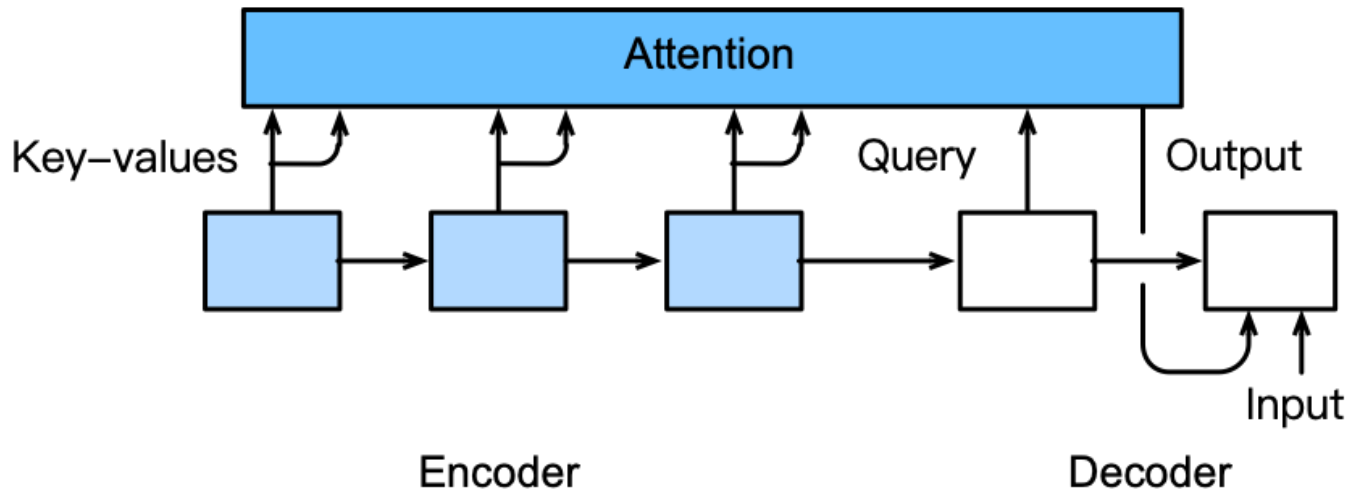
$$\alpha(\mathbf{k}, \mathbf{q}) = \mathbf{v}^T \tanh(\mathbf{W}_k \mathbf{k} + \mathbf{W}_q \mathbf{q})$$

- Equals to concatenate the key and query, and then feed into a single hidden-layer perception with hidden size h and output size 1

Seq2Seq with Attention

Model Architecture

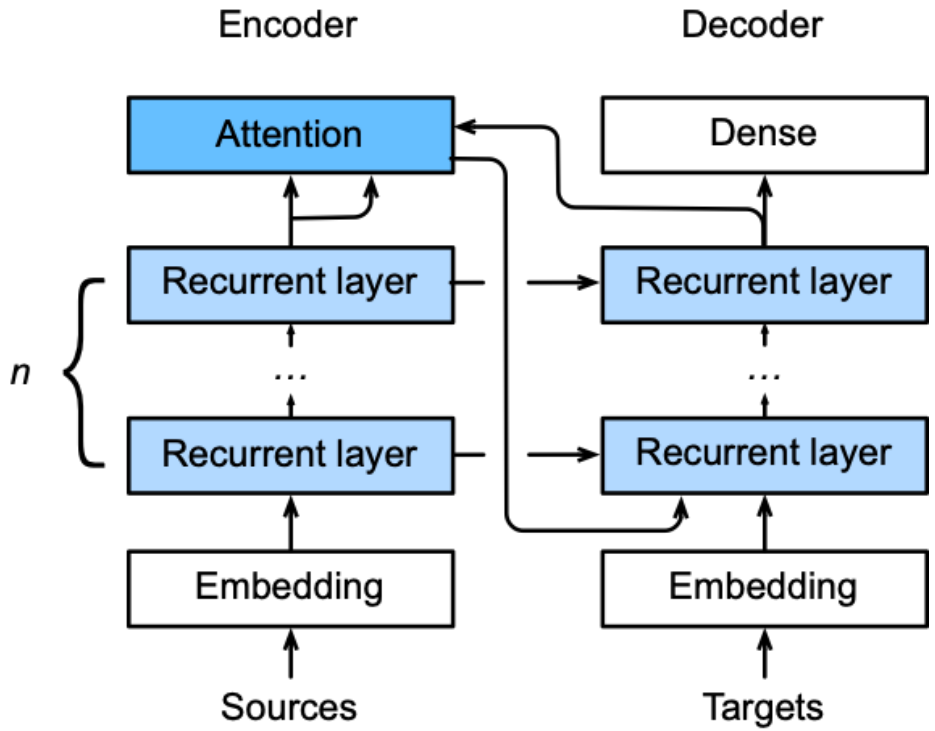
- Add an additional attention layer to use encoder's outputs as memory
- The attention output is used as the decoder's input



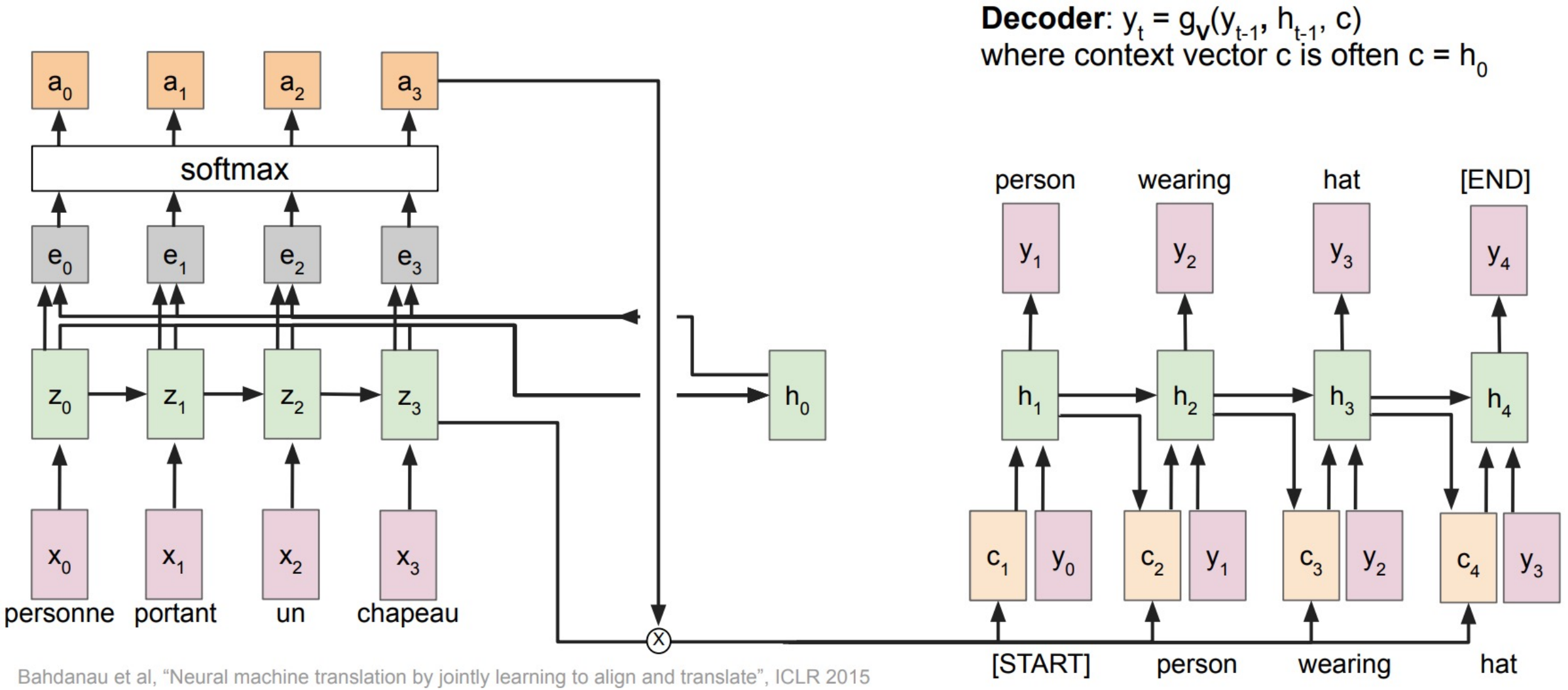
Seq2Seq with Attention

Encoder/Decoder Details

- The output of the last recurrent layer in the encoder is used
- The attention output is then concatenated with the embedding output to feed into the first recurrent layer in the decoder



Seq2Seq with Attention



Bahdanau et al, "Neural machine translation by jointly learning to align and translate", ICLR 2015

courses.d2l.ai/berkeley-stat-157/index.html

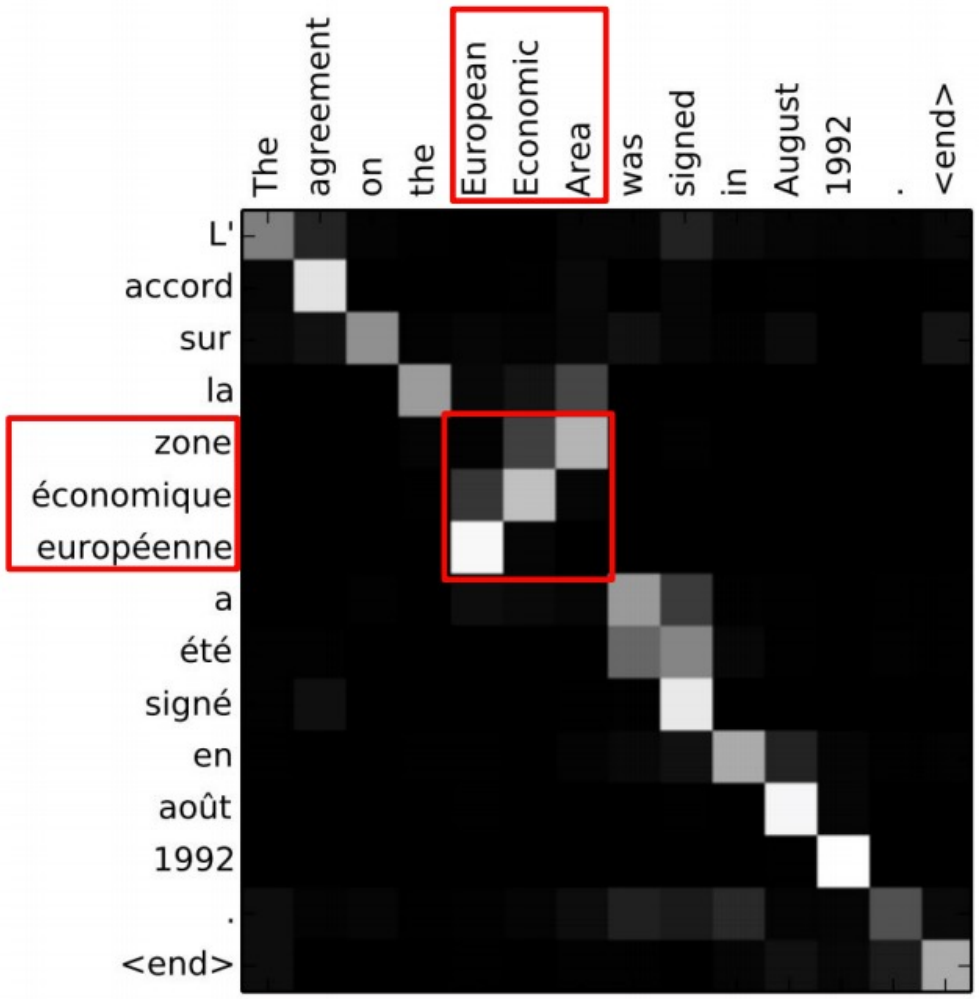
Seq2Seq with Attention

English to French translation example:

Input: "The agreement on the European Economic Area was signed in August 1992."

Output: "L'accord sur la zone économique européenne a été signé en août 1992."

Without any attention supervision, model learns different word orderings for different languages



courses.d2l.ai/berkeley-stat-157/index.html

Visual Captioning

Describe the content of an *image* or *video* with *natural language sentence*.

A cat is sitting next to a pine tree, looking up.

A dog is playing piano with a girl.

Applications of Visual Captioning

- Alt-text generation (from PowerPoint)
- Content-based image retrieval (CBIR)
- Or just for fun!

Image Captioning with CNN-LSTM

- Problem Formulation

$$\theta^\star = \arg\max_\theta \sum_{(I,S)} \log p(S|I;\theta)$$

$$\log p(S|I) = \sum_{t=0}^{N} \log p(S_t|I, S_0, \ldots, S_{t-1})$$

- The Encoder-Decoder framework

Visual Encoder → Language Decoder → "Cat sitting outside"

"Show and Tell"

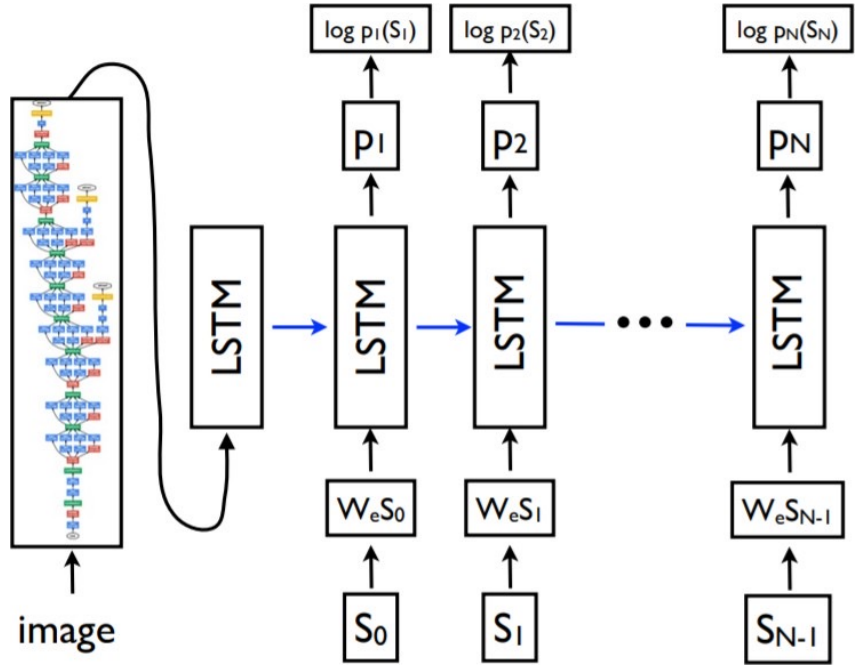
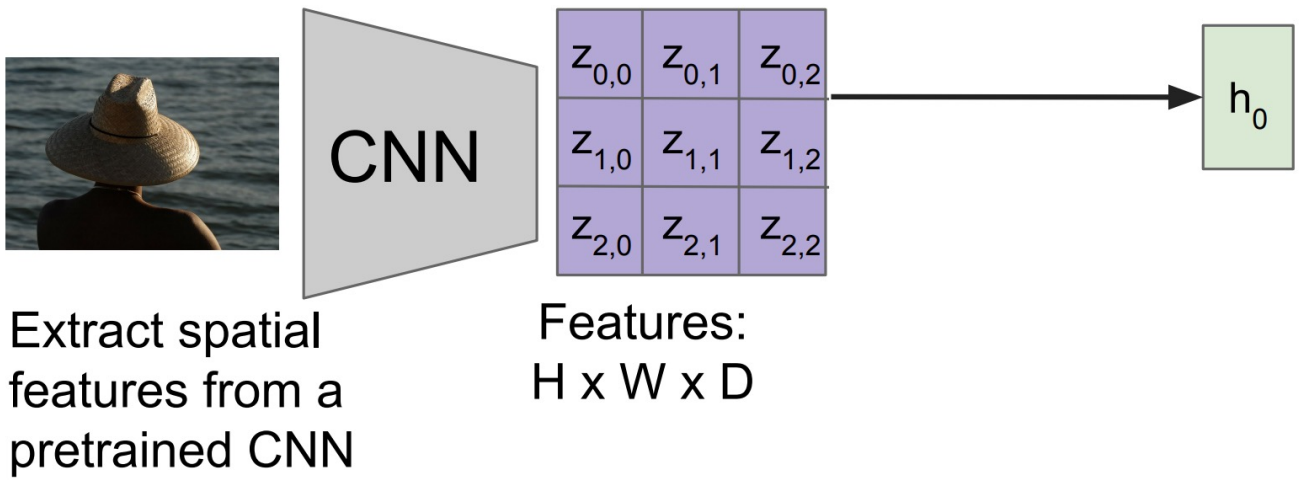


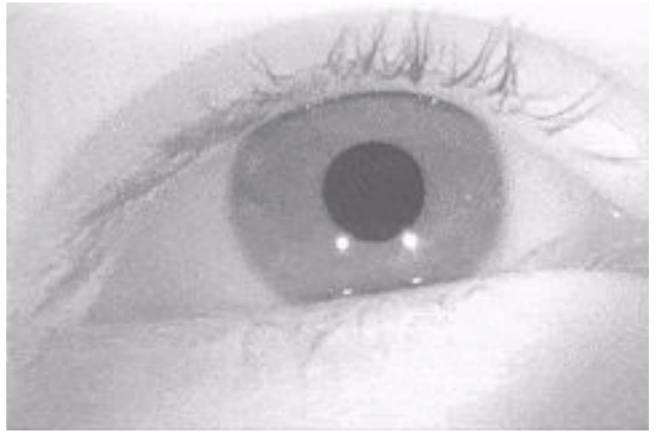
Image credit: Vinyals et al. "Show and Tell: A Neural Image Caption Generator", CVPR 2015.

Image Captioning with RNNs & Attention

Attention idea: New context vector at every time step.

Each context vector will attend to different image regions





Attention Saccades in humans

Xu et al, “Show, Attend and Tell: Neural Image Caption Generation with Visual Attention”, ICML 2015

Image Captioning with Soft Attention

- Soft Attention – Dynamically attend to input content based on query.
- Basic elements: query – q, keys - K, and values – V
- In this case, keys and values are usually identical. They come from the CNN activation map.
- Query q is determined by the global image feature or LSTM's hidden states.

Bahdanau et al. "Neural Machine Translation by Jointly Learning to Align and Translate", ICLR 2015.
Xu et al. "Show, Attend and Tell", ICML 2015.

Image Captioning with Soft Attention

$e_{t,i,j} = f_{att}(s_{t-1}, h_{i,j})$

$a_{t,:,:} = \text{softmax}(e_{t,:,:})$

$c_t = \sum_{i,j} a_{t,i,j} h_{i,j}$

CNN

Use a CNN to compute a grid of features for an image

Alignment scores

$e_{1,1,1}$ $e_{1,1,2}$ $e_{1,1,3}$
$e_{1,2,1}$ $e_{1,2,2}$ $e_{1,2,3}$
$e_{1,3,1}$ $e_{1,3,2}$ $e_{1,3,3}$

softmax

Attention weights

$a_{1,1,1}$ $a_{1,1,2}$ $a_{1,1,3}$
$a_{1,2,1}$ $a_{1,2,2}$ $a_{1,2,3}$
$a_{1,3,1}$ $a_{1,3,2}$ $a_{1,3,3}$

$h_{1,1}$ $h_{1,2}$ $h_{1,3}$
$h_{2,1}$ $h_{2,2}$ $h_{2,3}$
$h_{3,1}$ $h_{3,2}$ $h_{3,3}$

s_0 s_1

y_1 cat

c_1 y_0

[START]

Image Captioning with Soft Attention

$$e_{t,i,j} = f_{att}(s_{t-1}, h_{i,j})$$
$$a_{t,:,:} = \text{softmax}(e_{t,:,:})$$
$$c_t = \sum_{i,j} a_{t,i,j} h_{i,j}$$

CNN

$h_{1,1}$	$h_{1,2}$	$h_{1,3}$
$h_{2,1}$	$h_{2,2}$	$h_{2,3}$
$h_{3,1}$	$h_{3,2}$	$h_{3,3}$

Use a CNN to compute a grid of features for an image

cat

y_1

s_0

s_1

c_1

y_0

[START]

Image Captioning with Soft Attention

$e_{t,i,j} = f_{att}(s_{t-1}, h_{i,j})$

$a_{t,:,:} = \text{softmax}(e_{t,:,:})$

$c_t = \sum_{i,j} a_{t,i,j} h_{i,j}$

Alignment scores

$e_{2,1,1}$	$e_{2,1,2}$	$e_{2,1,3}$
$e_{2,2,1}$	$e_{2,2,2}$	$e_{2,2,3}$
$e_{2,3,1}$	$e_{2,3,2}$	$e_{2,3,3}$

CNN

$h_{1,1}$	$h_{1,2}$	$h_{1,3}$
$h_{2,1}$	$h_{2,2}$	$h_{2,3}$
$h_{3,1}$	$h_{3,2}$	$h_{3,3}$

Use a CNN to compute a grid of features for an image

cat

y_1

s_0

s_1

c_1

y_0

[START]

[CVPR tutorial]

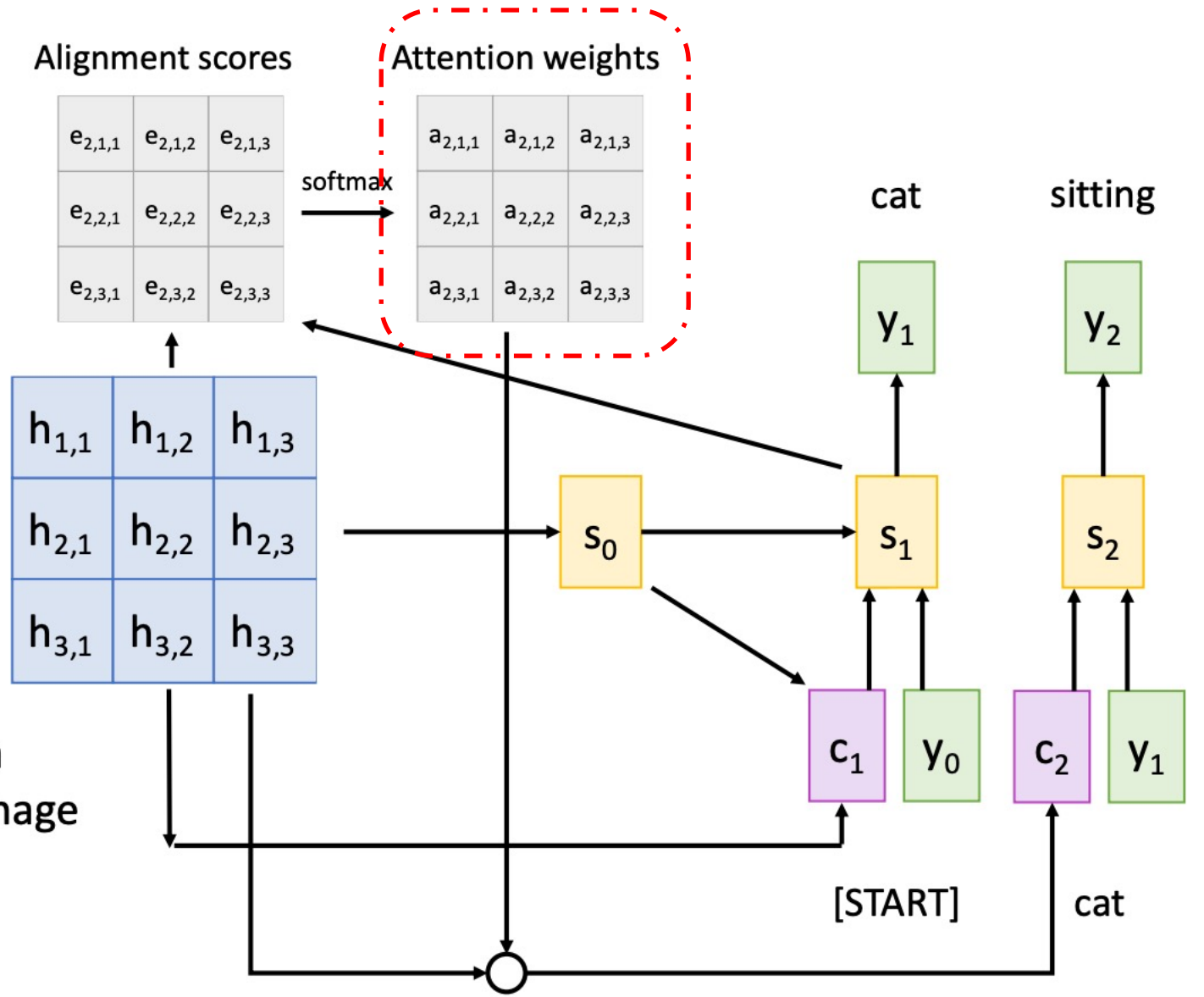
Image Captioning with Soft Attention

$e_{t,i,j} = f_{att}(s_{t-1}, h_{i,j})$

$a_{t,:,:} = \text{softmax}(e_{t,:,:})$

$c_t = \sum_{i,j} a_{t,i,j} h_{i,j}$

Use a CNN to compute a grid of features for an image



Slide credit: UMich EECS 498/598 DeepVision course by Justin Johnson. Method: "Show, Attend and Tell" by Xu et al. ICML 2015.

Image Captioning with Soft Attention

$e_{t,i,j} = f_{att}(s_{t-1}, h_{i,j})$
$a_{t,:,:} = \text{softmax}(e_{t,:,:})$
$c_t = \sum_{i,j} a_{t,i,j} h_{i,j}$

Each timestep of decoder uses a different context vector that looks at different parts of the input image

CNN

$h_{1,1}$ $h_{1,2}$ $h_{1,3}$
$h_{2,1}$ $h_{2,2}$ $h_{2,3}$
$h_{3,1}$ $h_{3,2}$ $h_{3,3}$

Use a CNN to compute a grid of features for an image

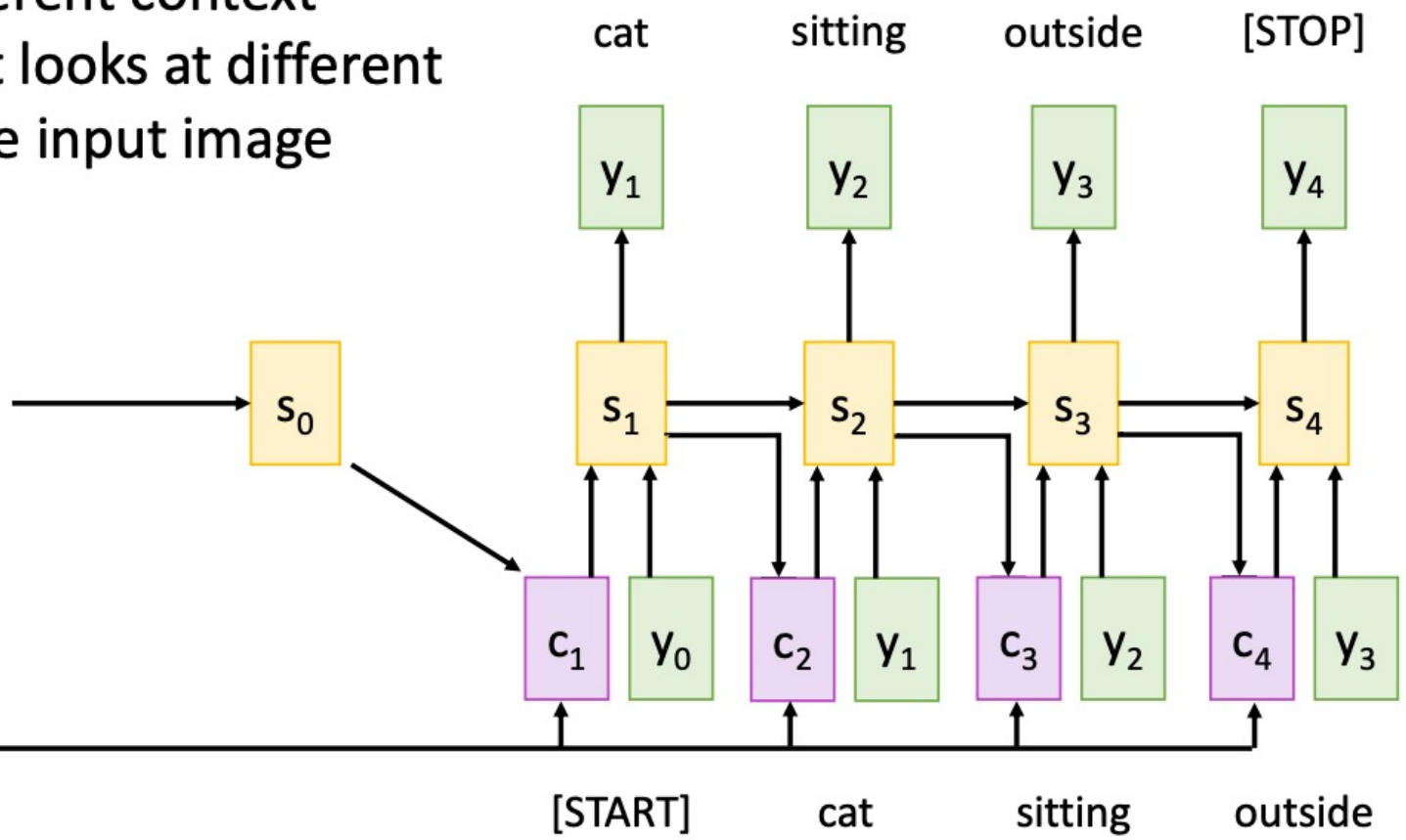
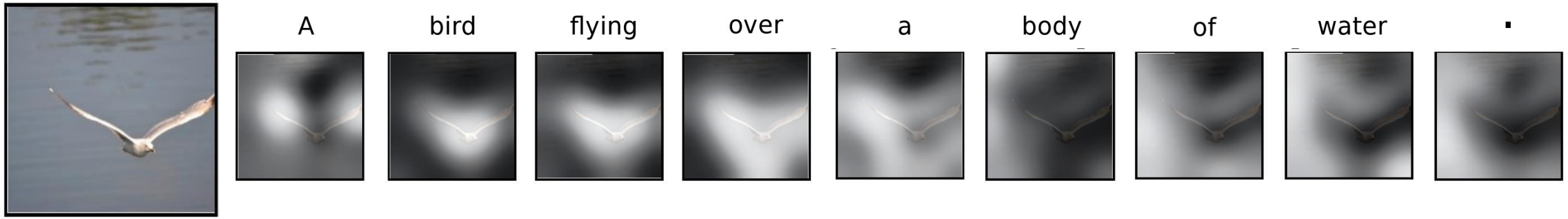


Image Captioning with Soft Attention



[CVPR tutorial]

Image Captioning with Soft Attention

A dog is standing on a hardwood floor.

A stop sign is on a road with a mountain in the background.

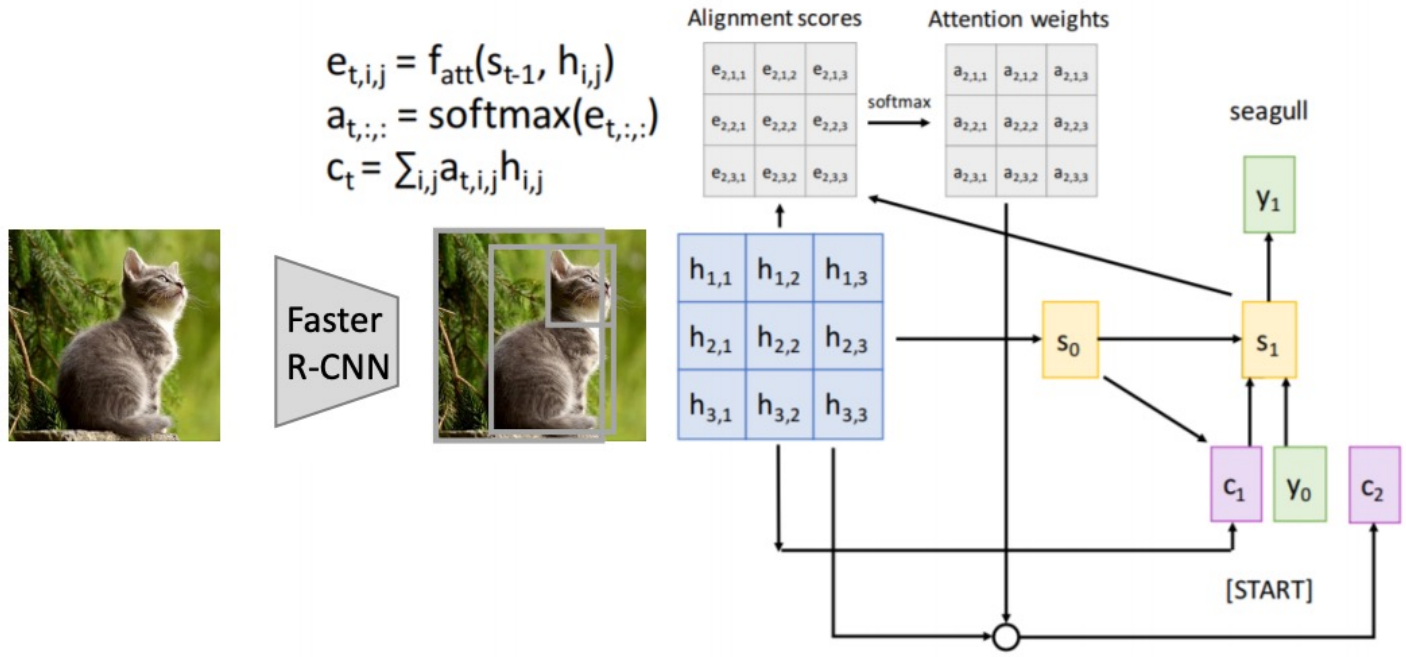
A group of people sitting on a boat in the water.

A giraffe standing in a forest with trees in the background.

[CVPR tutorial]

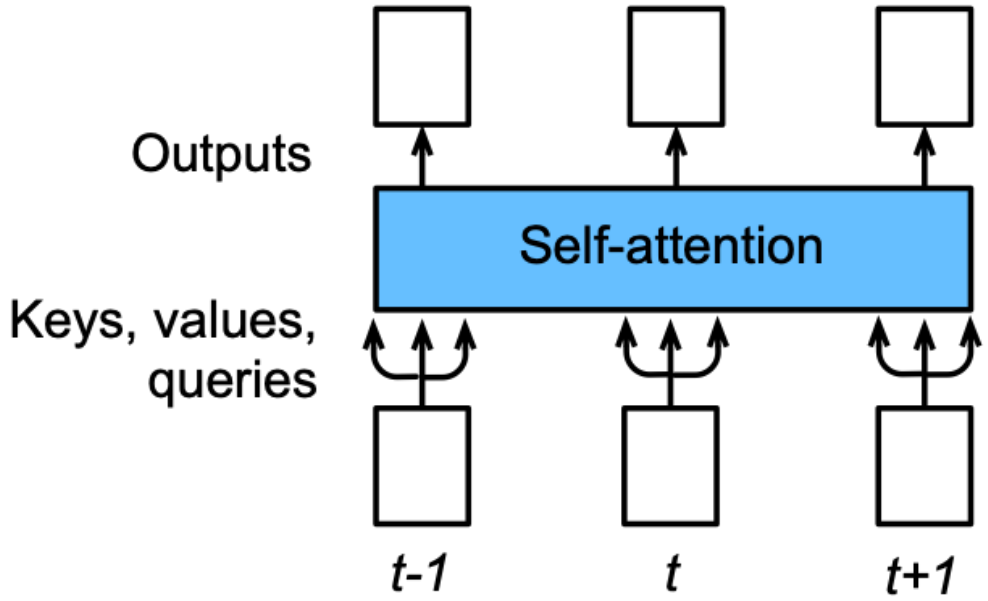
Image Captioning with Soft Attention

- Variants of Soft Attention based on the feature input
  - Grid activation features (covered)
  - Region proposal features



Self-attention in Transformers

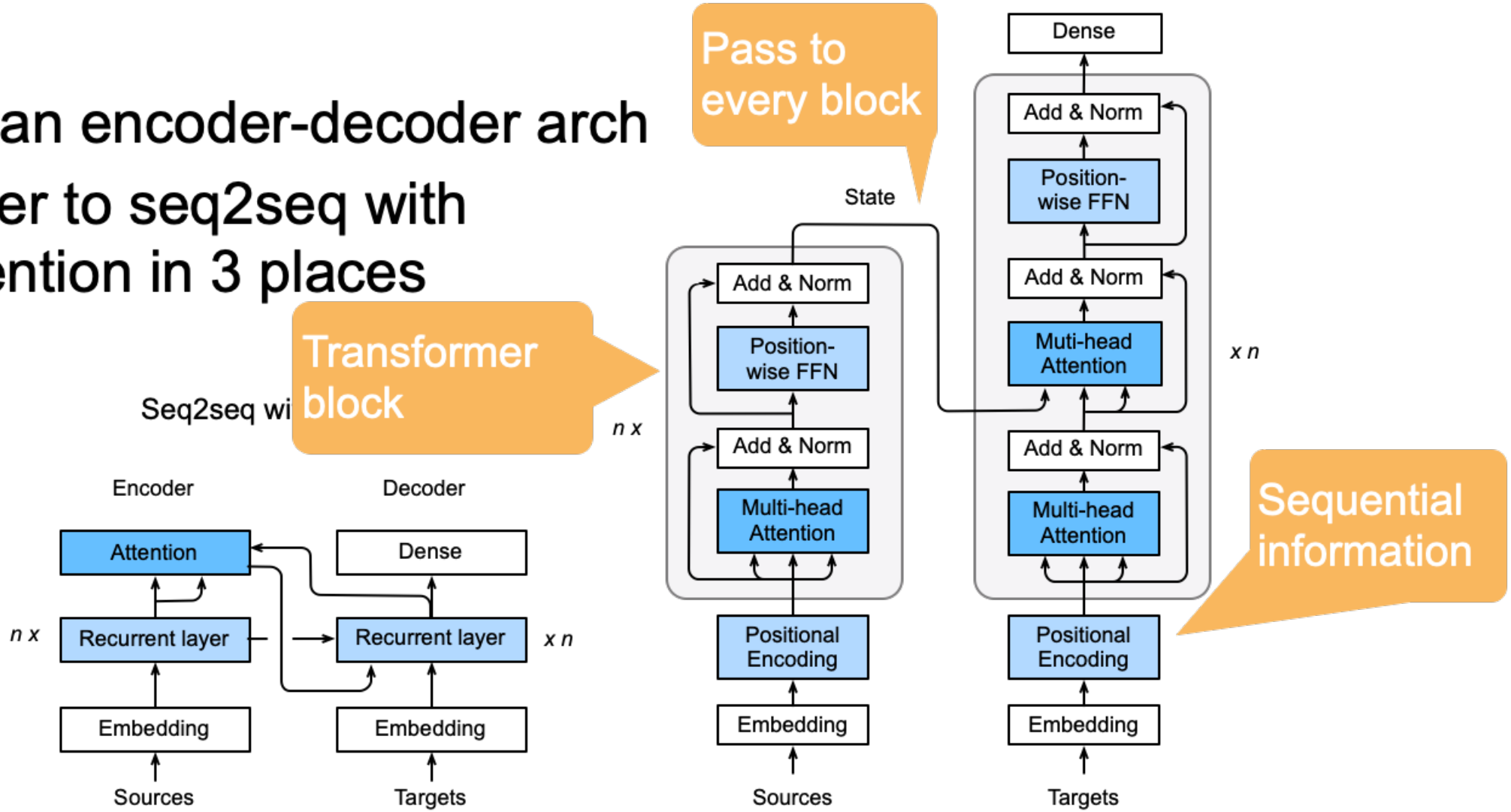
- To generate *n* outputs with *n* inputs, we can copy each input into a key, a value and a query
- No sequential information is preserved
- Run in parallel



Vaswani et al., “Attention is All You Need,” NIPS, 2017.

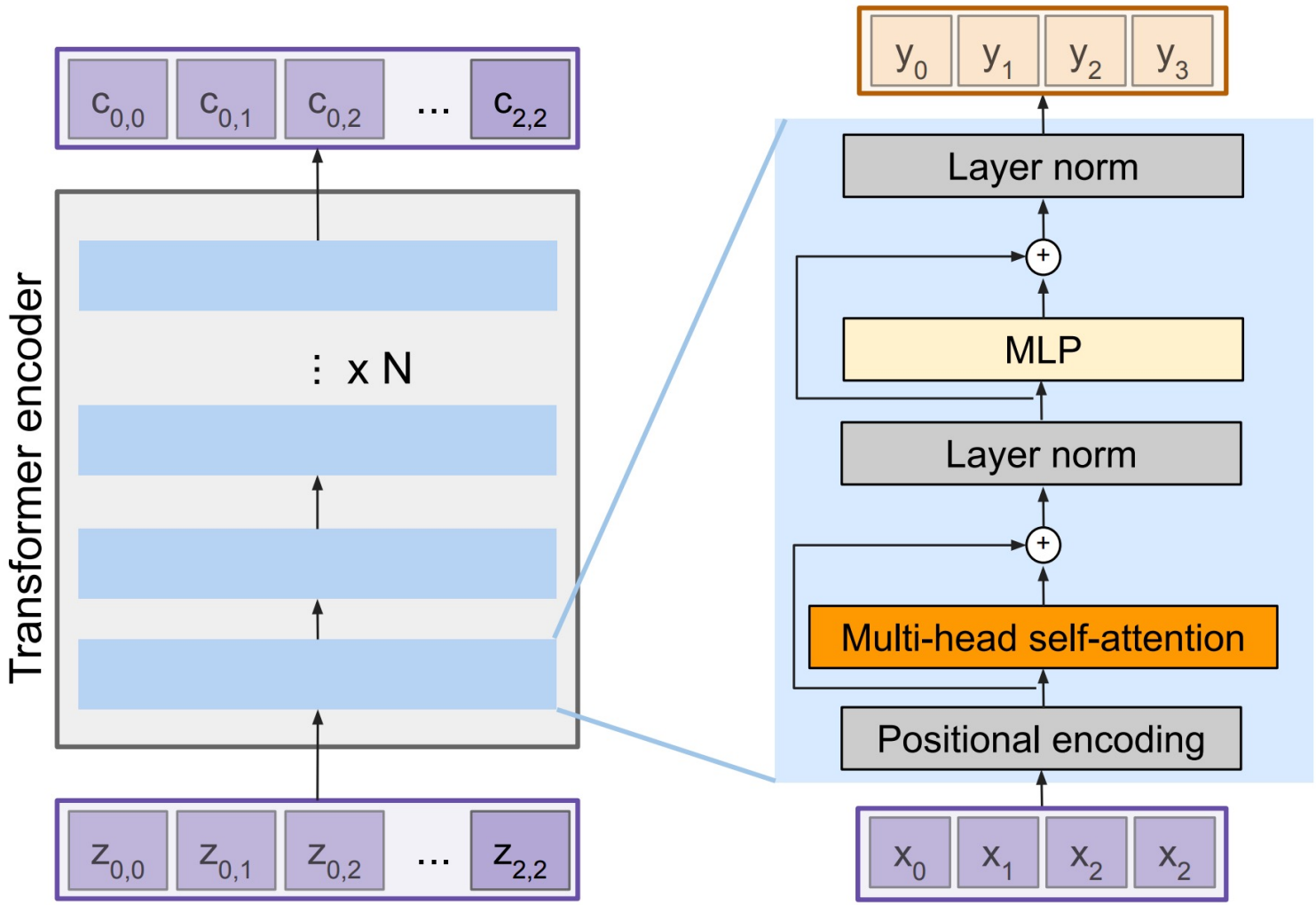
Transformer Architecture

- It's an encoder-decoder arch
- Differ to seq2seq with attention in 3 places



Vaswani et al., "Attention is All You Need," NIPS, 2017.

Transformer Encoder Block



Transformer Encoder Block:

Inputs: Set of vectors **x**
Outputs: Set of vectors **y**

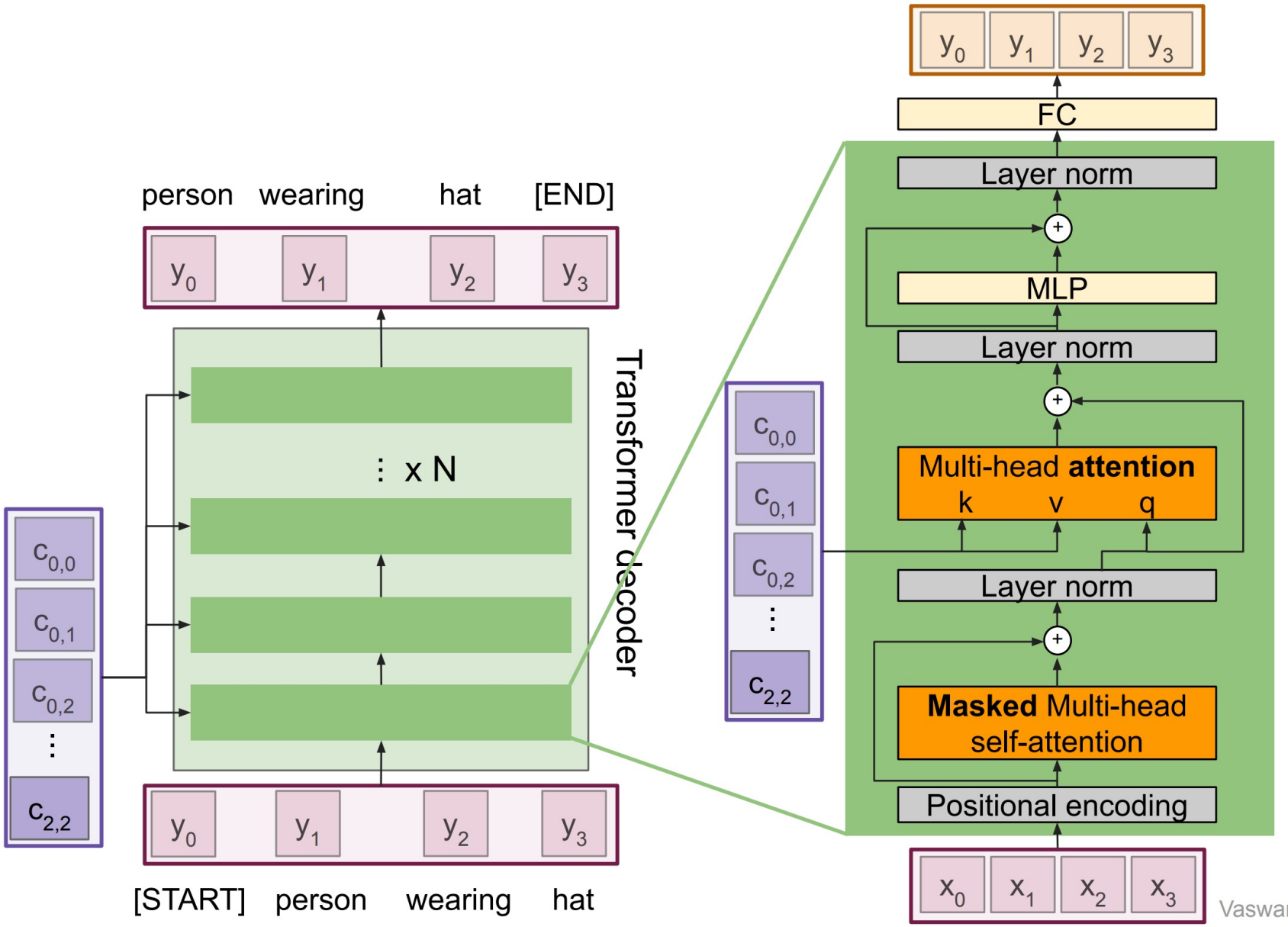
Self-attention is the only interaction between vectors.

Layer norm and MLP operate independently per vector.

Highly scalable, highly parallelizable, but high memory usage.

Vaswani et al., "Attention is All You Need," NIPS, 2017.

Transformer Decoder Block



Transformer Decoder Block:

Inputs: Set of vectors **x** and Set of context vectors **c.**
Outputs: Set of vectors **y.**

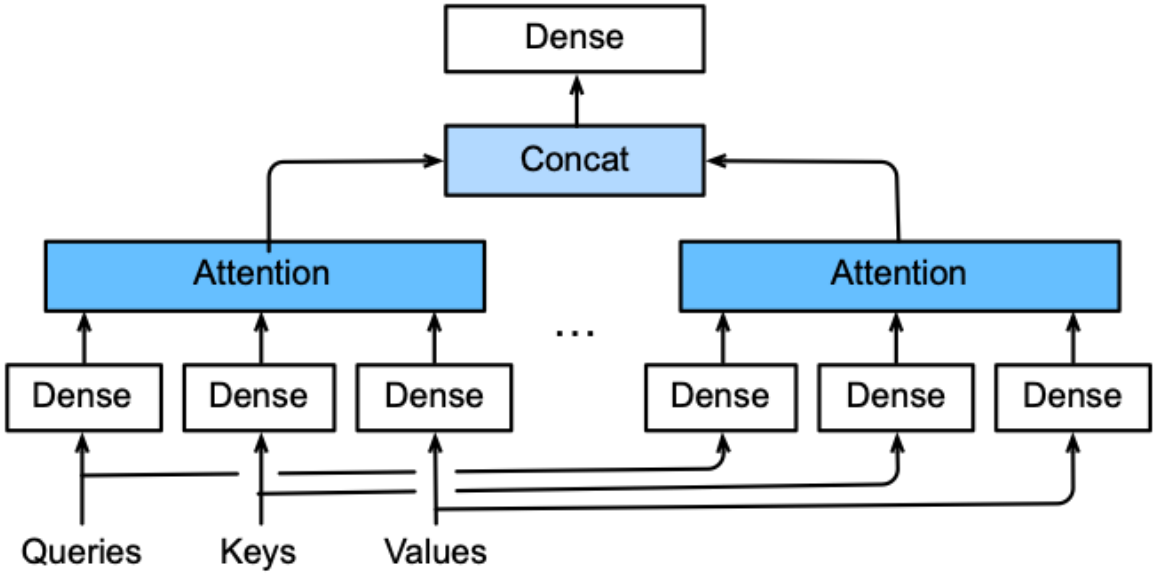
Masked Self-attention only interacts with past inputs.

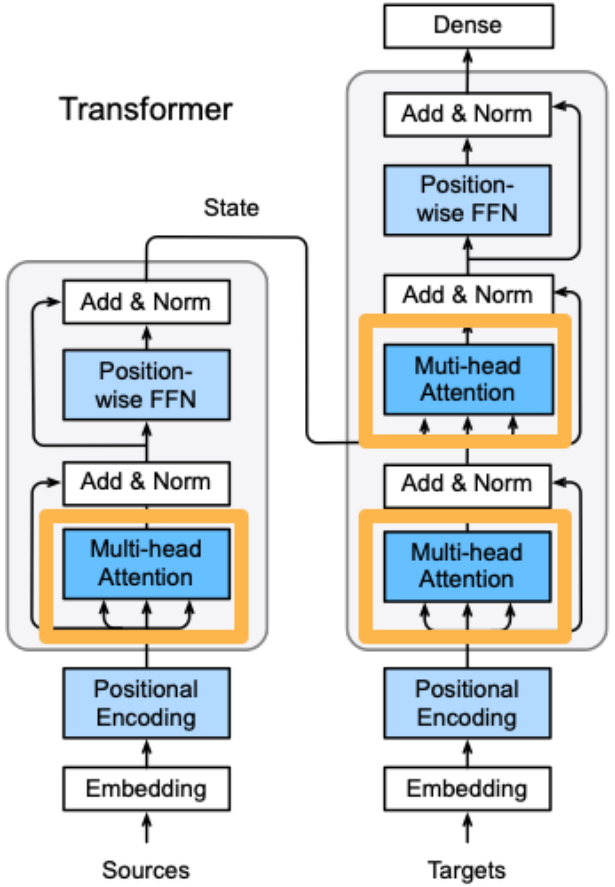
Multi-head attention block is NOT self-attention. It attends over encoder outputs.

Highly scalable, highly parallelizable, but high memory usage.

Vaswani et al, "Attention is all you need", NeurIPS 2017

Multi-head Attention





$$\mathbf{W}_q^{(i)} \in \mathbb{R}^{p_q \times d_q}, \mathbf{W}_k^{(i)} \in \mathbb{R}^{p_k \times d_k}, \text{ and } \mathbf{W}_v^{(i)} \in \mathbb{R}^{p_v \times d_v}$$

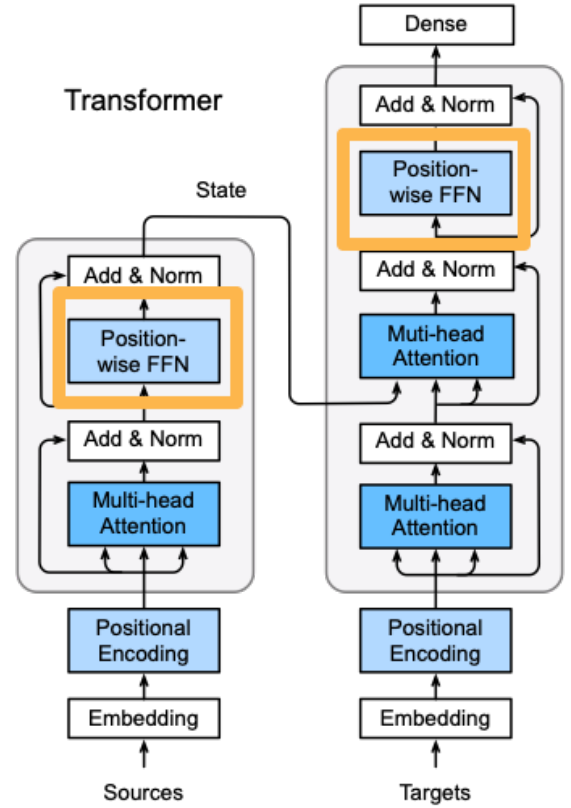
$$\mathbf{o}^{(i)} = \text{attention}(\mathbf{W}_q^{(i)}\mathbf{q}, \mathbf{W}_k^{(i)}\mathbf{k}, \mathbf{W}_v^{(i)}\mathbf{v})$$

$$\text{for } i = 1, \ldots, h \qquad \mathbf{o} = \mathbf{W}_o \begin{bmatrix} \mathbf{o}^{(1)} \\ \vdots \\ \mathbf{o}^{(h)} \end{bmatrix}$$

Vaswani et al., “Attention is All You Need,” NIPS, 2017.

Position-wise Feed-Forward Networks

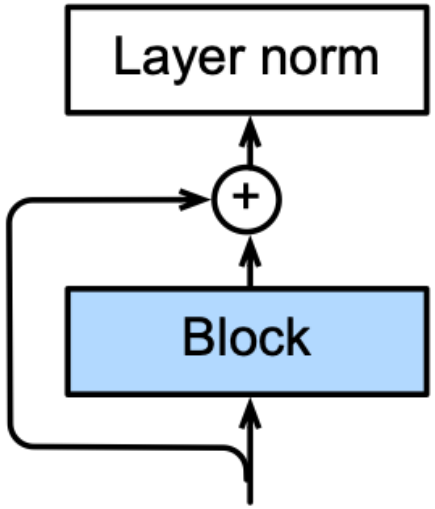
- Reshape input (batch, seq len, fea size) into (batch * seq len, fea size)
- Apply a two layer MLP
- Reshape back into 3-D
- Equals to apply two (1,1) conv layers

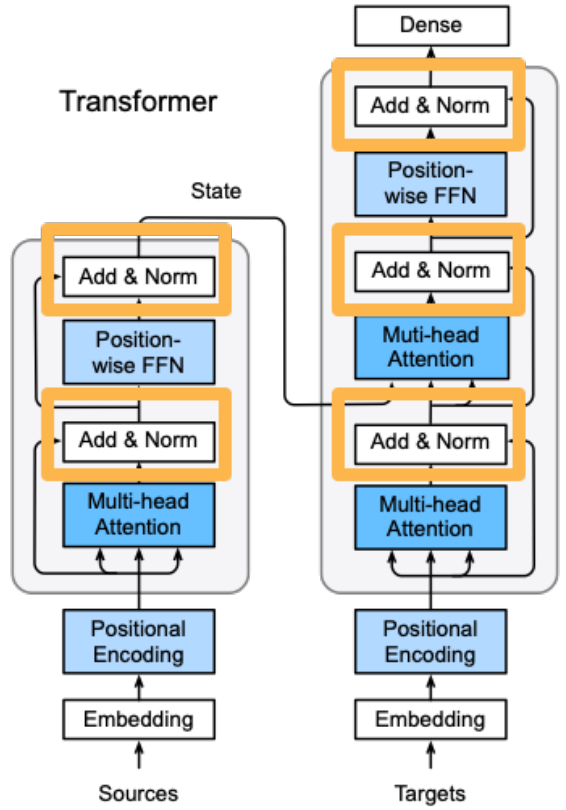


Vaswani et al., “Attention is All You Need,” NIPS, 2017.

Add and Norm

- Layer norm is similar to batch norm
- But the mean and variances are calculated along the last dimension
- `X.mean(axis=-1)` instead of the first batch dimension in batch norm `X.mean(axis=0)`





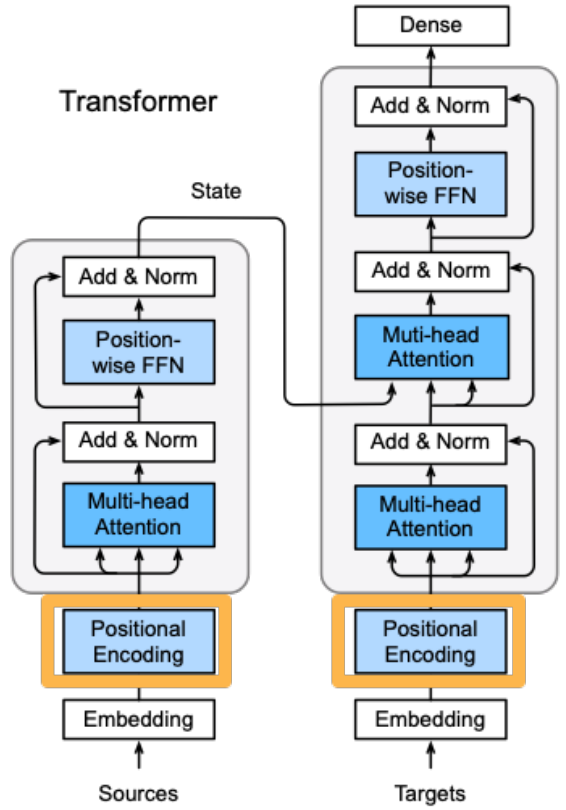
Vaswani et al., “Attention is All You Need,” NIPS, 2017.

Positional Encoding

- Assume embedding output $X \in \mathbb{R}^{l \times d}$ with shape (seq len, embed dim)
- Create $P \in \mathbb{R}^{l \times d}$ with

$$P_{i,2j} = \sin(i/10000^{2j/d})$$
$$P_{i,2j+1} = \cos(i/10000^{2jd})$$

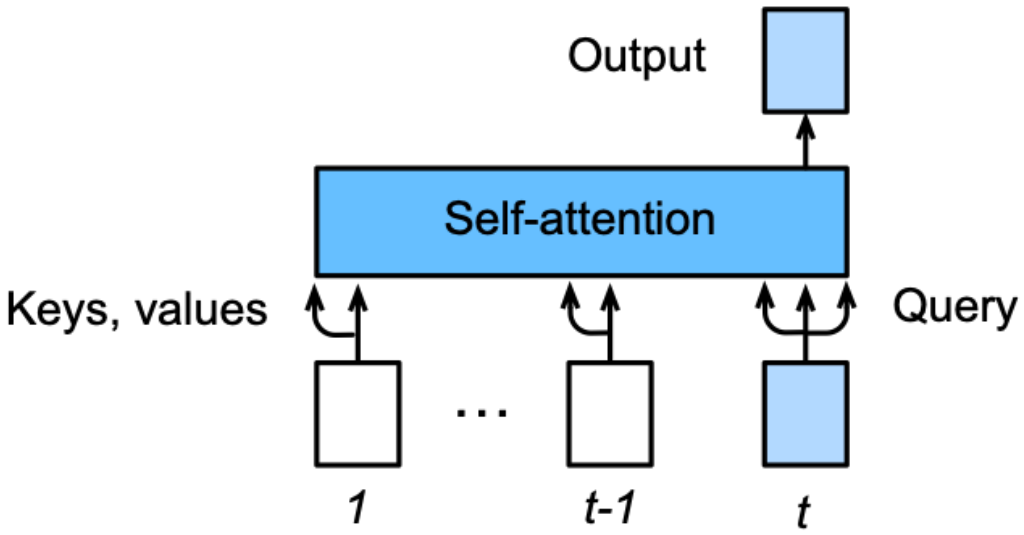
- Output $X + P$



Vaswani et al., "Attention is All You Need," NIPS, 2017.

Predicting

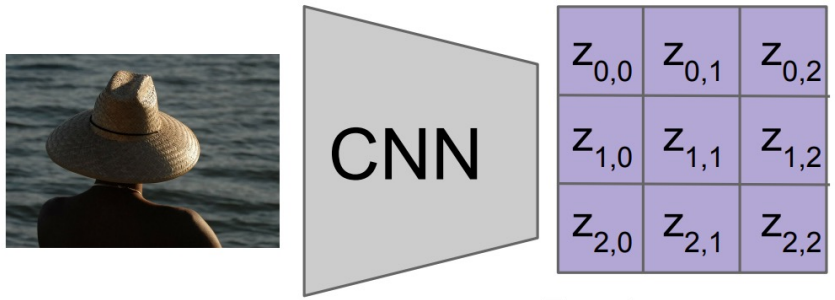
- Predict at time *t*:
  - Inputs of previous times as keys and values
  - Input at time *t* as query, as well as key and value, to predict output



Vaswani et al., “Attention is All You Need,” NIPS, 2017.

Image Captioning with Transformer

Input: Image **I**
Output: Sequence **y** = $y_1, y_2, \ldots, y_T$



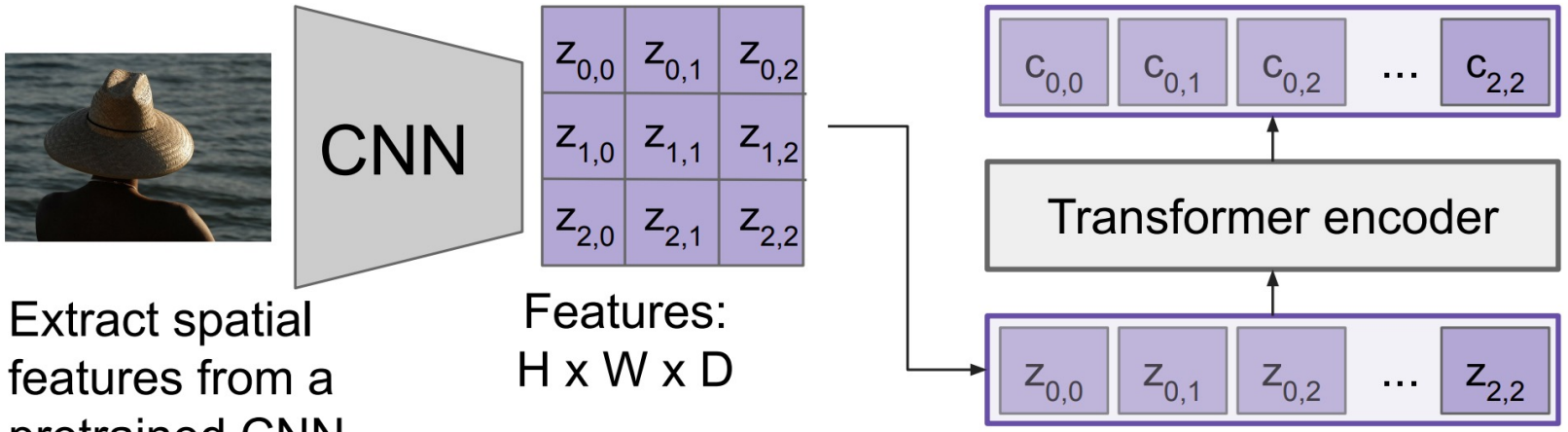
Extract spatial features from a pretrained CNN

Features: H x W x D

Image Captioning with Transformer

Input: Image **I**
Output: Sequence **y** = $y_1, y_2, ..., y_T$

Encoder: **c** = $T_W(\mathbf{z})$
where **z** is spatial CNN features
$T_W(.)$ is the transformer encoder



Extract spatial features from a pretrained CNN

Features: H x W x D

Image Captioning with Transformer

Input: Image **I**
Output: Sequence **y** = $y_1, y_2, ..., y_T$

Encoder: $\mathbf{c} = T_W(\mathbf{z})$
where **z** is spatial CNN features
$T_W(.)$ is the transformer encoder

Decoder: $y_t = T_D(\mathbf{y}_{0:t-1}, \mathbf{c})$
where $T_D(.)$ is the transformer decoder

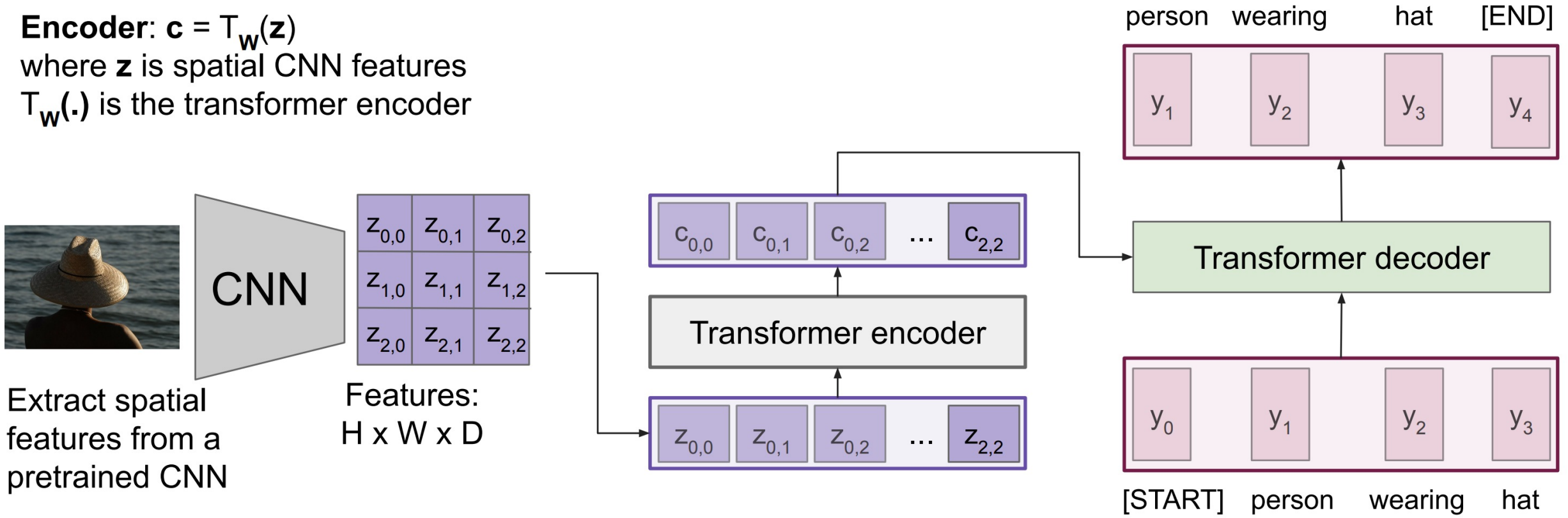


Image Captioning with Only Transformer

- **Perhaps we don't need convolutions at all?**

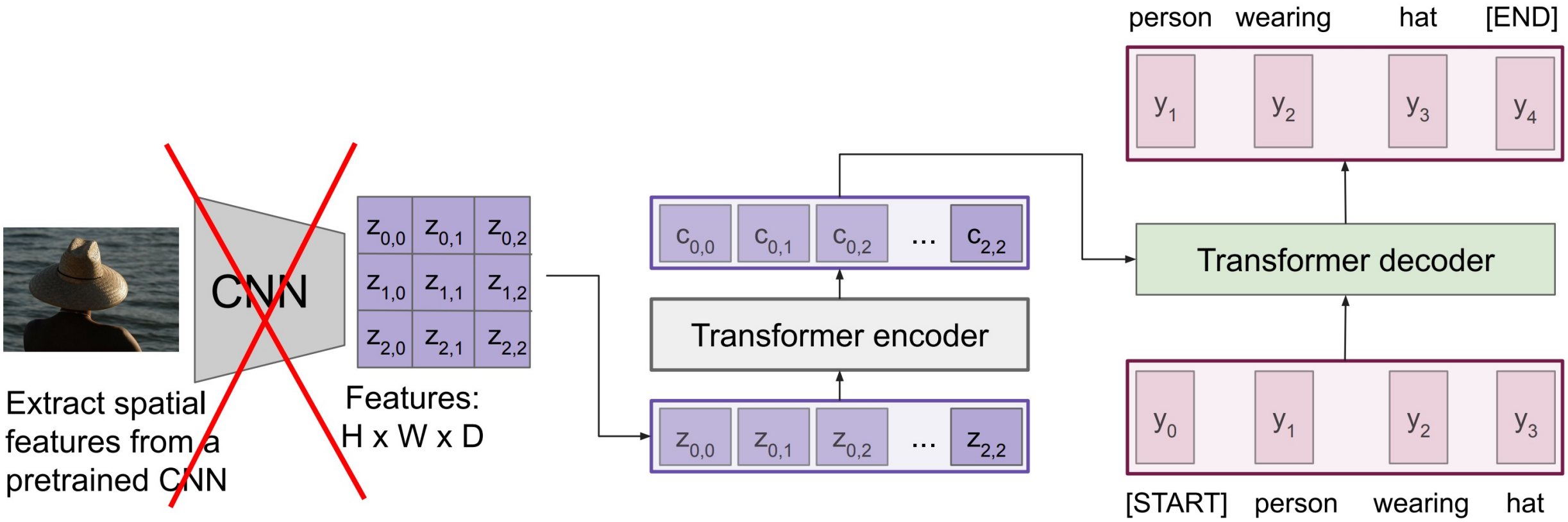
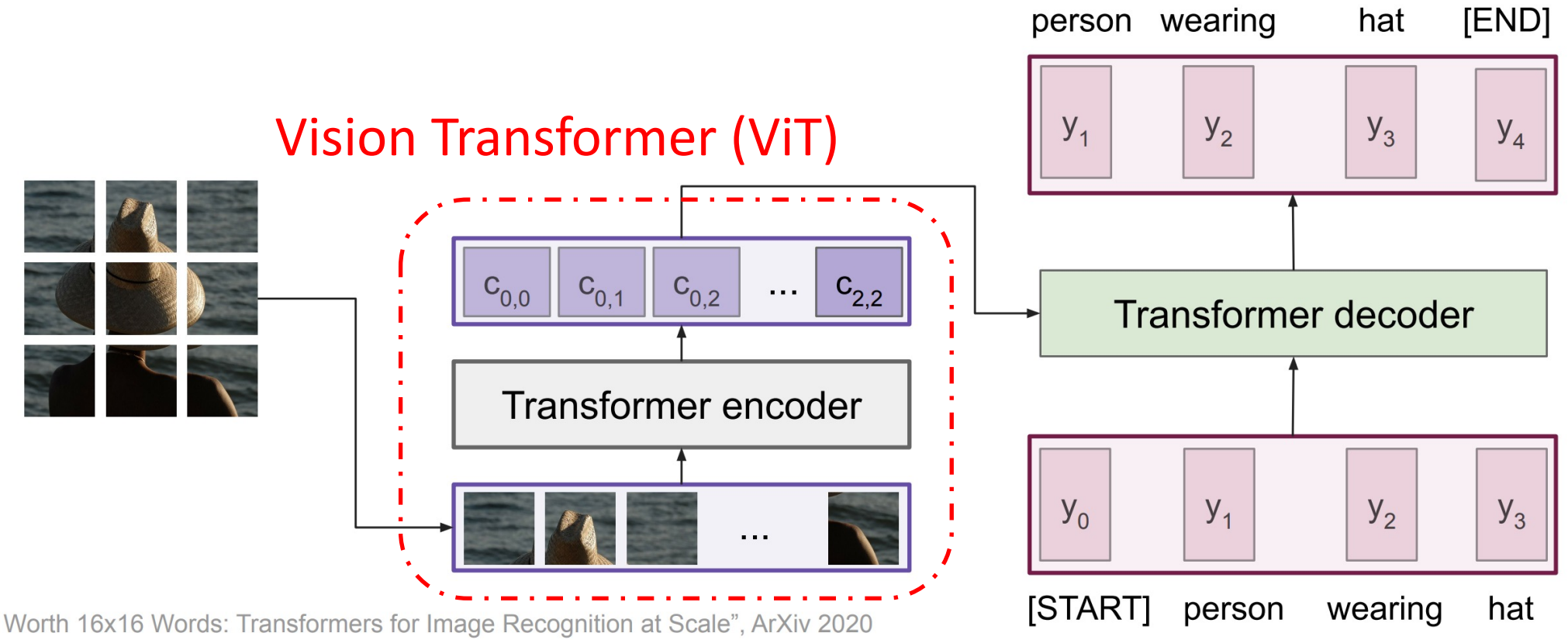


Image Captioning with Only Transformer

- **Transformers from pixels to language**



Dosovitskiy et al, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale", ArXiv 2020

Comparing RNNs to Transformers

RNNs

(+) LSTMs work reasonably well for long sequences.

(-) Expects an ordered sequences of inputs

(-) Sequential computation: subsequent hidden states can only be computed after the previous ones are done.

Transformers:

(+) Good at long sequences. Each attention calculation looks at all inputs.

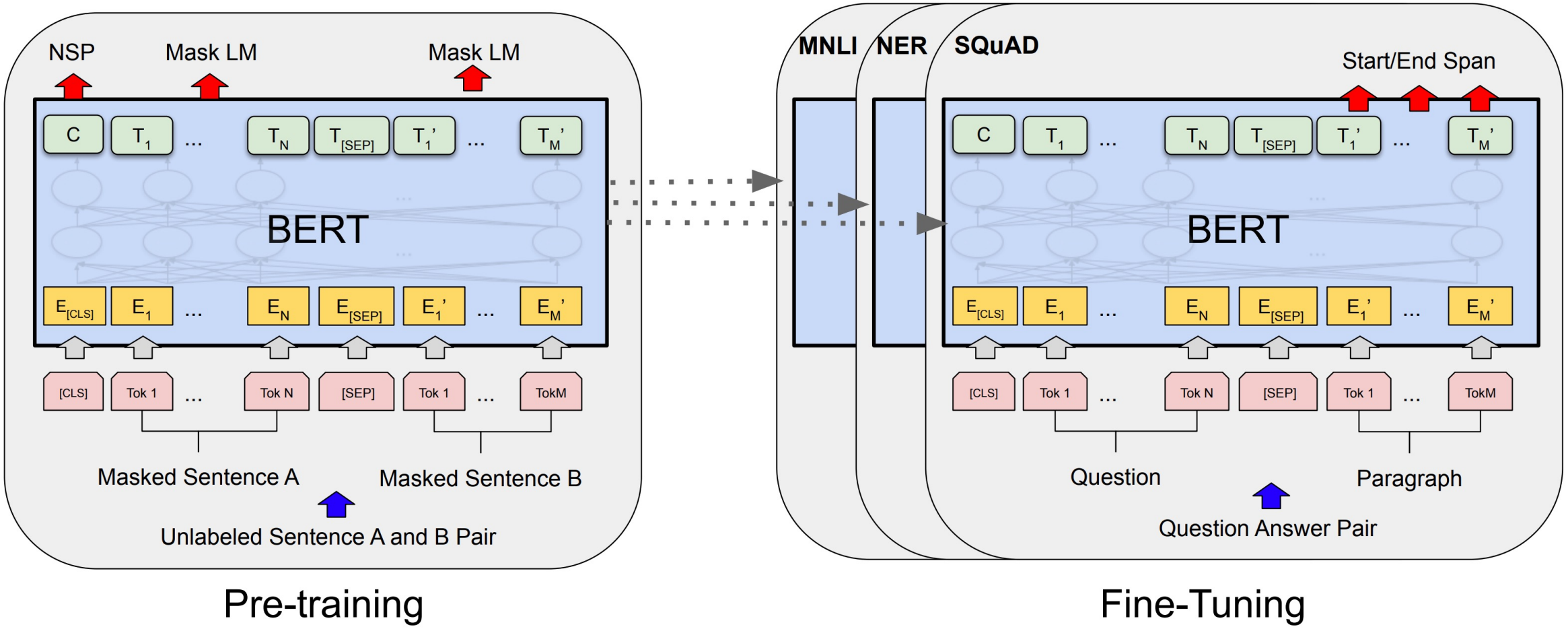
(+) Can operate over unordered sets or ordered sequences with positional encodings.

(+) Parallel computation: All alignment and attention scores for all inputs can be done in parallel.

(-) Requires a lot of memory: N x M alignment and attention scalers need to be calculated and stored for a single self-attention head. (but GPUs are getting bigger and better)

Vaswani et al., “Attention is All You Need,” NIPS, 2017.

BERT Architecture



Devlin et al., "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," NAACL, 2019.

BERT Architecture

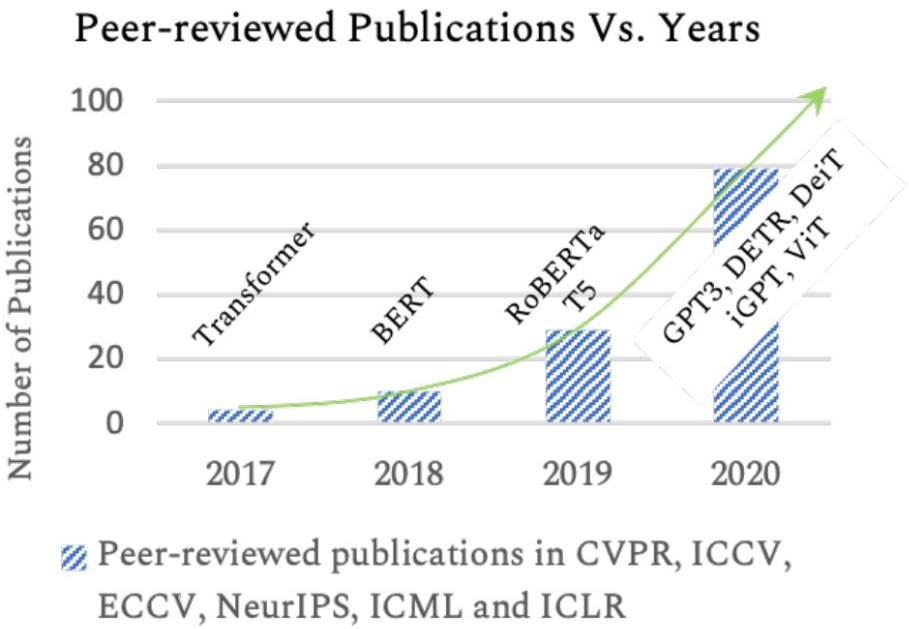
- **A (big) Transformer encoder (without the decoder)**
- **Two variants:**
  - Base: #blocks = 12, hidden size = 768, #heads = 12, #parameters = 110M
  - Large: #blocks = 24, hidden size = 1024, #heads = 16, #parameter = 340M
- **Train on large-scale corpus (books and wikipedia) with > 3B words**

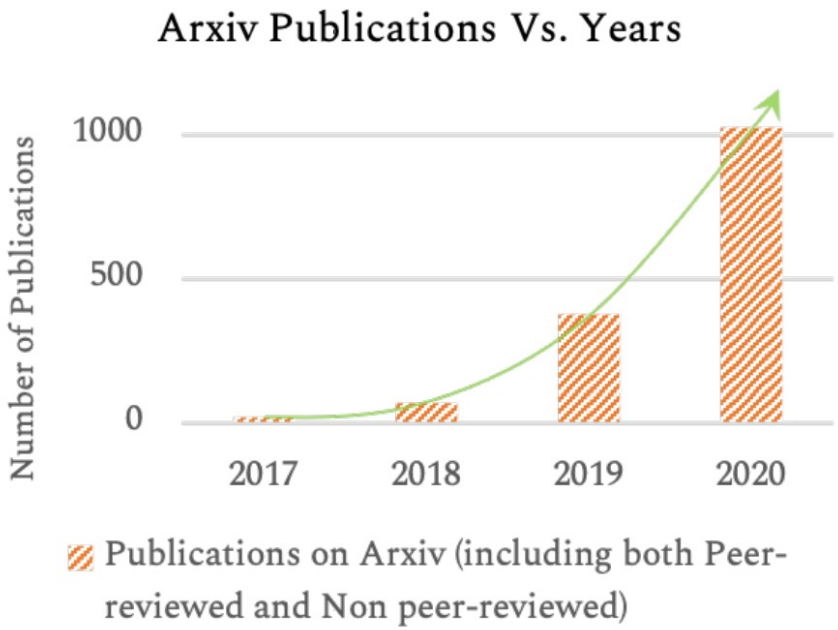
Devlin et al., “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” NAACL, 2019.

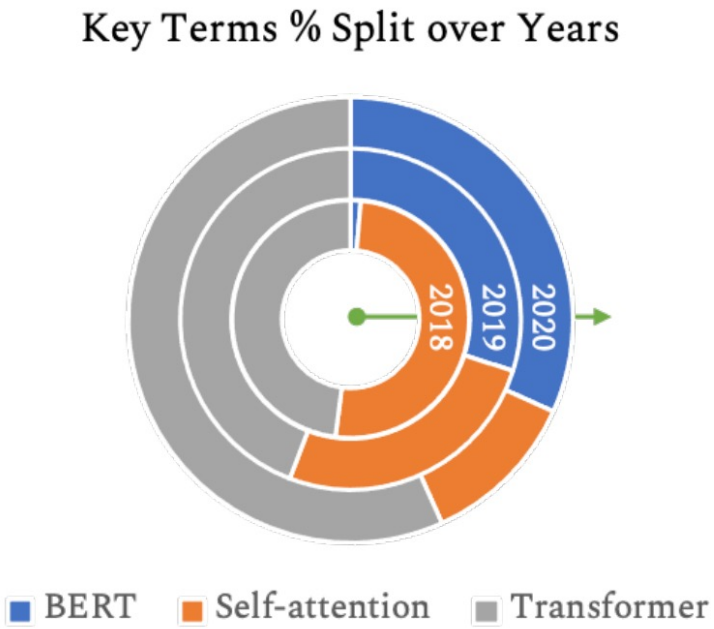
Vision Transformer (ViT)

- Dosovitskiy et al., "An Image is Worth 16 x 16 words: Transformers for Image Recognition at Scale," ICLR, 2021.
- This paper shows that the reliance on CNNs in computer vision is **NOT** necessary and a pure structure in place.

Transformers in Vision



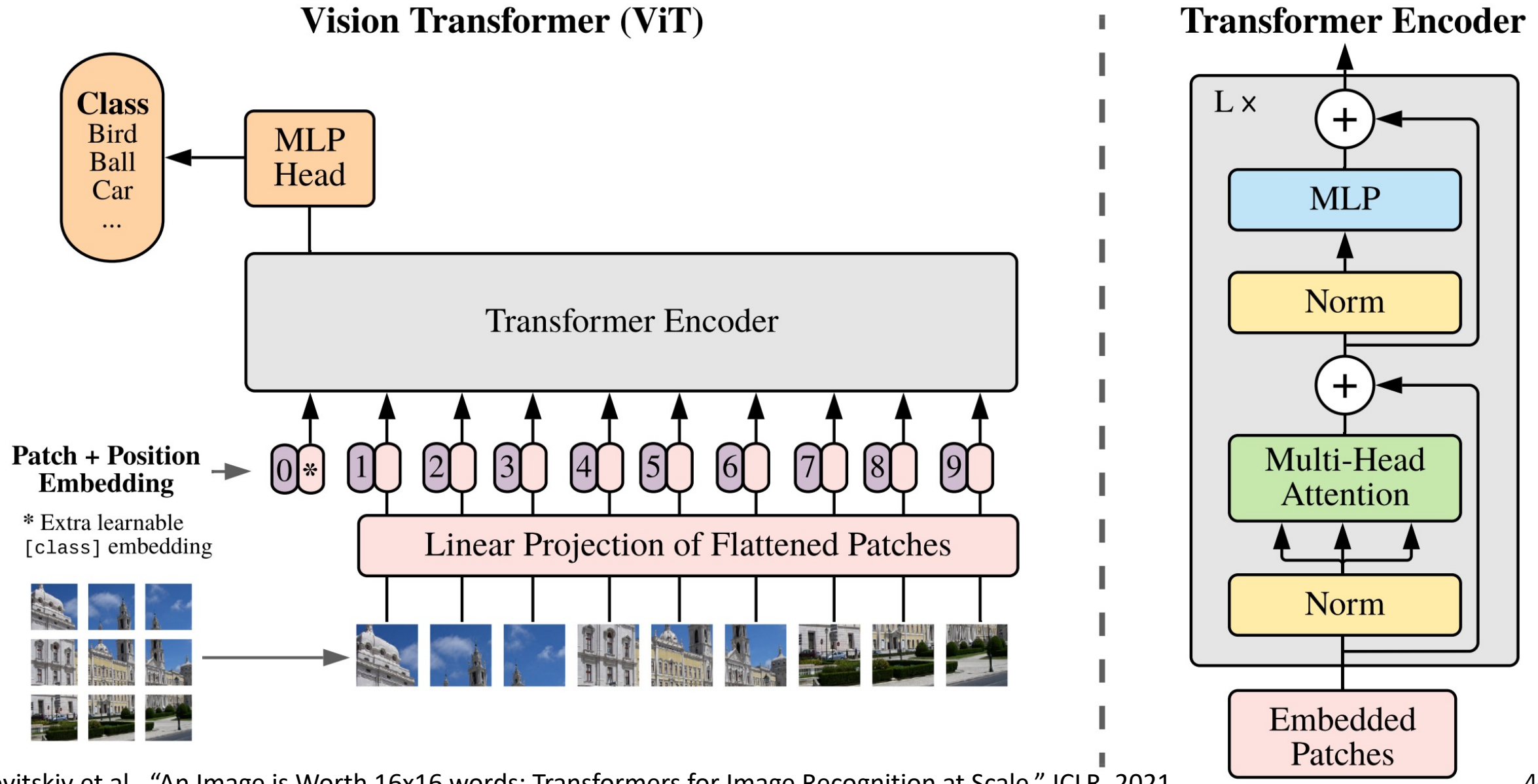




Statistics on the number of times keywords such as BERT, Self-Attention, and Transformers

Khan et al., "Transformers in Vision: A Survey," arxiv, 2021.

Vision Transformer (ViT)



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

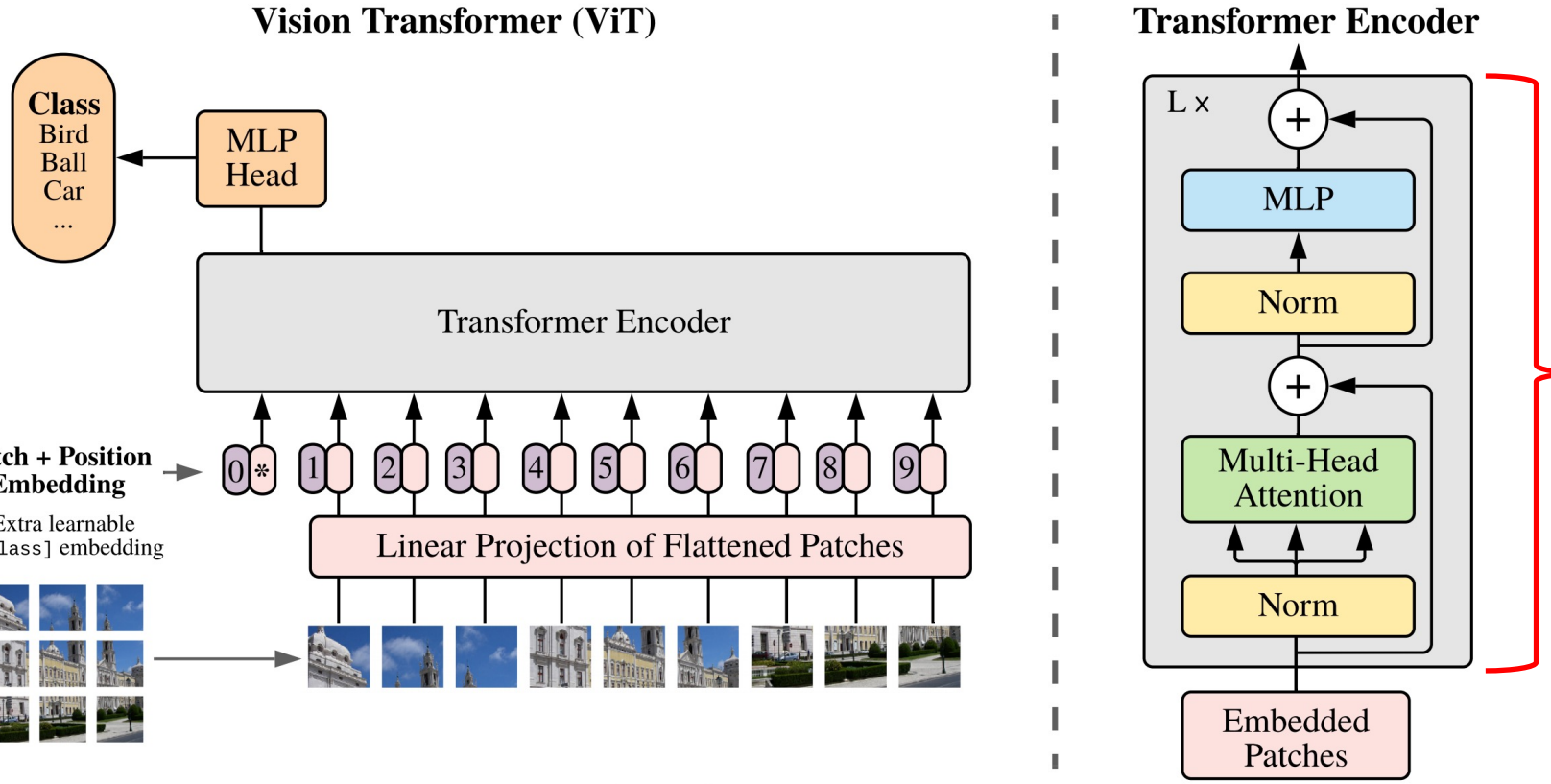
Vision Transformer (ViT)

$$\mathbf{z}_0 = [\mathbf{x}_{\text{class}};\ \mathbf{x}_p^1\mathbf{E};\ \mathbf{x}_p^2\mathbf{E};\cdots;\ \mathbf{x}_p^N\mathbf{E}] + \mathbf{E}_{pos}, \qquad \mathbf{E} \in \mathbb{R}^{(P^2\cdot C)\times D},\ \mathbf{E}_{pos} \in \mathbb{R}^{(N+1)\times D}$$

$$\mathbf{z}'_\ell = \text{MSA}(\text{LN}(\mathbf{z}_{\ell-1})) + \mathbf{z}_{\ell-1}, \qquad \ell = 1 \ldots L$$

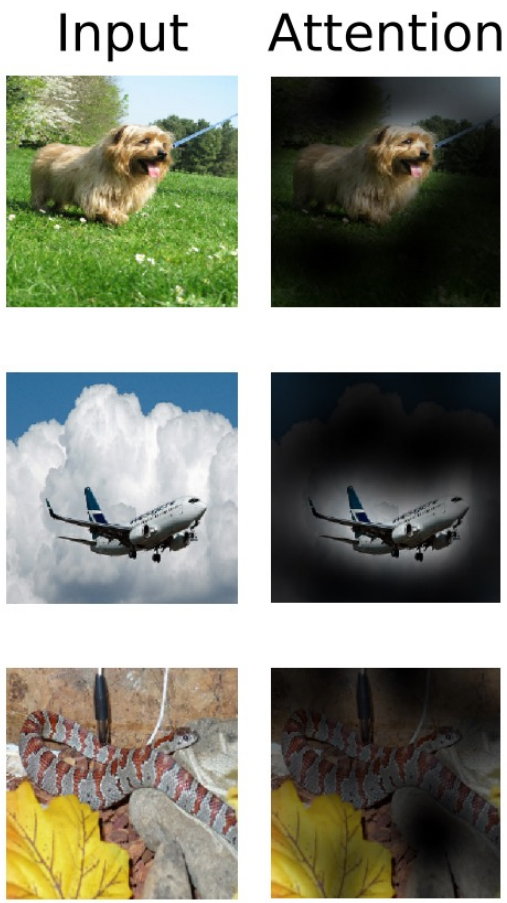
$$\mathbf{z}_\ell = \text{MLP}(\text{LN}(\mathbf{z}'_\ell)) + \mathbf{z}'_\ell, \qquad \ell = 1 \ldots L$$

$$\mathbf{y} = \text{LN}(\mathbf{z}_L^0)$$



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

Vision Transformer (ViT)

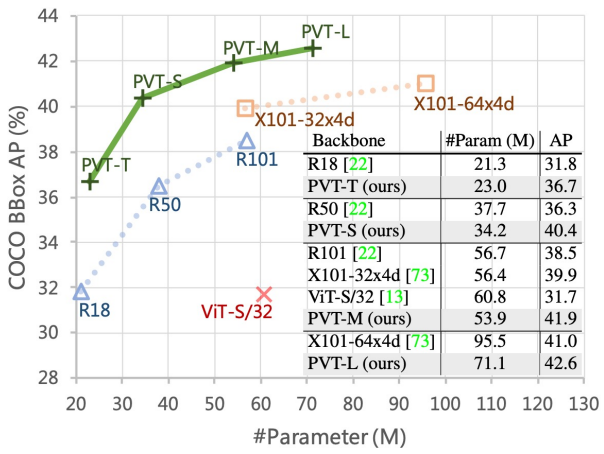


	Ours-JFT (ViT-H/14)	Ours-JFT (ViT-L/16)	Ours-I21k (ViT-L/16)	BiT-L (ResNet152x4)	Noisy Student (EfficientNet-L2)
ImageNet	**88.55** ± 0.04	87.76 ± 0.03	85.30 ± 0.02	87.54 ± 0.02	88.4/88.5*
ImageNet ReaL	**90.72** ± 0.05	90.54 ± 0.03	88.62 ± 0.05	90.54	90.55
CIFAR-10	**99.50** ± 0.06	99.42 ± 0.03	99.15 ± 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

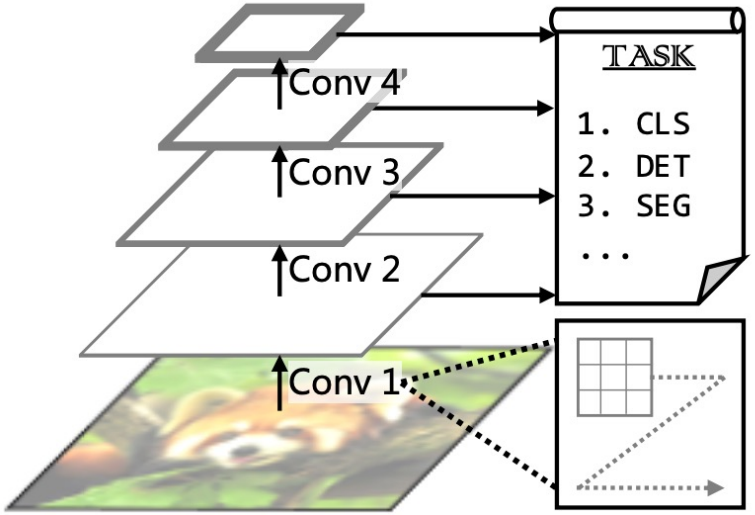
Dosovitskiy et al., “An Image is Worth 16x16 words: Transformers for Image Recognition at Scale,” ICLR, 2021.

Pyramid Vision Transformer (PVT)

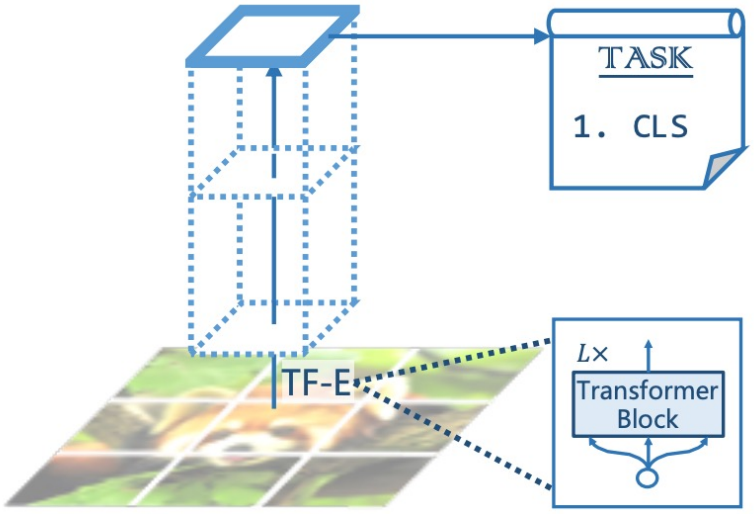
Comparison



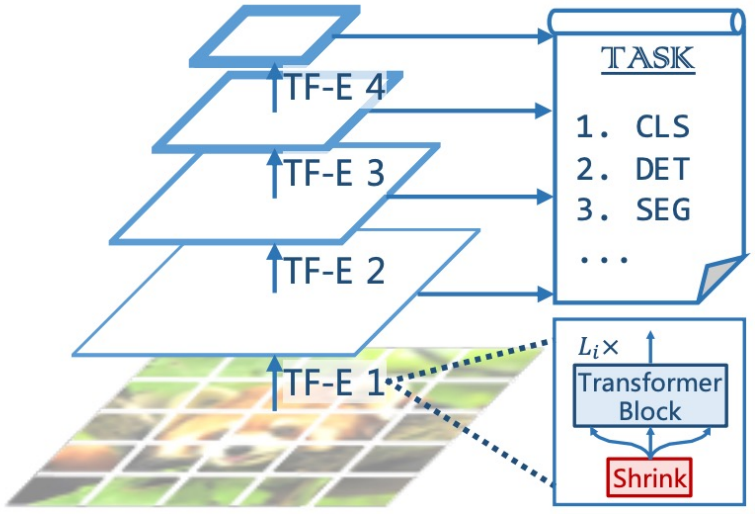
Backbone	#Param (M)	AP
R18 [22]	21.3	31.8
PVT-T (ours)	23.0	36.7
R50 [22]	37.7	36.3
PVT-S (ours)	34.2	40.4
R101 [22]	56.7	38.5
X101-32x4d [73]	56.4	39.9
ViT-S/32 [13]	60.8	31.7
PVT-M (ours)	53.9	41.9
X101-64x4d [73]	95.5	41.0
PVT-L (ours)	71.1	42.6



(a) CNNs: VGG [54], ResNet [22], *etc.*



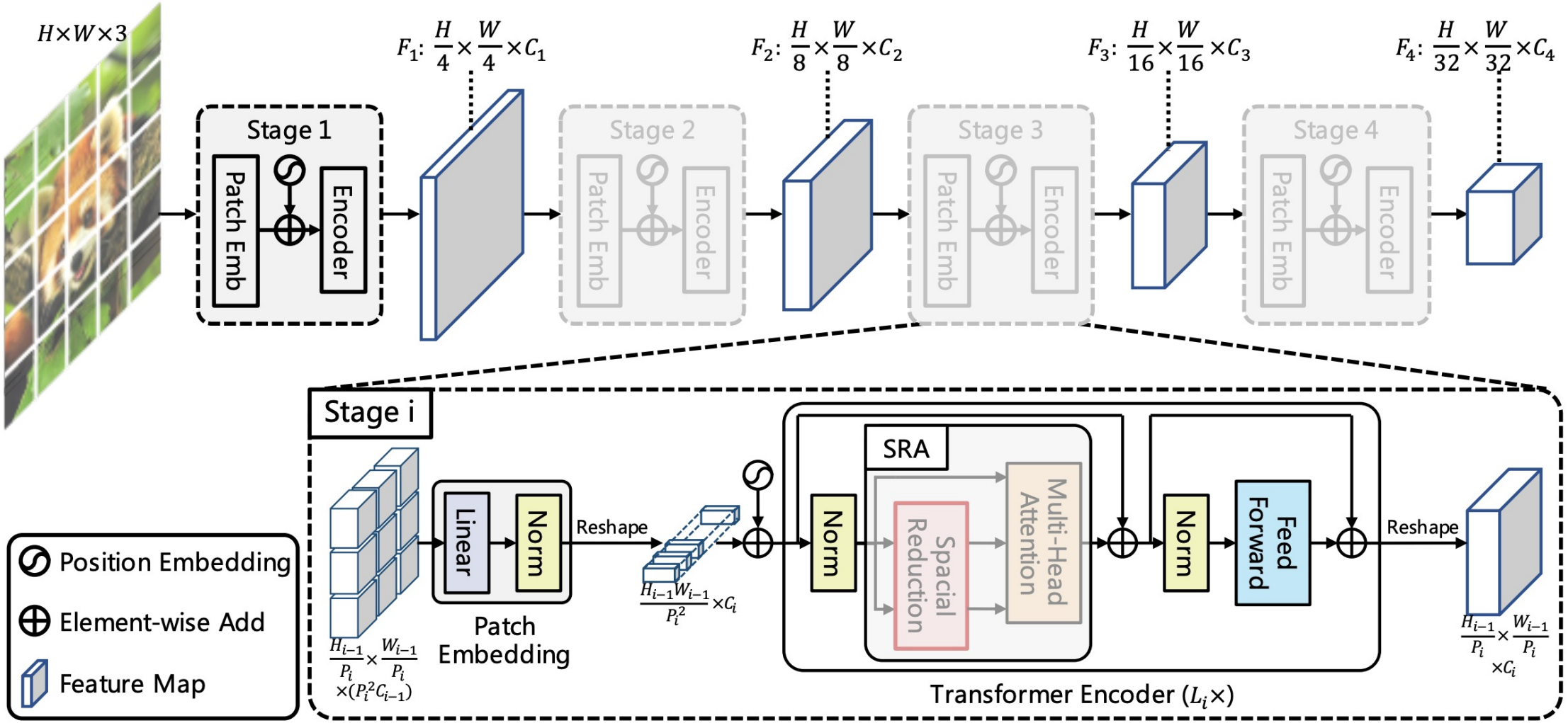
(b) Vision Transformer [13]



(c) Pyramid Vision Transformer (ours)

Pyramid Vision Transformer (PVT)

Network Architecture



Wang et al., "Pyramid Vision Transformer: A Versatile Backbone for Dense Prediction without Convolutions," ArXiv, 2021.

Pyramid Vision Transformer (PVT)

Experimental Results

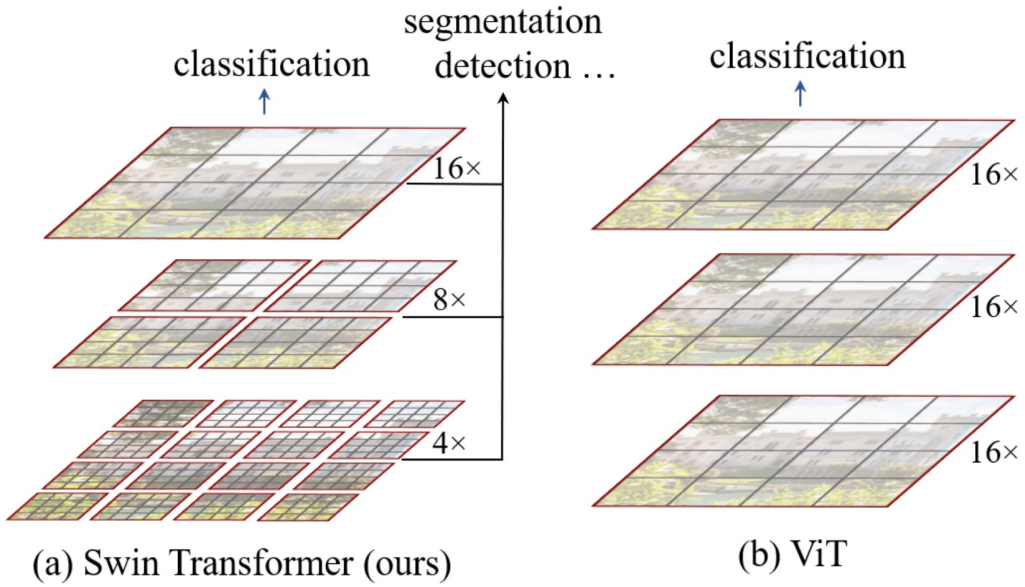
Backbone	#Param (M)	RetinaNet 1x						RetinaNet 3x + MS					
		AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
ResNet18 [22]	21.3	31.8	49.6	33.6	16.3	34.3	43.2	35.4	53.9	37.6	19.5	38.2	46.8
PVT-Tiny (ours)	23.0	36.7(+4.9)	56.9	38.9	22.6	38.8	50.0	39.4(+4.0)	59.8	42.0	25.5	42.0	52.1
ResNet50 [22]	37.7	36.3	55.3	38.6	19.3	40.0	48.8	39.0	58.4	41.8	22.4	42.8	51.6
PVT-Small (ours)	34.2	40.4(+4.1)	61.3	43.0	25.0	42.9	55.7	42.2(+3.2)	62.7	45.0	26.2	45.2	57.2
ResNet101 [22]	56.7	38.5	57.8	41.2	21.4	42.6	51.1	40.9	60.1	44.0	23.7	45.0	53.8
ResNeXt101-32x4d [73]	56.4	39.9(+1.4)	59.6	42.7	22.3	44.2	52.5	41.4(+0.5)	61.0	44.3	23.9	45.5	53.7
PVT-Medium (ours)	53.9	41.9(+3.4)	63.1	44.3	25.0	44.9	57.6	43.2(+2.3)	63.8	46.1	27.3	46.3	58.9
ResNeXt101-64x4d [73]	95.5	41.0	60.9	44.0	23.9	45.2	54.0	41.8	61.5	44.4	25.2	45.4	54.6
PVT-Large (ours)	71.1	42.6(+1.6)	63.7	45.4	25.8	46.0	58.4	43.4(+1.6)	63.6	46.1	26.1	46.0	59.5

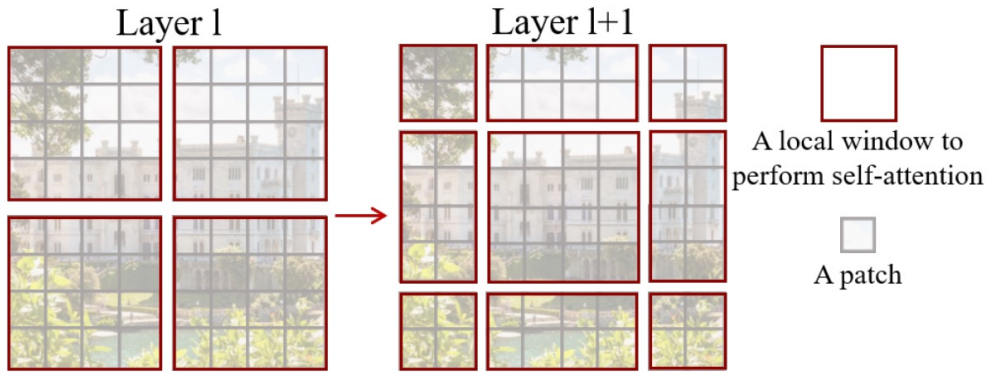
Backbone	#Param (M)	Mask R-CNN 1x						Mask R-CNN 3x + MS					
		AP^b	AP^b_{50}	AP^b_{75}	AP^m	AP^m_{50}	AP^m_{75}	AP^b	AP^b_{50}	AP^b_{75}	AP^m	AP^m_{50}	AP^m_{75}
ResNet18 [22]	31.2	34.0	54.0	36.7	31.2	51.0	32.7	36.9	57.1	40.0	33.6	53.9	35.7
PVT-Tiny (ours)	32.9	36.7(+2.7)	59.2	39.3	35.1(+3.9)	56.7	37.3	39.8(+2.9)	62.2	43.0	37.4(+3.8)	59.3	39.9
ResNet50 [22]	44.2	38.0	58.6	41.4	34.4	55.1	36.7	41.0	61.7	44.9	37.1	58.4	40.1
PVT-Small (ours)	44.1	40.4(+2.4)	62.9	43.8	37.8(+3.4)	60.1	40.3	43.0(+2.0)	65.3	46.9	39.9(+2.8)	62.5	42.8
ResNet101 [22]	63.2	40.4	61.1	44.2	36.4	57.7	38.8	42.8	63.2	47.1	38.5	60.1	41.3
ResNeXt101-32x4d [73]	62.8	41.9(+1.5)	62.5	45.9	37.5(+1.1)	59.4	40.2	44.0(+1.2)	64.4	48.0	39.2(+0.7)	61.4	41.9
PVT-Medium (ours)	63.9	42.0(+1.6)	64.4	45.6	39.0(+2.6)	61.6	42.1	44.2(+1.4)	66.0	48.2	40.5(+2.0)	63.1	43.5
ResNeXt101-64x4d [73]	101.9	42.8	63.8	47.3	38.4	60.6	41.3	44.4	64.9	48.8	39.7	61.9	42.6
PVT-Large (ours)	81.0	42.9(+0.1)	65.0	46.6	39.5(+1.1)	61.9	42.5	44.5(+0.1)	66.0	48.3	40.7(+1.0)	63.4	43.7

Wang et al., "Pyramid Vision Transformer: A Versatile Backbone for Dense Prediction without Convolutions," ArXiv, 2021.

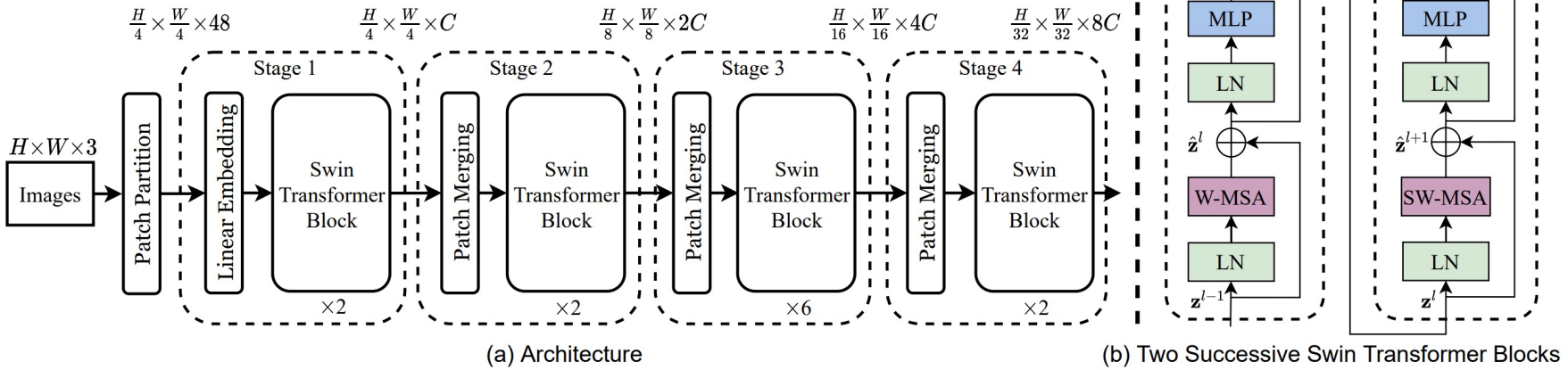
Swin Transformer

Hierarchical Vision Transformer Using Shifted Window



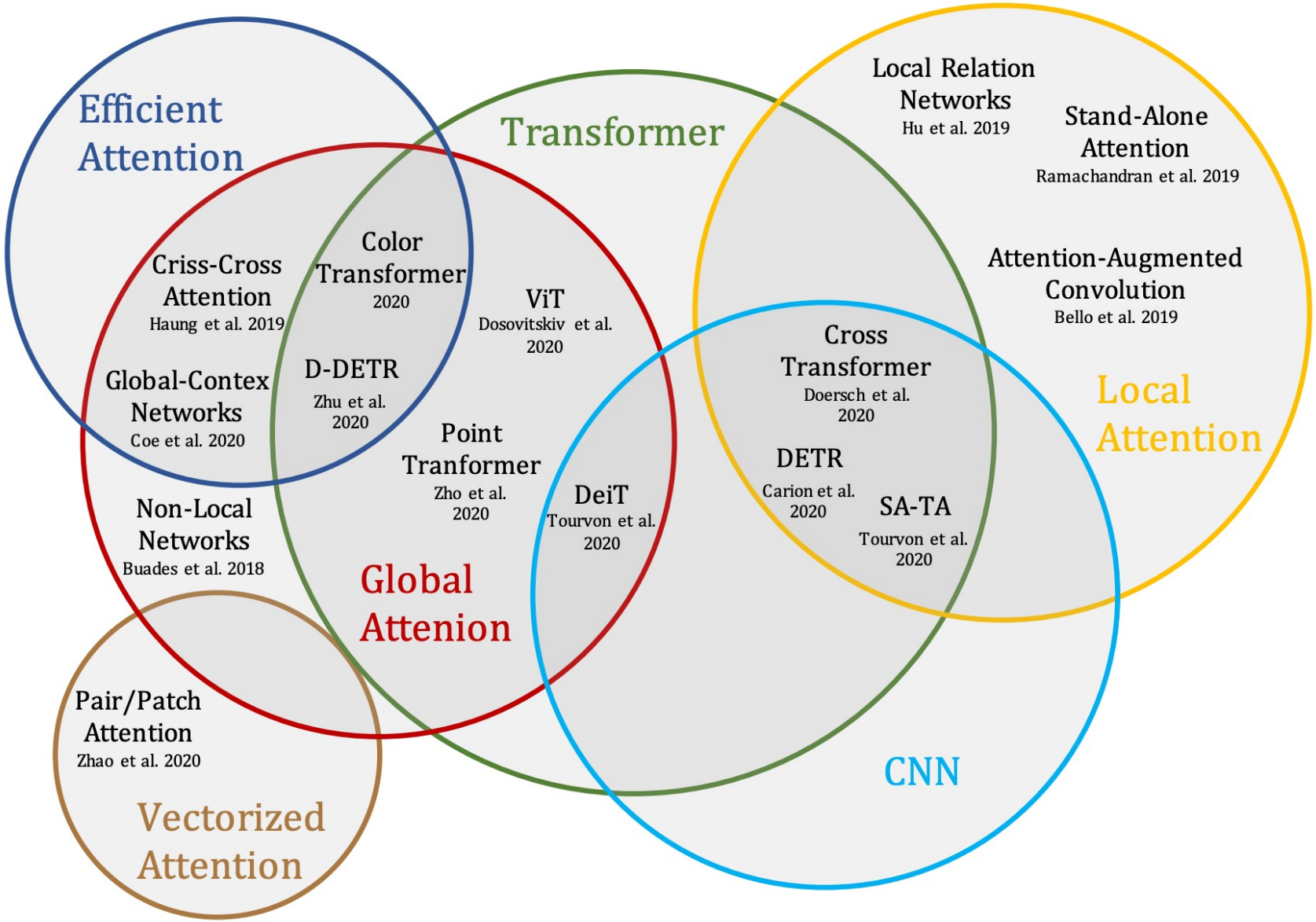


Shifted Window for Self Attention



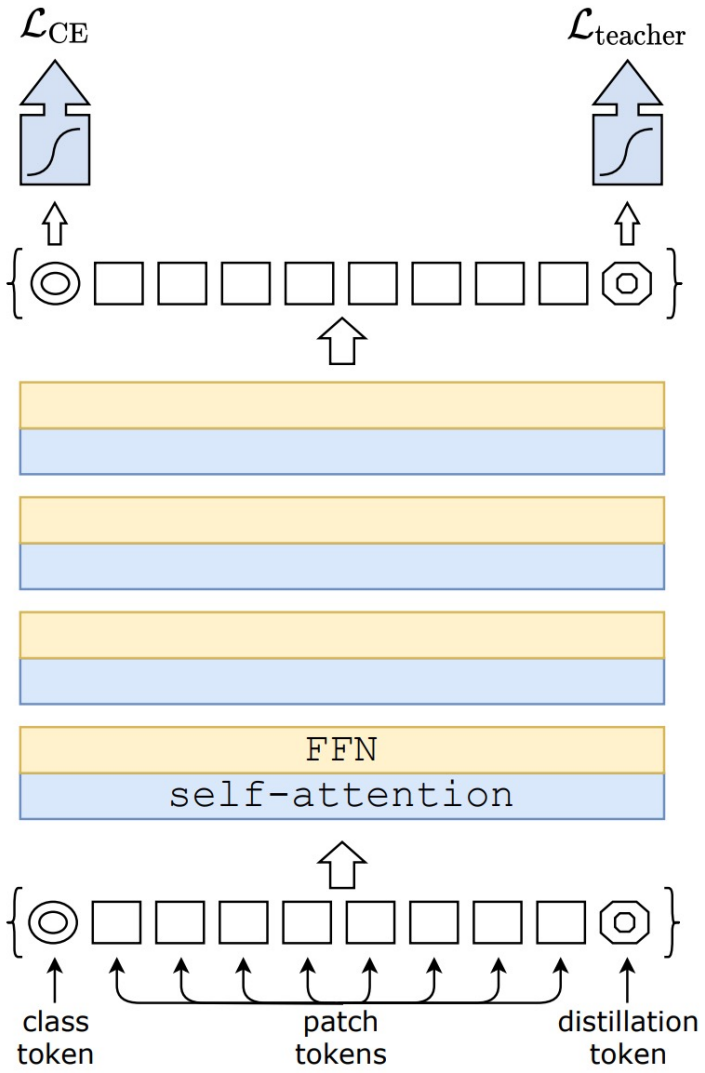
Liu et al., "Swin Transformer: Hierarchical Vision Transformer Using Shifted Window," ArXiv, 2021.

A Taxonomy of Self-Attention Design Space



Khan et al., "Transformers in Vision: A Survey," arxiv, 2021.

Data-efficient Image Transformer (DeiT)



ViT-B/16 [15]	86M	384^2	85.9	77.9	83.6	-
ViT-L/16 [15]	307M	384^2	27.3	76.5	82.2	-
DeiT-Ti	5M	224^2	2536.5	72.2	80.1	60.4
DeiT-S	22M	224^2	940.4	79.8	85.7	68.5
DeiT-B	86M	224^2	292.3	81.8	86.7	71.5
DeiT-B↑384	86M	384^2	85.9	83.1	87.7	72.4
DeiT-Ti⚗	6M	224^2	2529.5	74.5	82.1	62.9
DeiT-S⚗	22M	224^2	936.2	81.2	86.8	70.0
DeiT-B⚗	87M	224^2	290.9	83.4	88.3	73.2
DeiT-Ti⚗ / 1000 epochs	6M	224^2	2529.5	76.6	83.9	65.4
DeiT-S⚗ / 1000 epochs	22M	224^2	936.2	82.6	87.8	71.7
DeiT-B⚗ / 1000 epochs	87M	224^2	290.9	84.2	88.7	73.9
DeiT-B⚗ ↑384	87M	384^2	85.8	84.5	89.0	74.8
DeiT-B⚗ ↑384 / 1000 epochs	87M	384^2	85.8	85.2	89.3	75.2

Touvron et al., "Training Data-Efficient Image Transformer and Distillation Through Attention," arxiv, 2021.

Self-sup. Learning for Transformer (DINO)

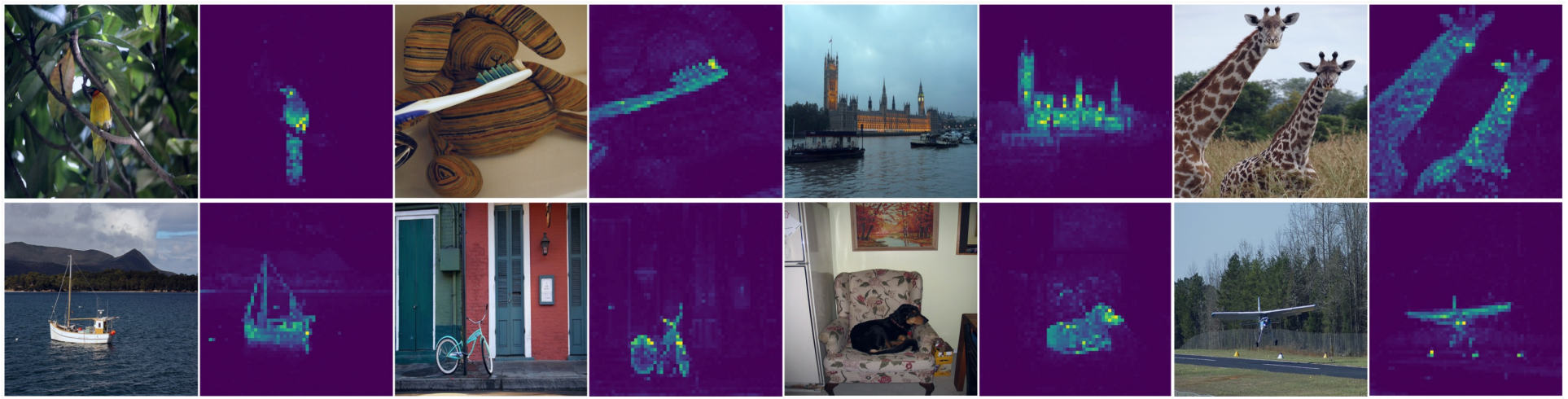
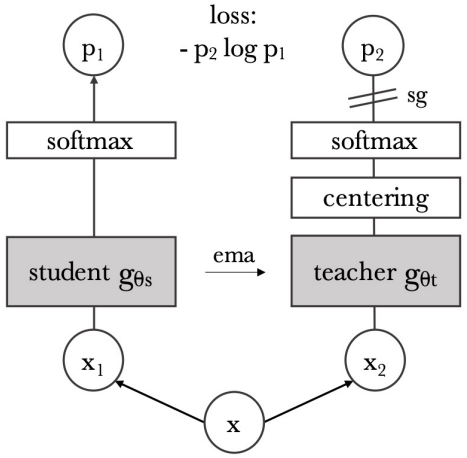


Figure 1: **Self-attention from a Vision Transformer with 8×8 patches trained with no supervision.** We look at the self-attention of the `[CLS]` token on the heads of the last layer. This token is not attached to any label nor supervision. These maps show that the model automatically learns class-specific features leading to unsupervised object segmentations.



SCLR [12]	RN50w4	375	117	76.8	69.3
SwAV [10]	RN50w2	93	384	77.3	67.3
BYOL [30]	RN50w2	93	384	77.4	–
DINO	ViT-B/16	85	312	78.2	76.1
SwAV [10]	RN50w5	586	76	78.5	67.1
BYOL [30]	RN50w4	375	117	78.6	–
BYOL [30]	RN200w2	250	123	79.6	73.9
DINO	ViT-S/8	21	180	79.7	**78.3**
SCLRv2 [13]	RN152w3+SK	794	46	79.8	73.1
DINO	ViT-B/8	85	63	**80.1**	77.4

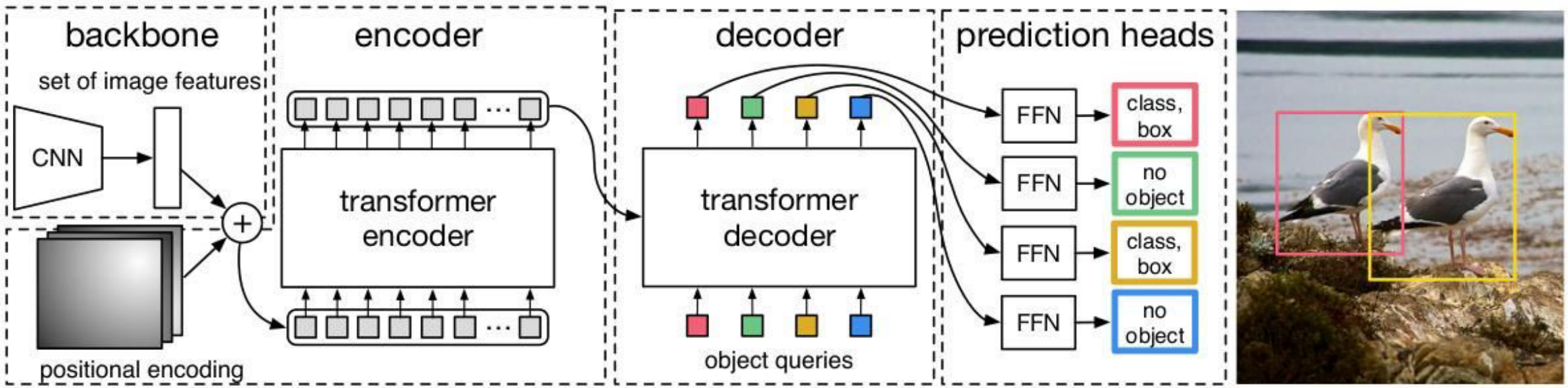
Caron et al., "Emerging Properties in Self-Supervised Vision Transformers," arxiv, 2021.

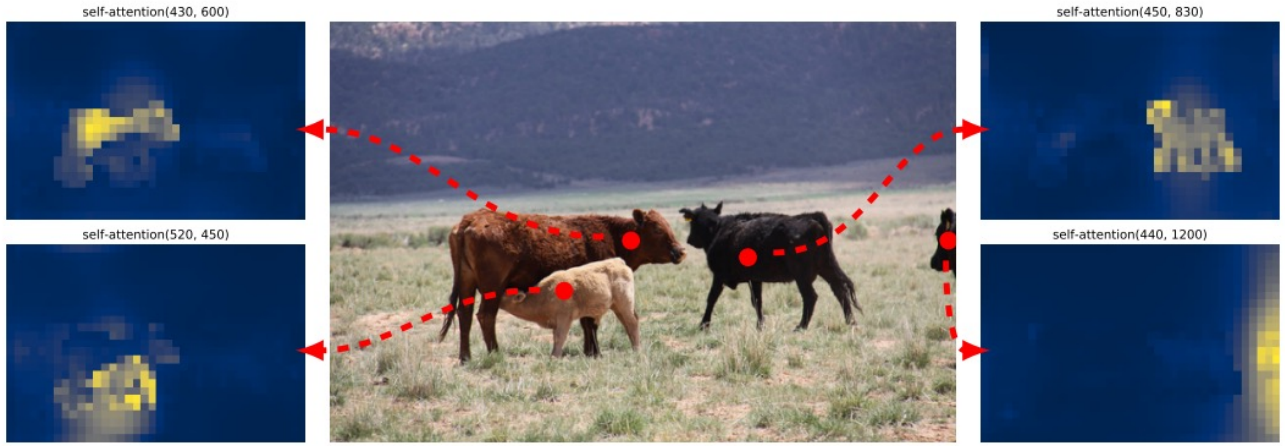
Empirical Study of Training Self-Sup. Vision Transformers

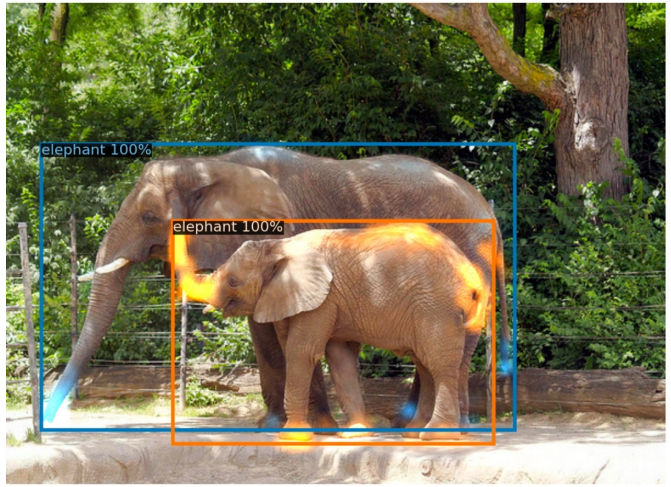
framework	model	params	acc. (%)
linear probing:			
iGPT [9]	iGPT-L	1362M	69.0
iGPT [9]	iGPT-XL	6801M	72.0
MoCo v3	ViT-B	86M	76.7
MoCo v3	ViT-L	304M	77.6
MoCo v3	ViT-H	632M	78.1
MoCo v3	ViT-BN-H	632M	79.1
MoCo v3	ViT-BN-L/7	304M	**81.0**
end-to-end fine-tuning:			
masked patch pred. [16]	ViT-B	86M	79.9†
MoCo v3	ViT-B	86M	83.2
MoCo v3	ViT-L	304M	**84.1**

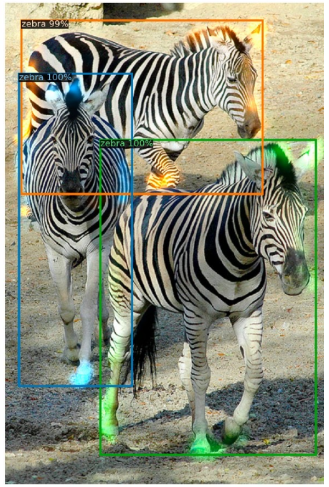
Chen et al., "An Empirical Study of Training Self-Supervised Vision Transformers," arxiv, 2021.

Object Detection with Transformers



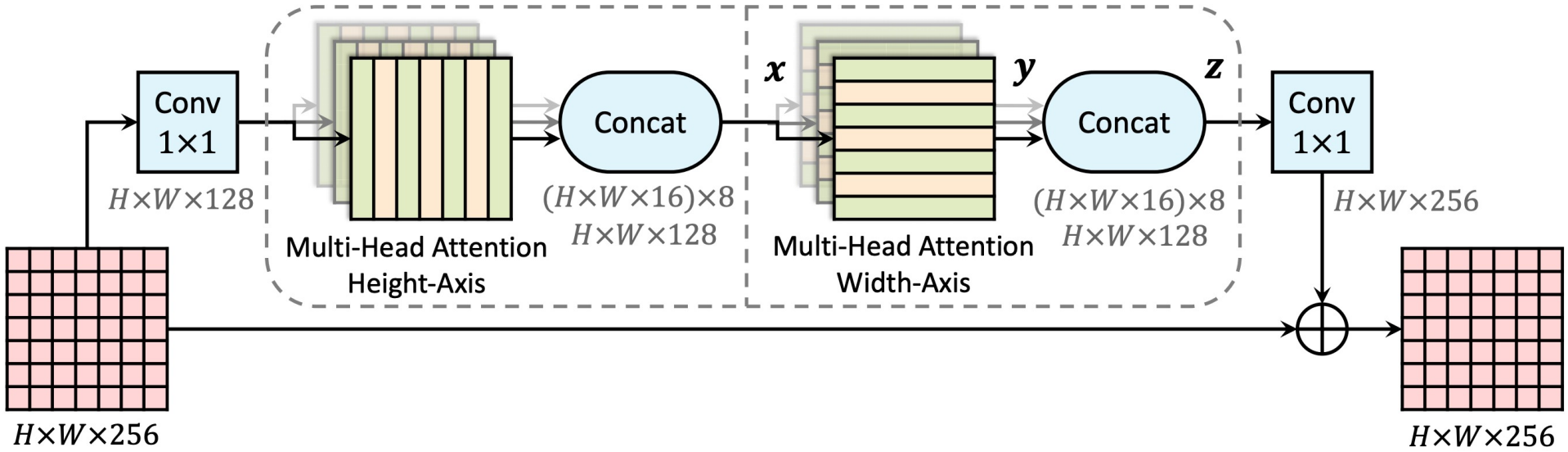




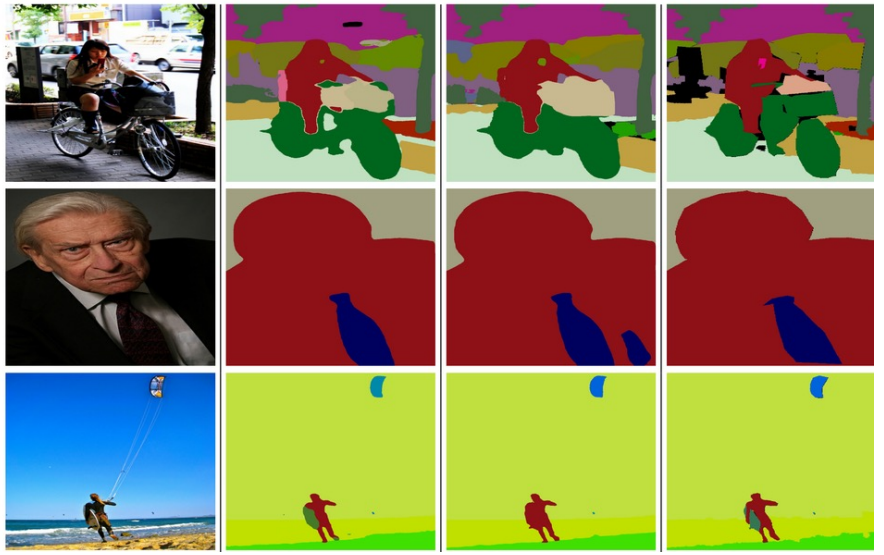


Carion et al., "End-to-End Object Detection with Transformers," ECCV, 2020.

Panoptic Segmentation with Transformers

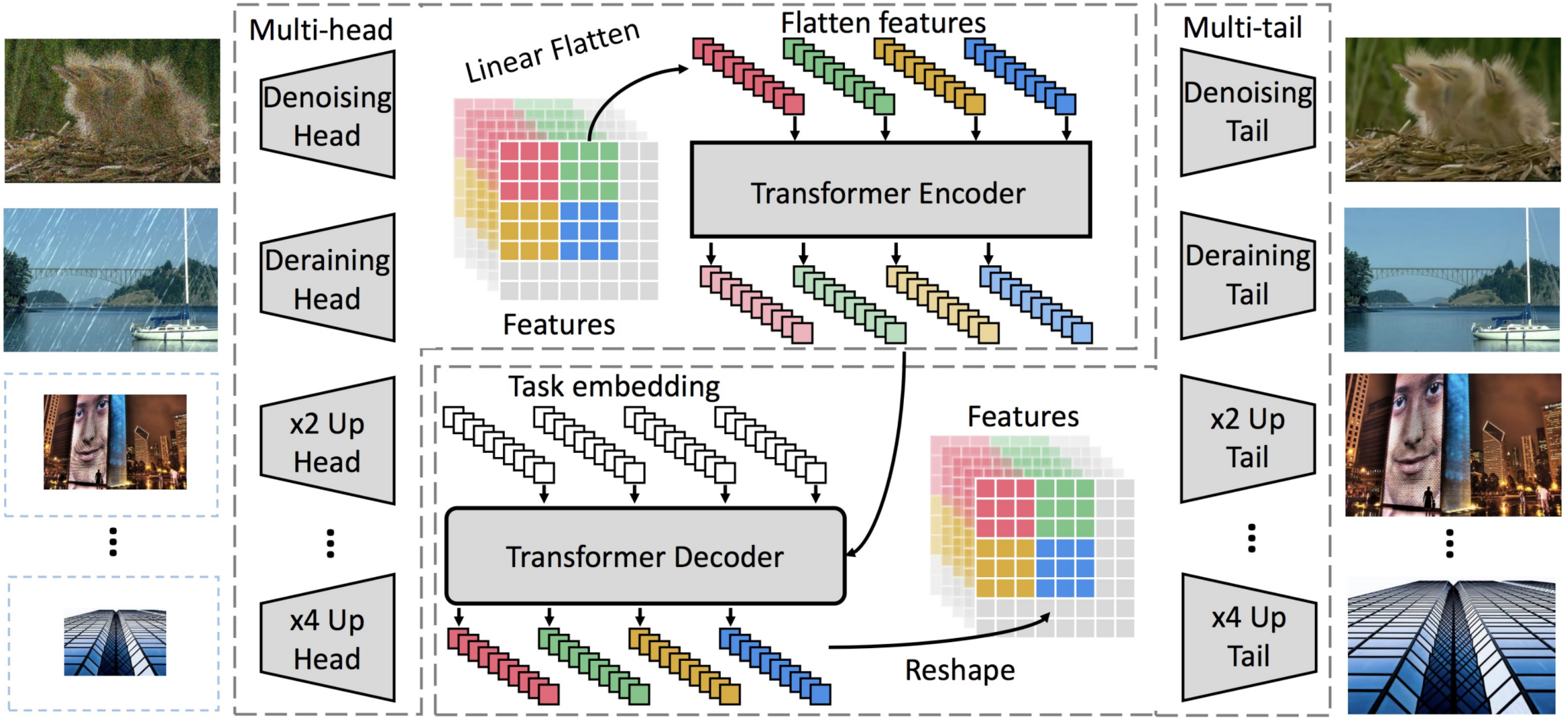


Method	Backbone	MS	PQ	PQ^{Th}	PQ^{St}
Top-down panoptic segmentation methods					
TASCNet [49]	ResNet-50		40.7	47.0	31.0
Panoptic-FPN [44]	ResNet-101		40.9	48.3	29.7
AdaptIS [77]	ResNeXt-101	✓	42.8	53.2	36.7
AUNet [52]	ResNeXt-152		46.5	55.8	32.5
UPSNet [87]	DCN-101 [23]	✓	46.6	53.2	36.7
Li *et al.* [50]	DCN-101 [23]		47.2	53.5	37.7
SpatialFlow [16]	DCN-101 [23]	✓	47.3	53.5	37.9
SOGNet [90]	DCN-101 [23]	✓	47.8	-	-
Bottom-up panoptic segmentation methods					
DeeperLab [89]	Xception-71		34.3	37.5	29.6
SSAP [28]	ResNet-101	✓	36.9	40.1	32.0
Panoptic-DeepLab [19]	Xception-71	✓	41.4	45.1	35.9
Axial-DeepLab-S	Axial-ResNet-S		42.2	46.5	35.7
Axial-DeepLab-M	Axial-ResNet-M		43.2	48.1	35.9
Axial-DeepLab-L	Axial-ResNet-L		43.6	48.9	35.6
Axial-DeepLab-L	Axial-ResNet-L	✓	44.2	49.2	36.8



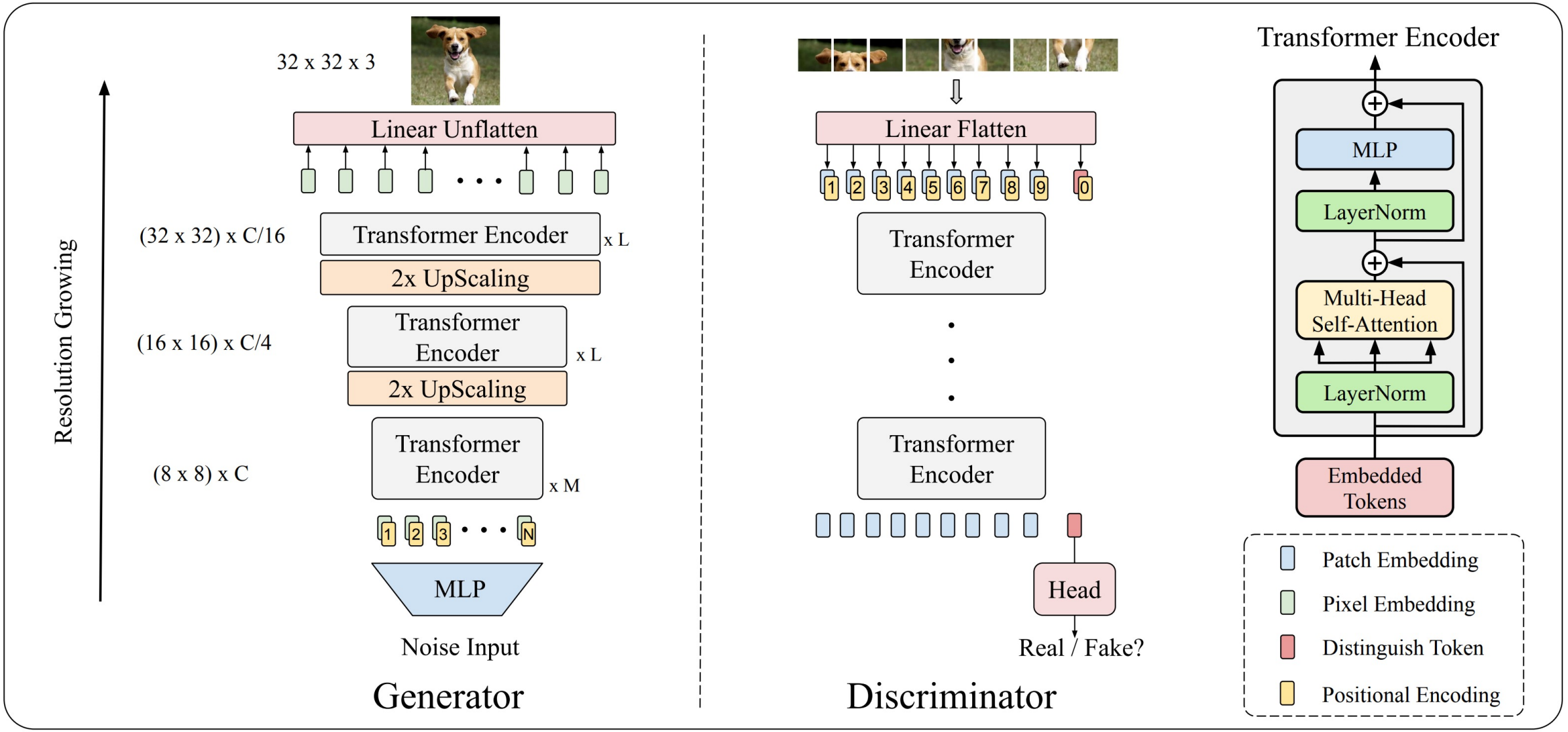
Wang et al., "Axial-DeepLab: Stand-Alone Axial-Attention for Panoptic Segmentation," ECCV, 2020.

Image Processing with Transformers



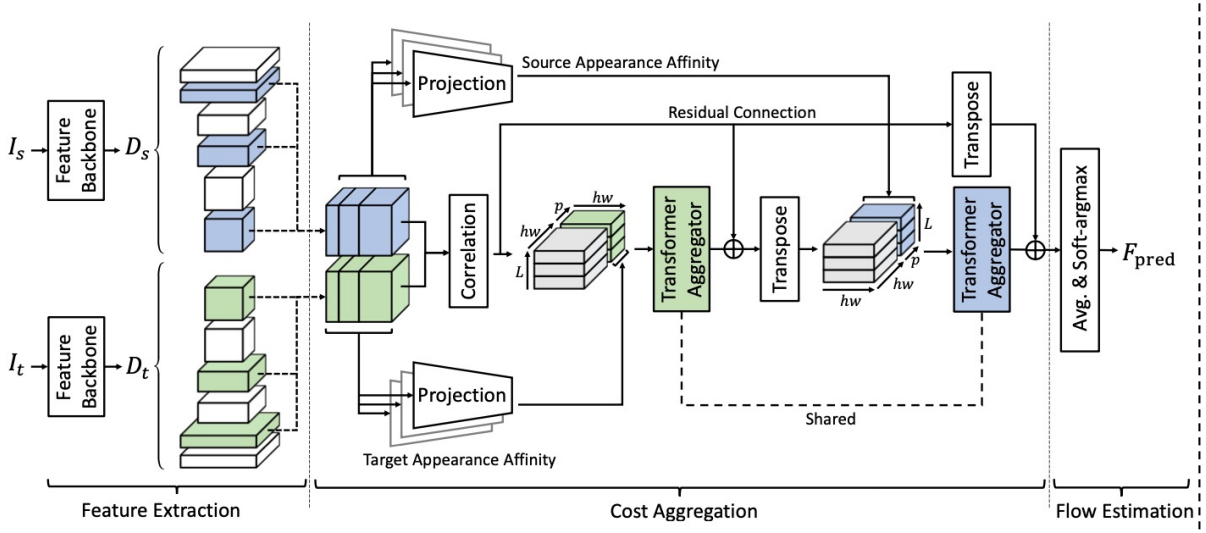
Chen et al., "Pre-Trained Image Processing Transformer," arxiv, 2020.

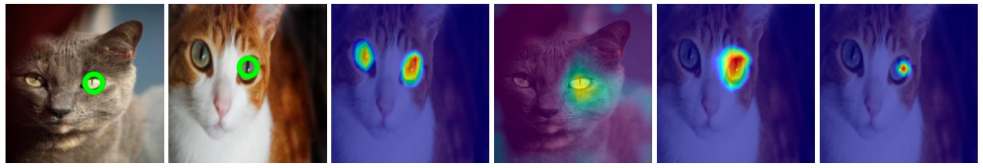
Image Generation with Transformers



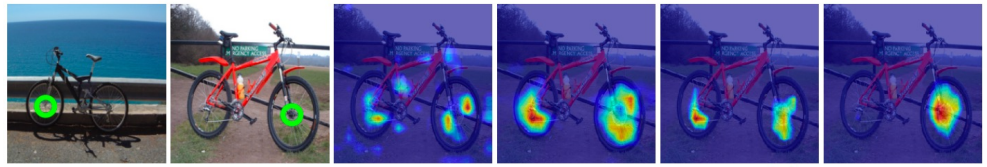
Jiang et al., "TransGAN: Two Transformers Can Make One Strong GAN," arxiv, 2020.

Image Matching with Transformers

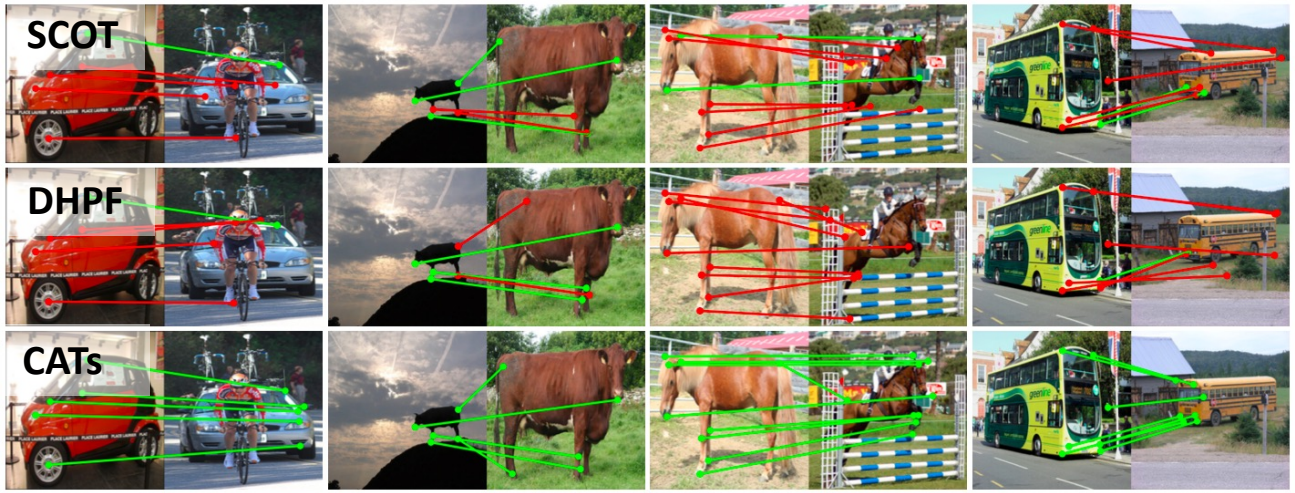




(a) Source (b) Target (c) Raw Corr. (d) Self-Attn. (e) Refined Corr. (f) Ground-truth



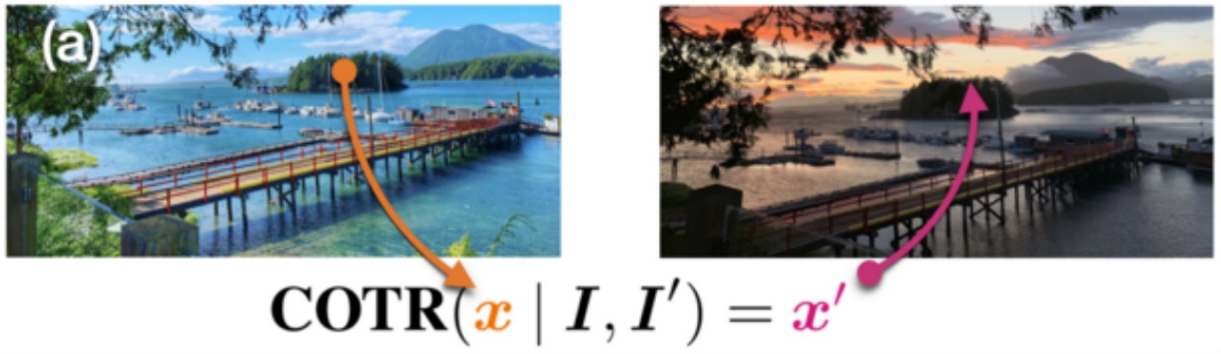
(a) Source (b) Target (c) Corr. #1 (d) Corr. #3 (e) HPF [35] (f) CATs

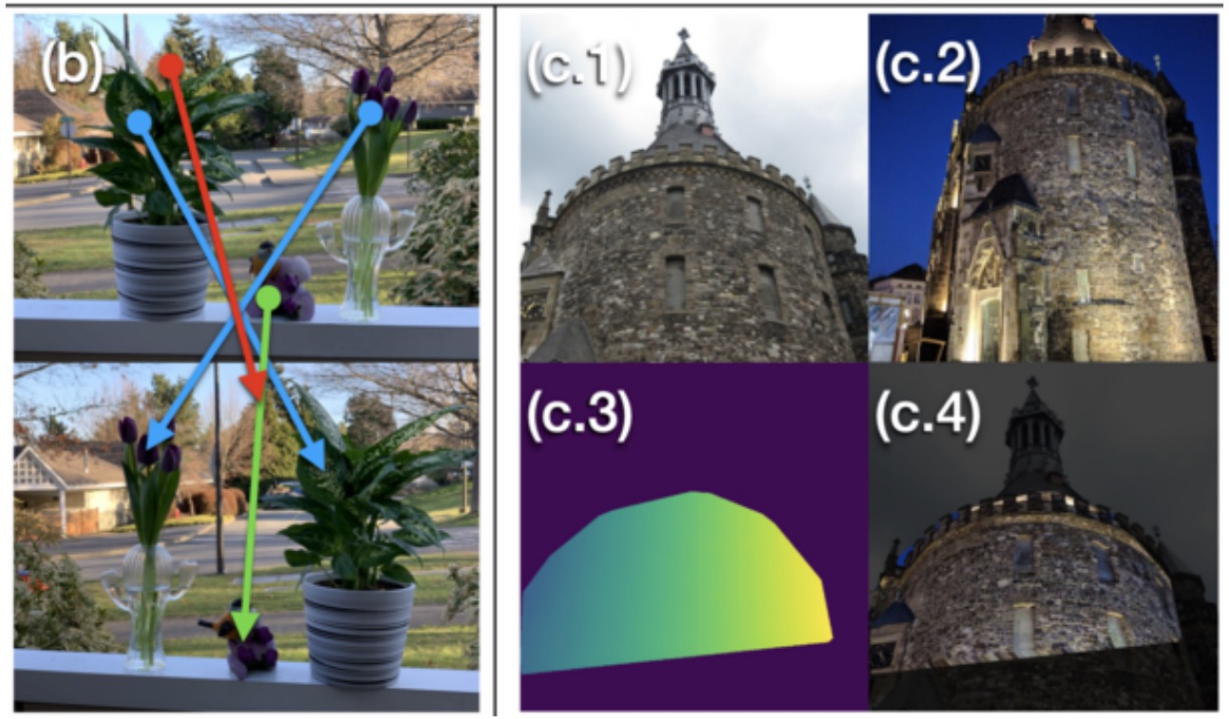


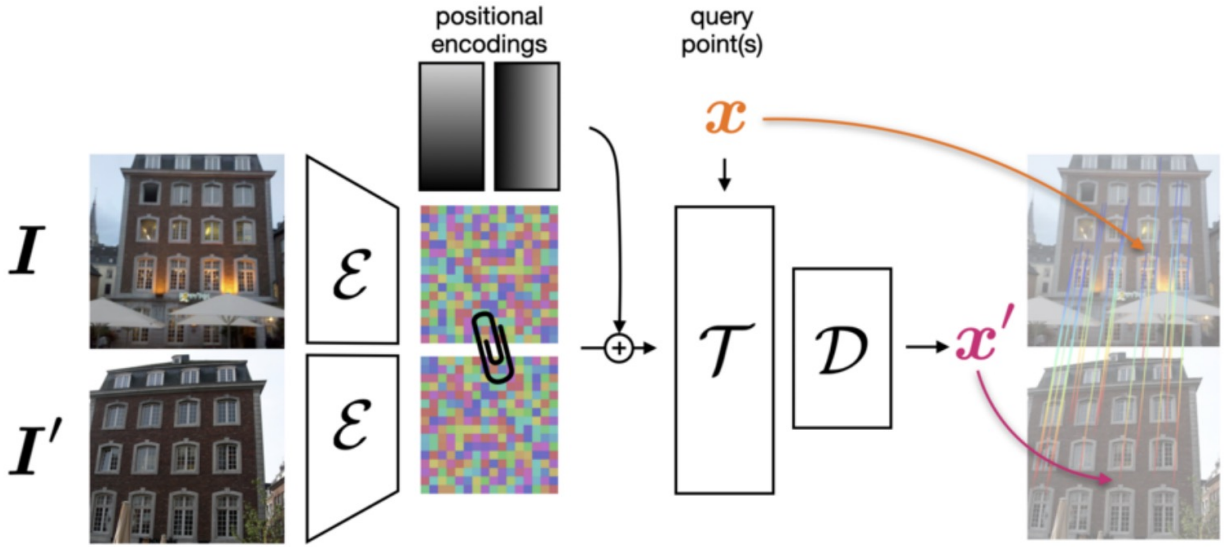
Matching Results

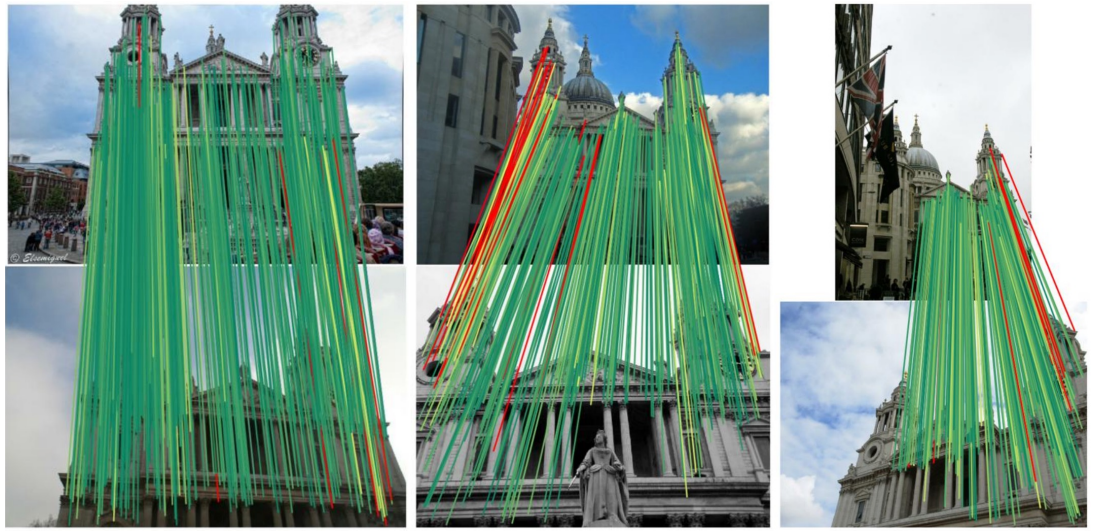
Hong et al., "Semantic Correspondence with Transformers," arxiv, 2021.

Image Matching with Transformers



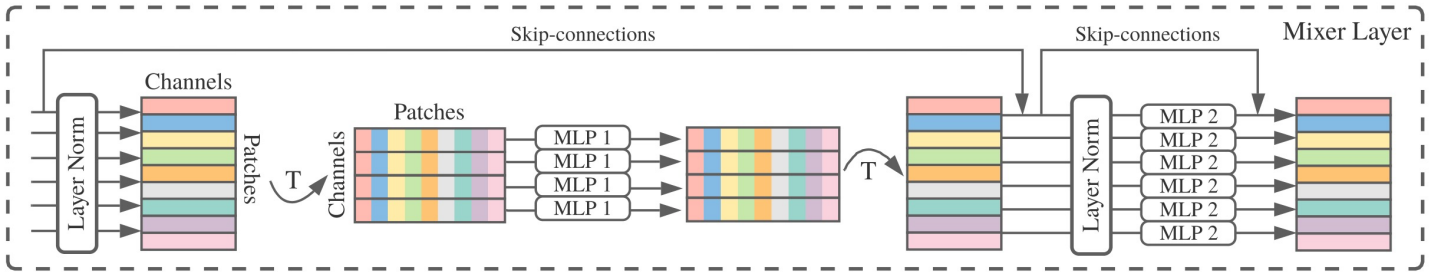


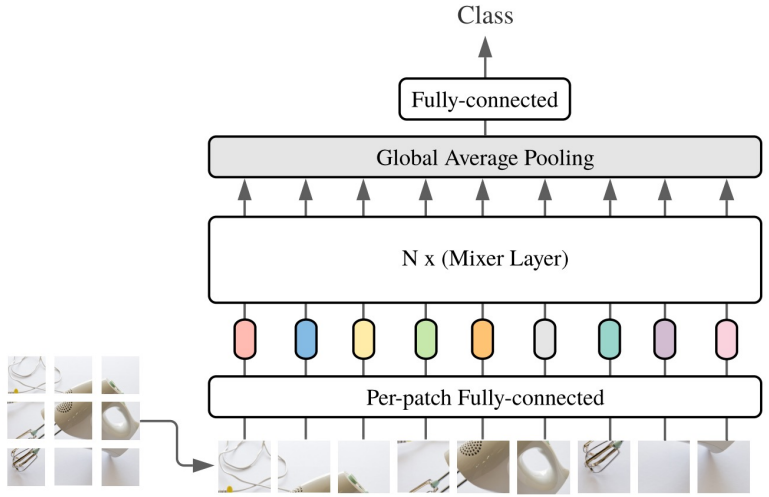


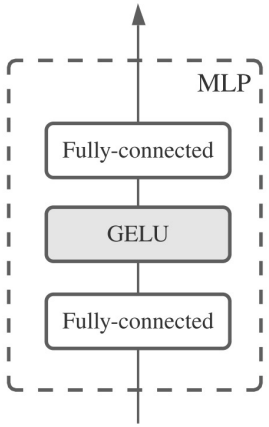


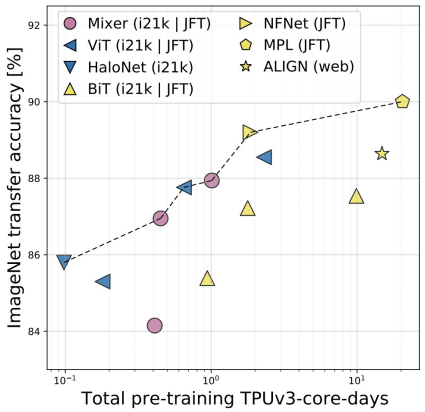
Jiang et al., "COTR: Correspondence Transformer for Matching Across Images," ICCV, 2021.

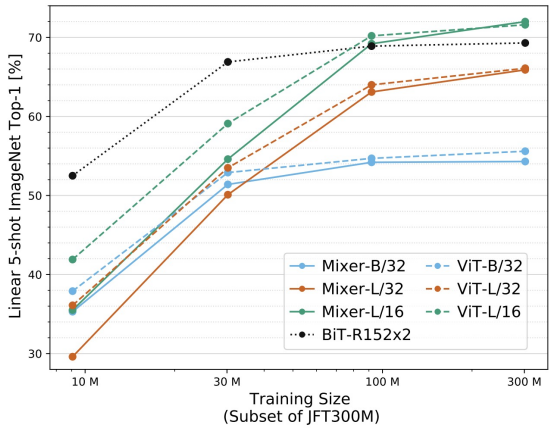
MLP-Mixer











	ImNet top-1	ReaL top-1	Avg 5 top-1	VTAB-1k 19 tasks	Throughput img/sec/core	TPUv3 core-days
Pre-trained on ImageNet-21k (public)						
HaloNet [50]	85.8	—	—	—	120	0.10k
Mixer-L/16	84.15	87.86	93.91	74.95	105	0.41k
ViT-L/16 [14]	85.30	88.62	94.39	72.72	32	0.18k
BiT-R152x4 [22]	85.39	—	94.04	70.64	26	0.94k
Pre-trained on JFT-300M (proprietary)						
NFNet-F4+ [7]	89.2	—	—	—	46	1.86k
Mixer-H/14	87.94	90.18	95.71	75.33	40	1.01k
BiT-R152x4 [22]	87.54	90.54	95.33	76.29	26	9.90k
ViT-H/14 [14]	88.55	90.72	95.97	77.63	15	2.30k
Pre-trained on unlabelled or weakly labelled data (proprietary)						
MPL [34]	90.0	91.12	—	—	—	20.48k
ALIGN [21]	88.64	—	—	79.99	15	14.82k

Tolstikhin et al., “MLP-Mixer: An all-MLP Architecture for Vision,” arxiv, 2021.

Thank you!
Q & A