



mp

max planck institut
informatik

SIC Saarland Informatics
Campus

High Level Computer Vision

Transformer Architecture & Computer Vision

@ May 24, 2023

Bernt Schiele

cms.sic.saarland/hlcvss23/

Max Planck Institute for Informatics & Saarland University,
Saarland Informatics Campus Saarbrücken

Transformer for Computer Vision Tasks

- Many Transformer Based Methods in the last Years... e.g. for
 - Image Classification [Dosovitskiy et al., 2020] [Touvron et al., 2020]
 - Object detection [Carion et al., 2020] [Zhu et al., 2020]
 - Image Segmentation [Ye et al., 2020] [Yie et al., 2021]
 - Image super resolution [Yang et al., 2020]
 - Video understanding [Sun et al., 2019] [Girdhar et al., 2019]
 - Image generation [Chen et al., 2020]
 - Visual question answering [Tan et al., 2019] [Su et al., 2019]
 - ...

Overview of Today's Lecture

- Introduction of Transformer
 - Attention Is All You Need @ NeurIPS 2017 - <https://arxiv.org/abs/1706.03762>
- Vision Transformer (ViT) for Image Classification
 - An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale @ ICLR 2021
<https://arxiv.org/abs/2010.11929>
- Object Detection with Transformers (DETR)
 - DETR: End-to-End Object Detection with Transformers @ ECCV 2020
<https://arxiv.org/abs/2005.12872>
 - Deformable DETR: Deformable Transformers for End-to-End Object Detection @ ICLR 2021
<https://arxiv.org/abs/2010.04159>
- Swin Transformer
 - Swin Transformer: Hierarchical Vision Transformer using Shifted Windows @ ICCV 2021
<https://arxiv.org/abs/2103.14030>

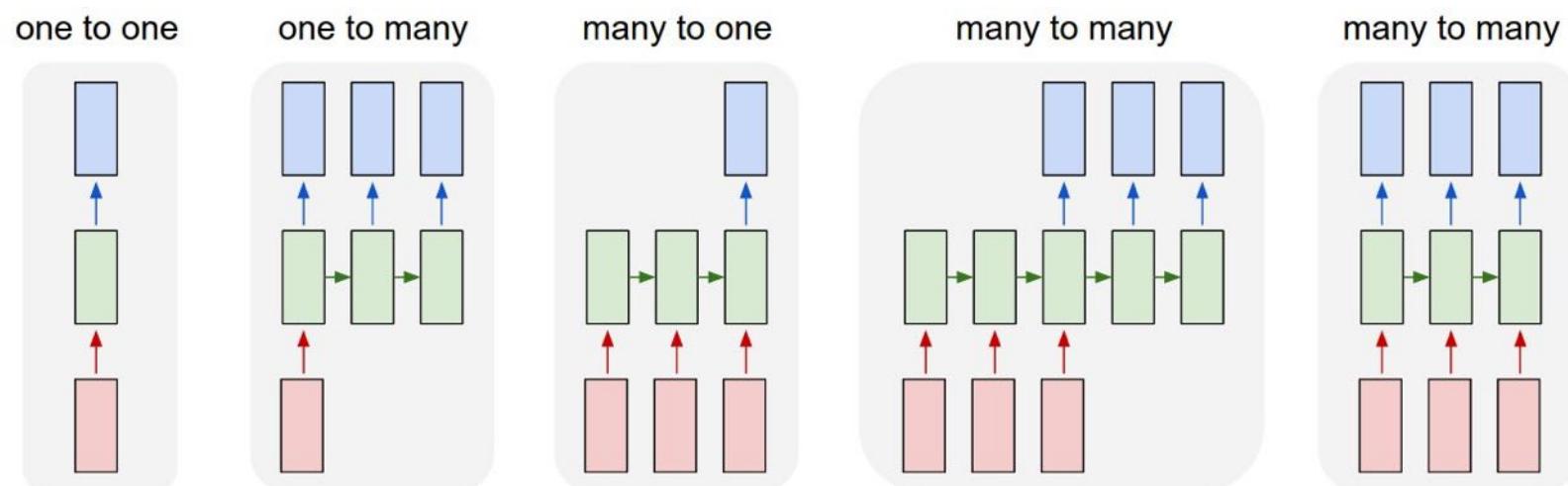
Transformer for Machine Translation



Attention is all you need. Vaswani et al. NeurIPS 2017



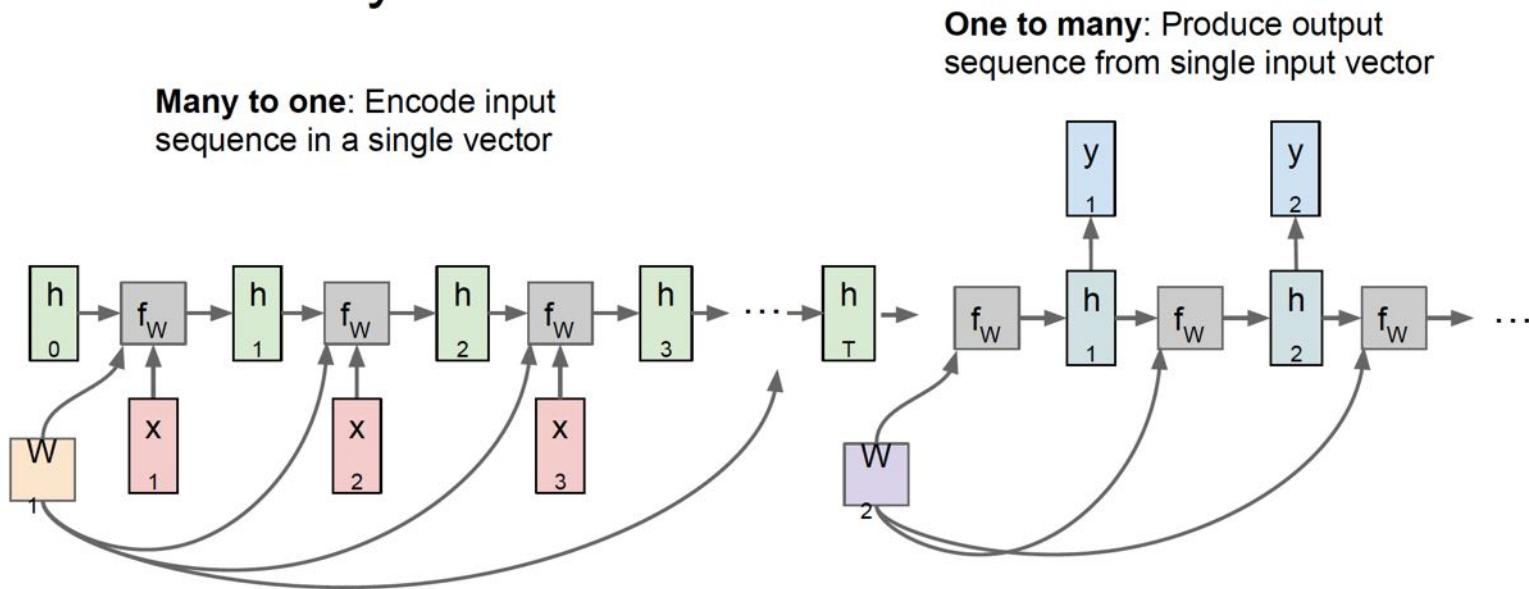
Recurrent Networks offer a lot of flexibility:



slide credit: Andrej Karpathy

Sequence to Sequence (e.g. Language Translation)

Sequence to Sequence: Many-to-one +
one-to-many

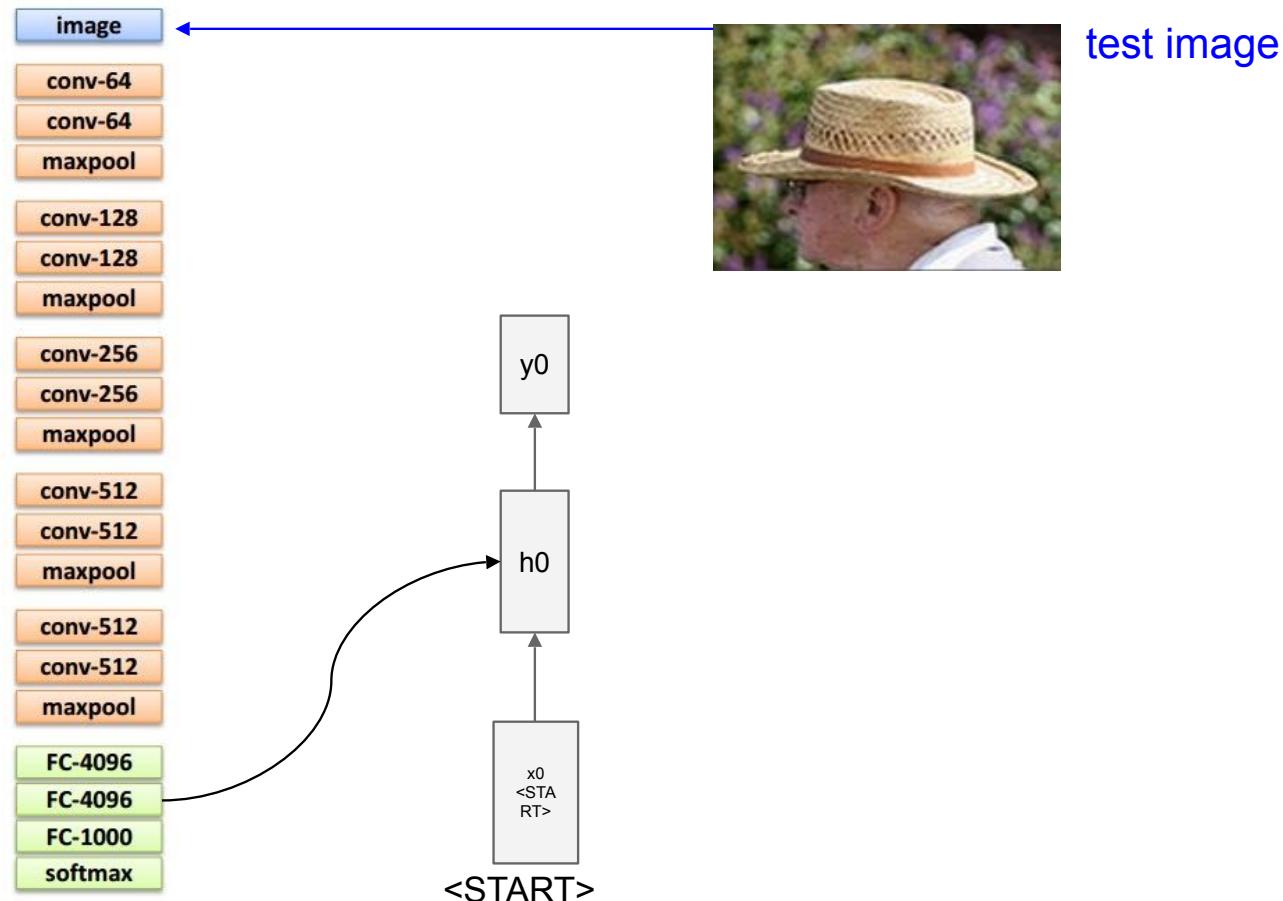


Sutskever et al, "Sequence to Sequence Learning with Neural Networks", NIPS 2014

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

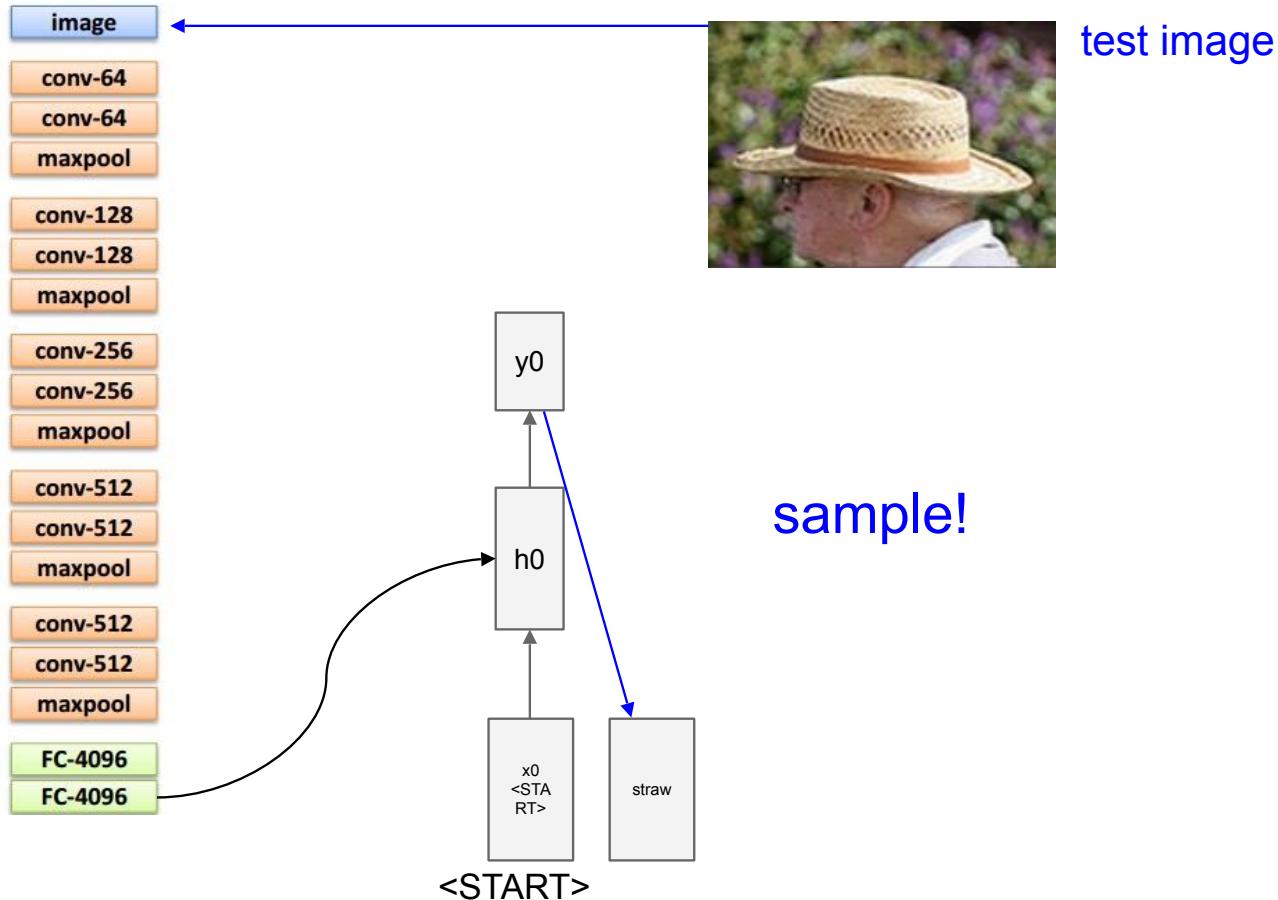


Image Captioning = Image to Sentence Translation



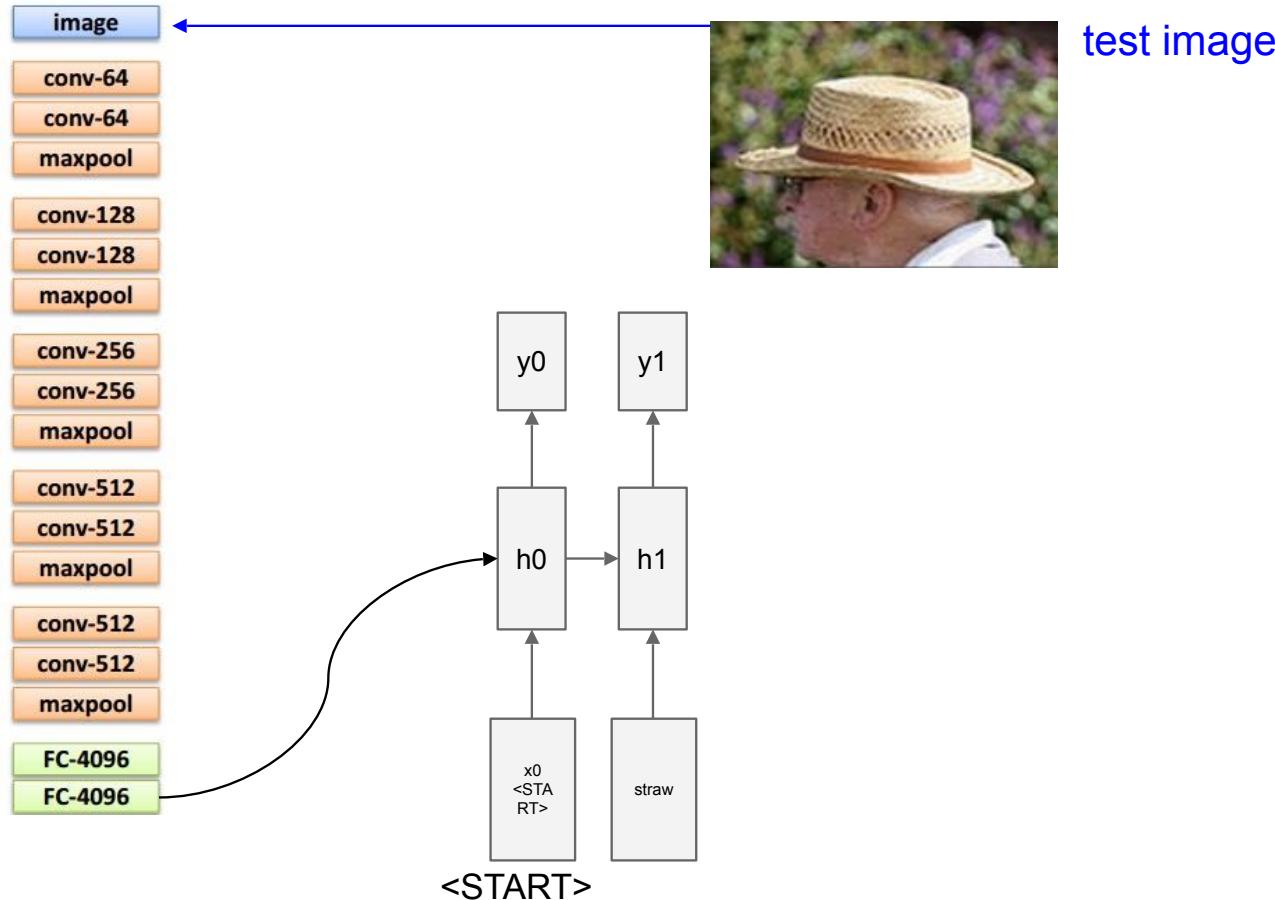
slide credit: Andrej Karpathy

Image Captioning = Image to Sentence Translation



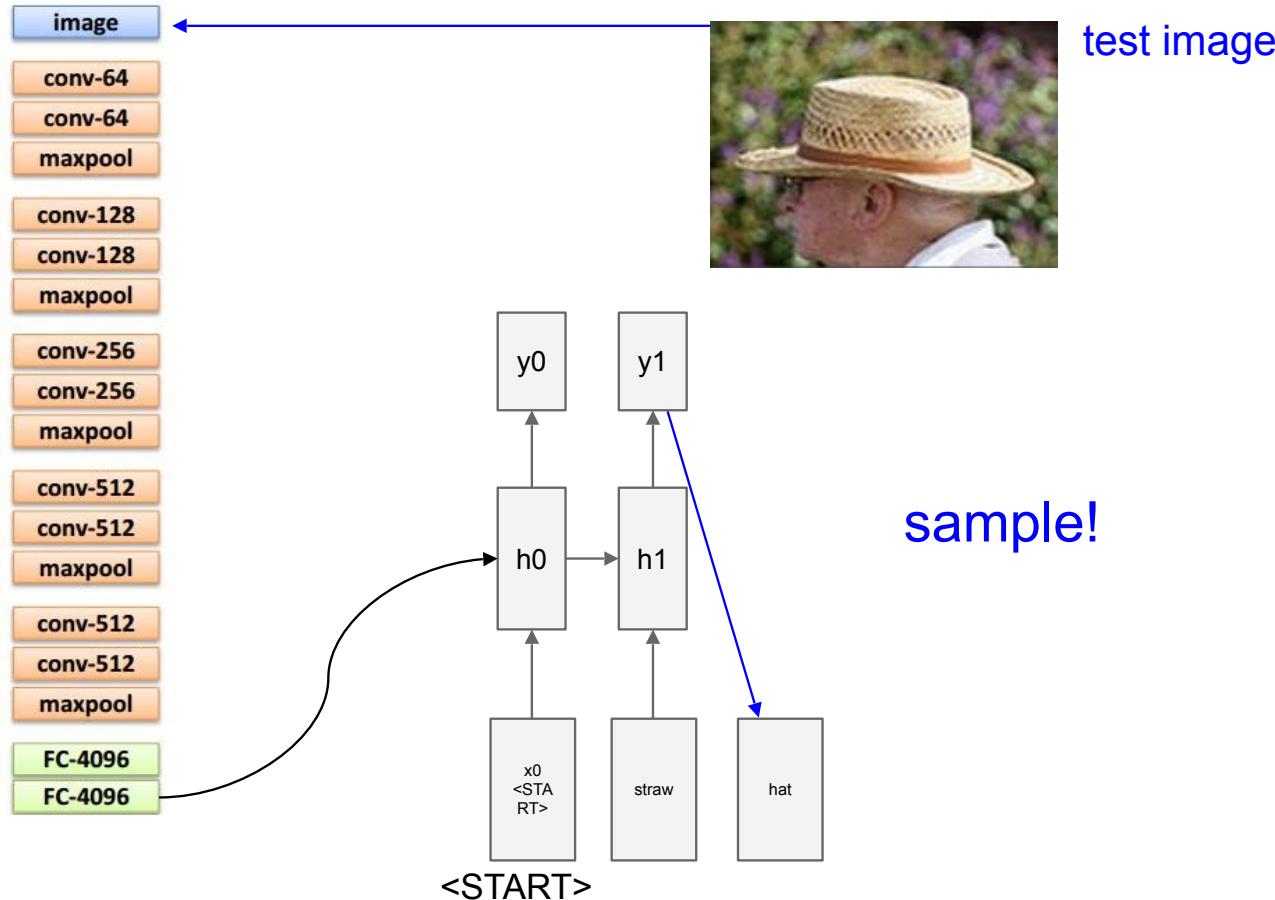
slide credit: Andrej Karpathy

Image Captioning = Image to Sentence Translation



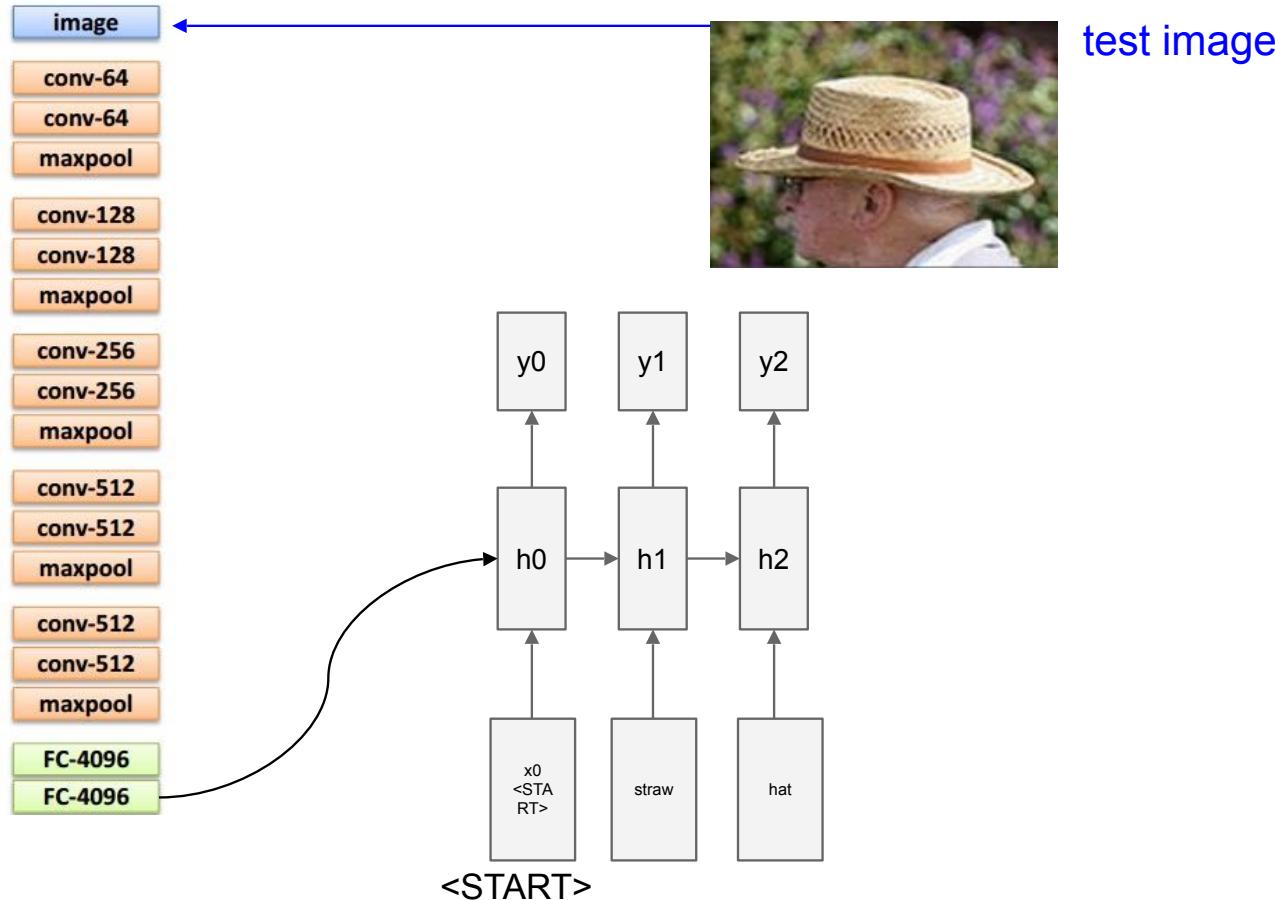
slide credit: Andrej Karpathy

Image Captioning = Image to Sentence Translation



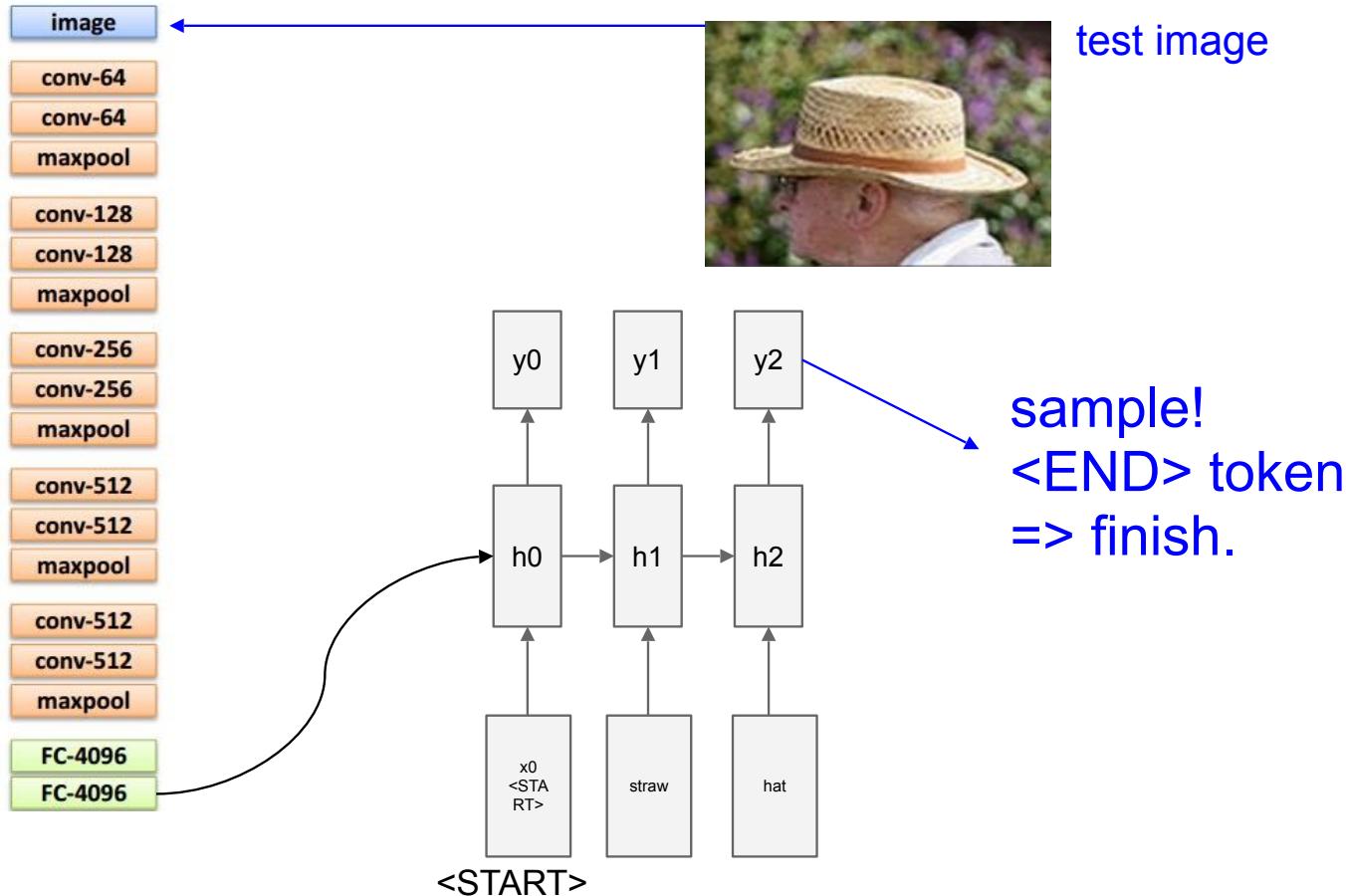
slide credit: Andrej Karpathy

Image Captioning = Image to Sentence Translation



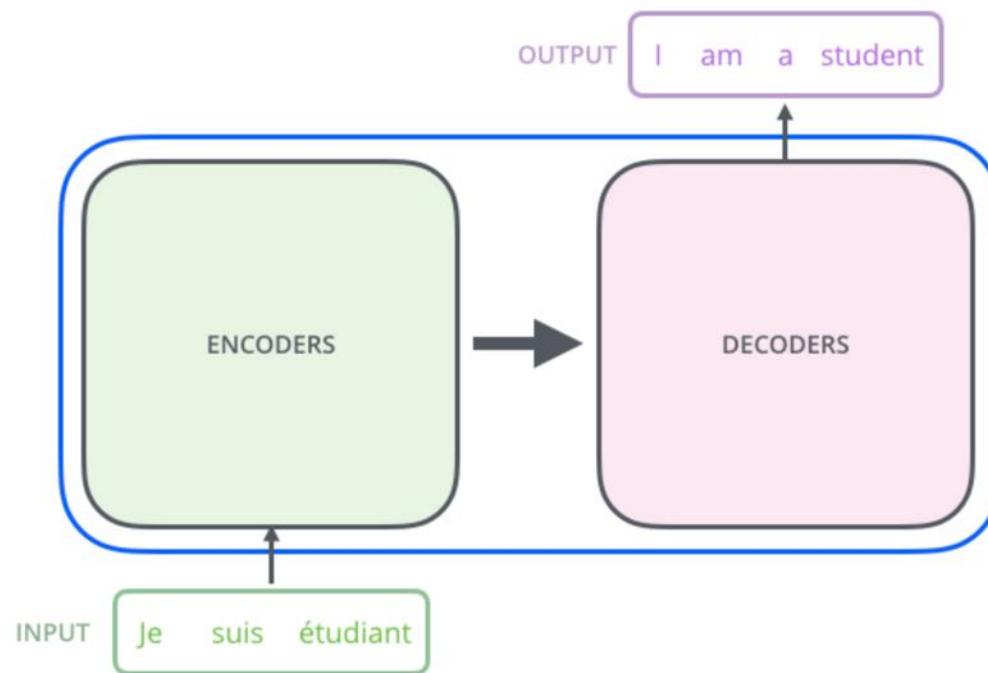
slide credit: Andrej Karpathy

Image Captioning = Image to Sentence Translation



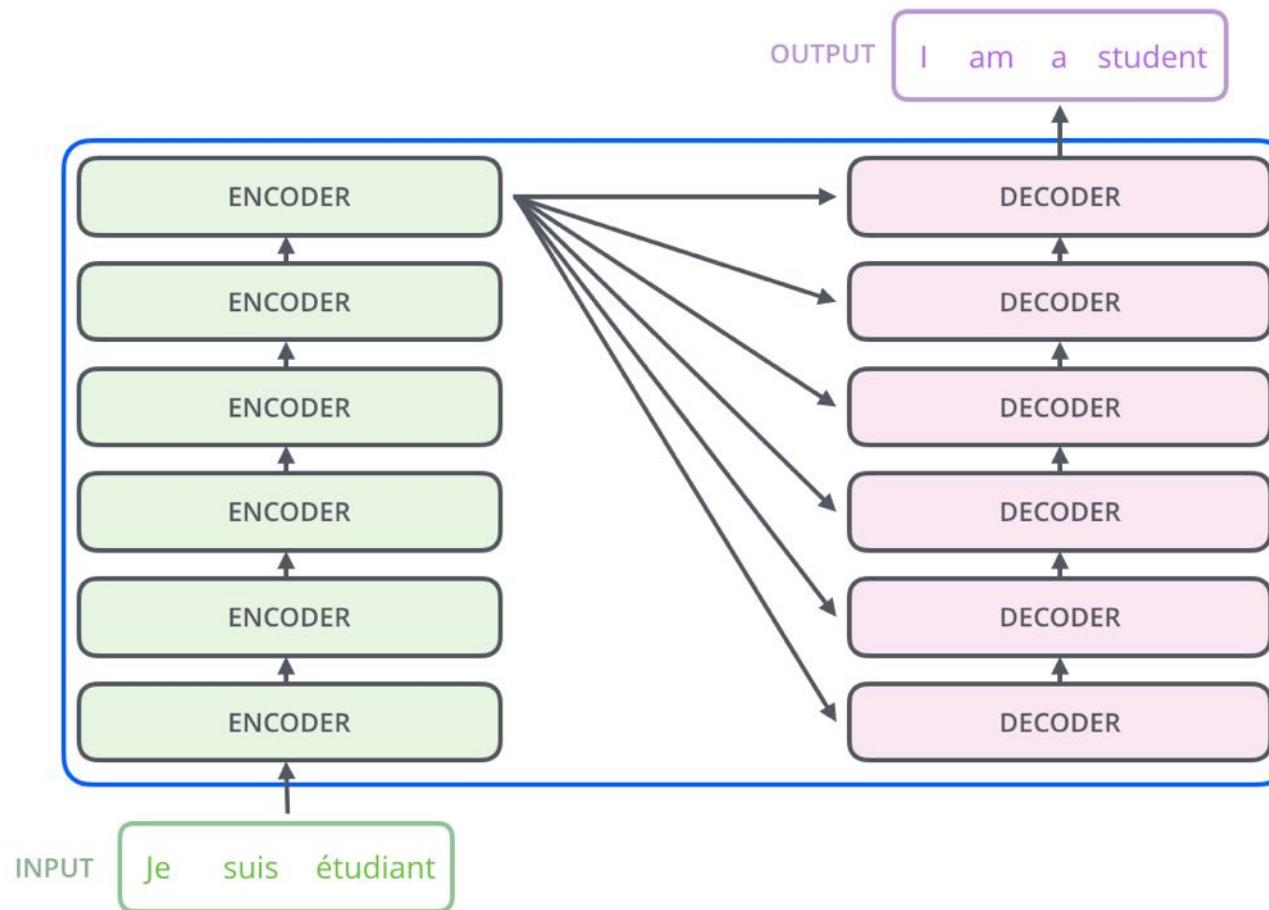
slide credit: Andrej Karpathy

Transformer: Architecture Overview



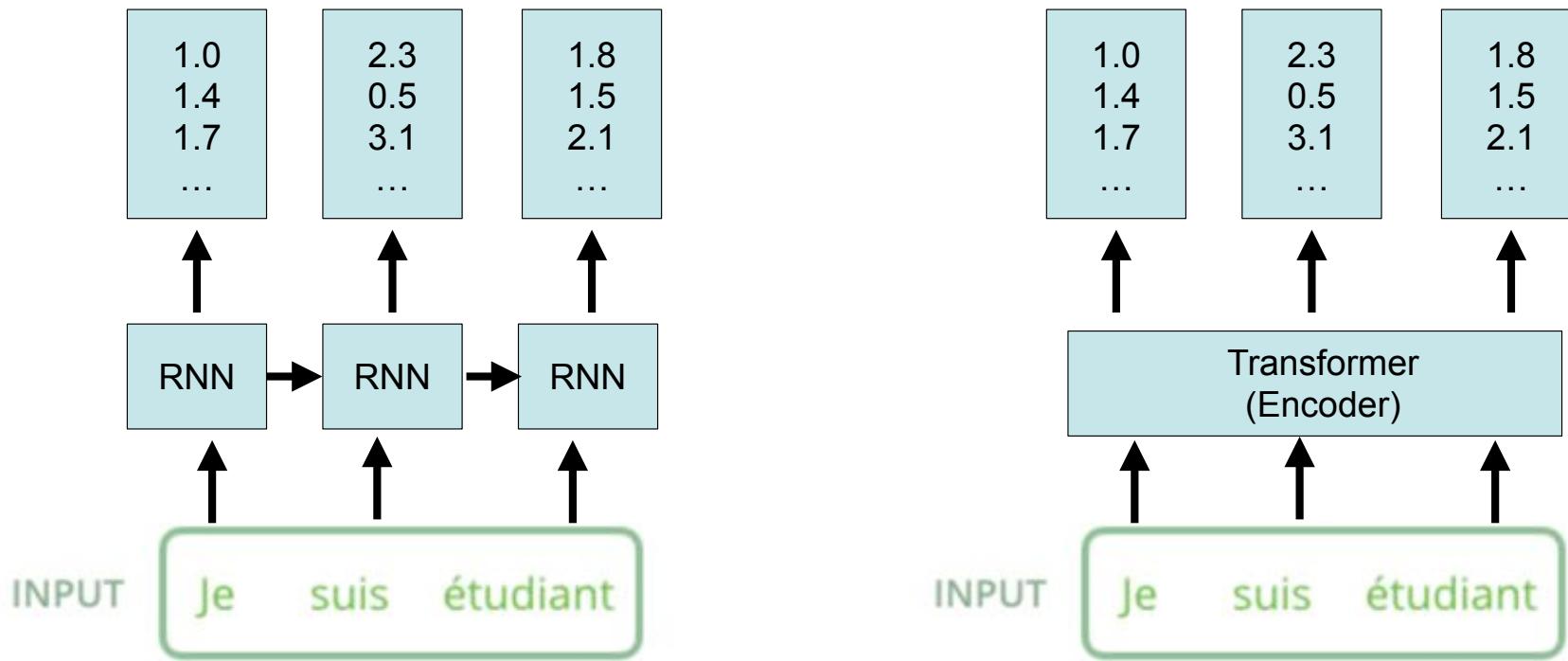
Attention is all you need. Vaswani et al. NeurIPS 2017

Transformer: Architecture Overview



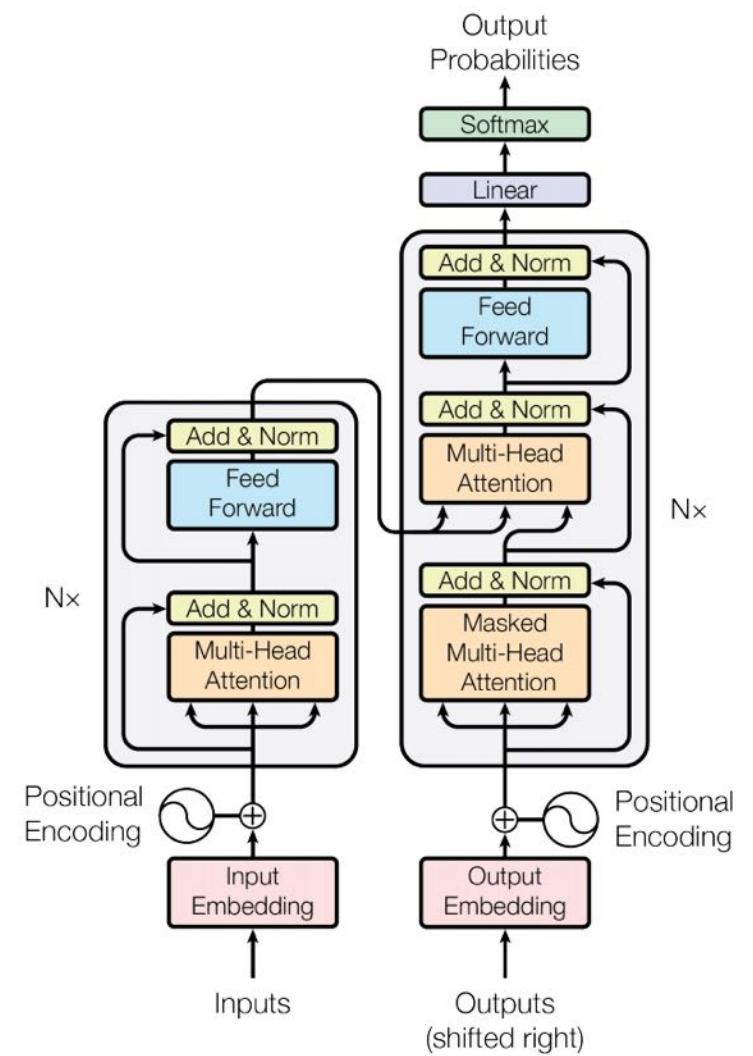
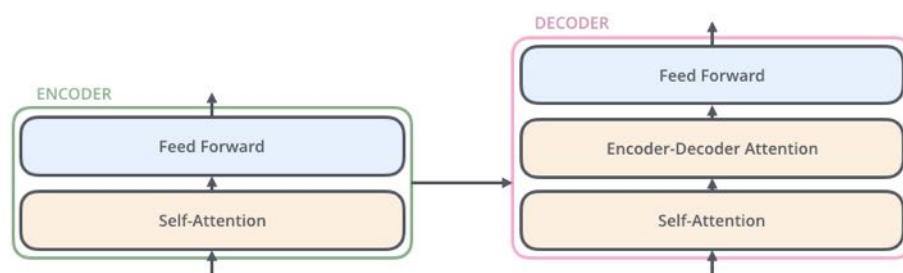
Encoder — Difference RNN vs. Transformer Encoder

- Transformer Language: here, each word is a “**token**”



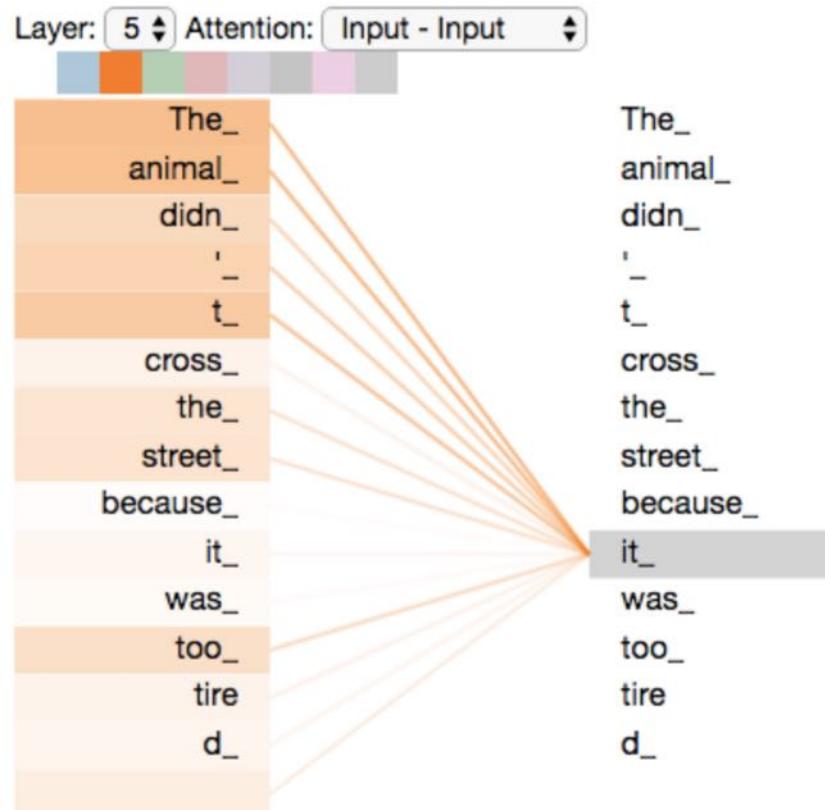
Attention is all you need. Vaswani et al. NeurIPS 2017

Transformer: Architecture Overview



Attention is all you need. Vaswani et al. NeurIPS 2017

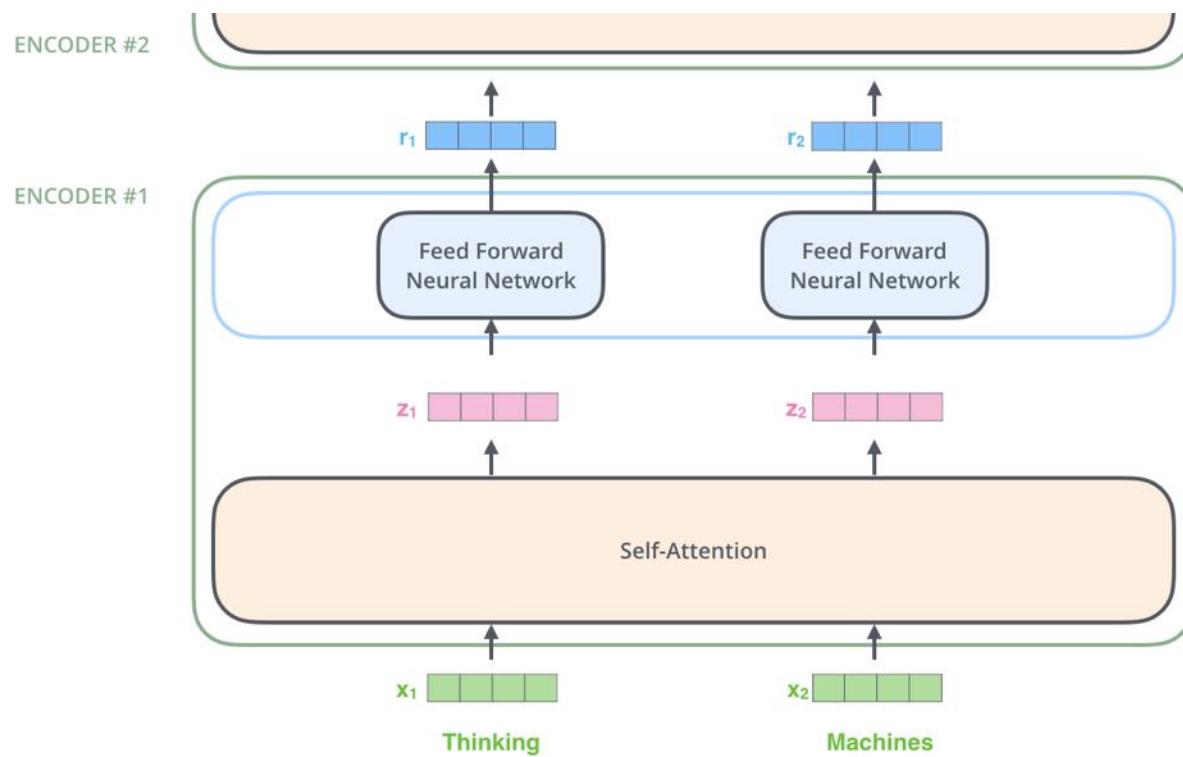
What is Self-Attention?



Attention is all you need. Vaswani et al. NeurIPS 2017



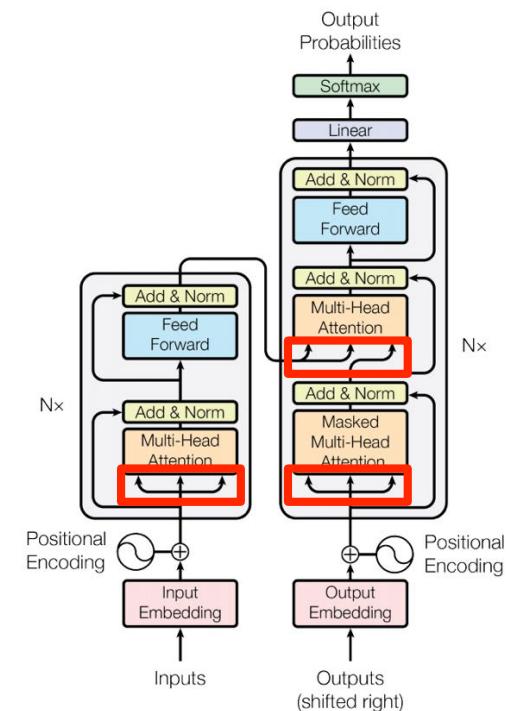
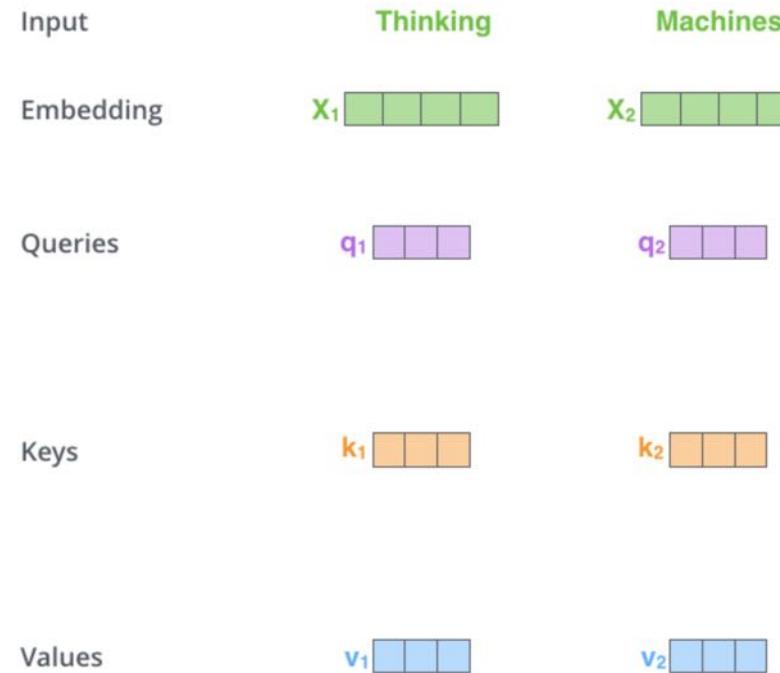
Encoder — Consumes a Set of (Word) Tokens



Attention is all you need. Vaswani et al. NeurIPS 2017



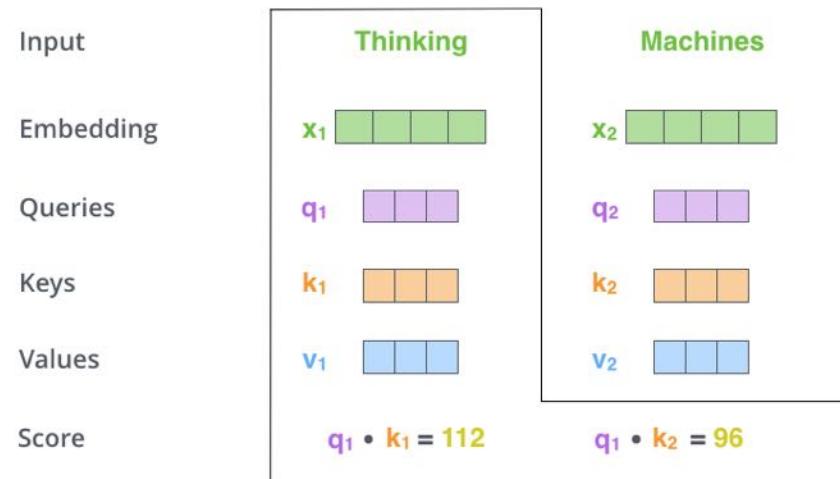
Self-Attention: Query, Key and Value



Attention is all you need. Vaswani et al. NeurIPS 2017

slide credit: Yongqin Xian

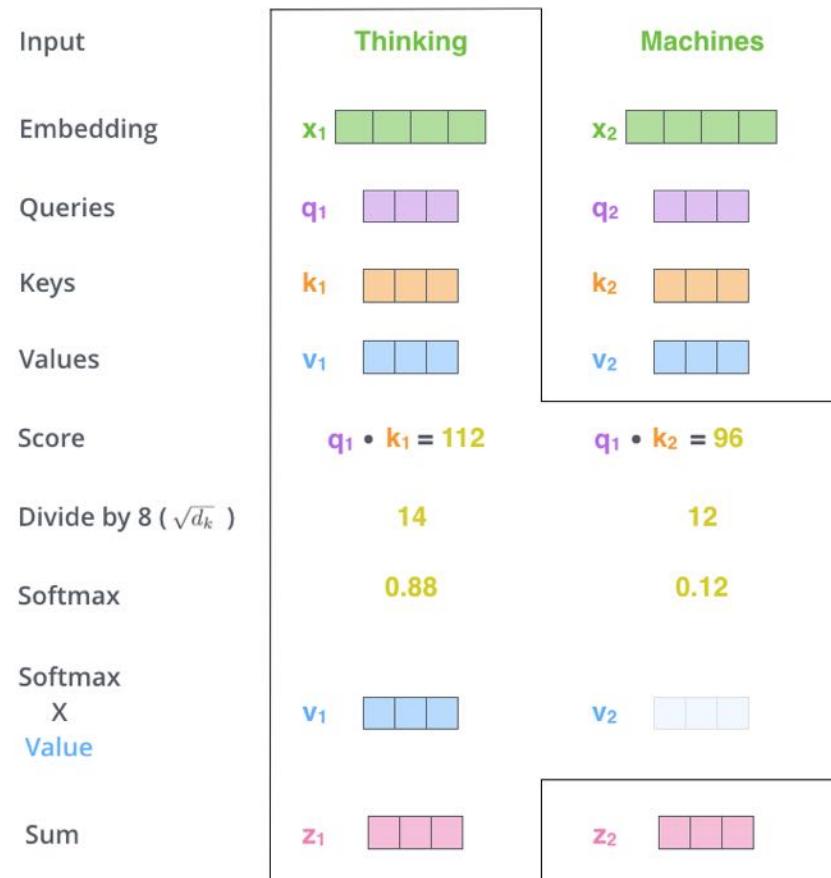
Self-Attention: Query, Key and Value



slide credit: Yongqin Xian



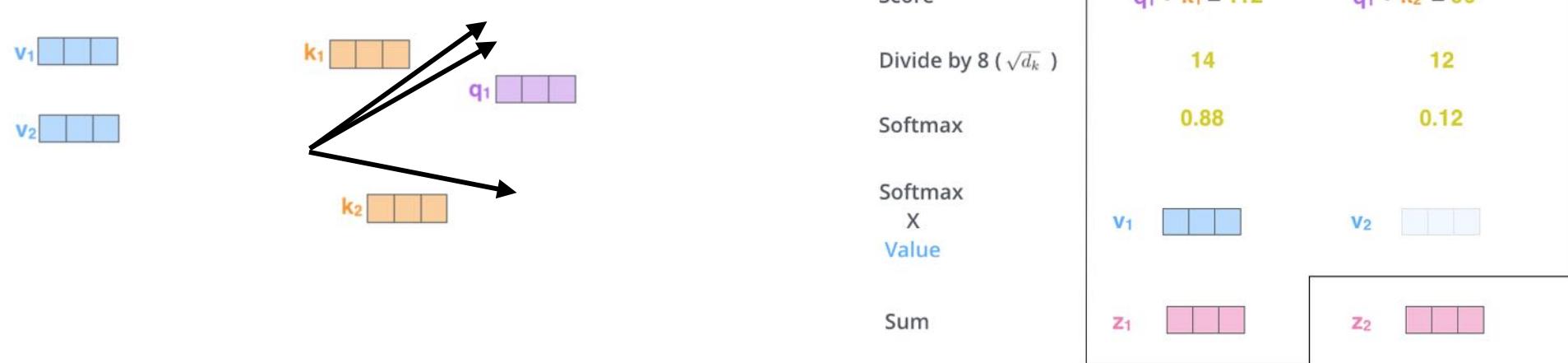
Self-Attention: Query, Key and Value



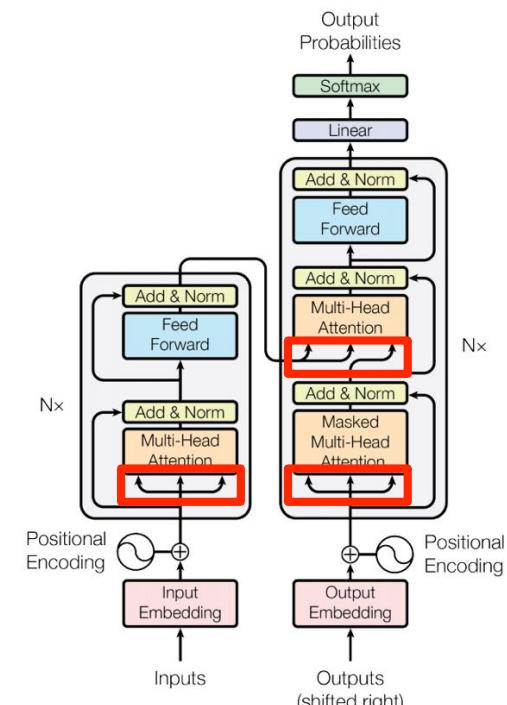
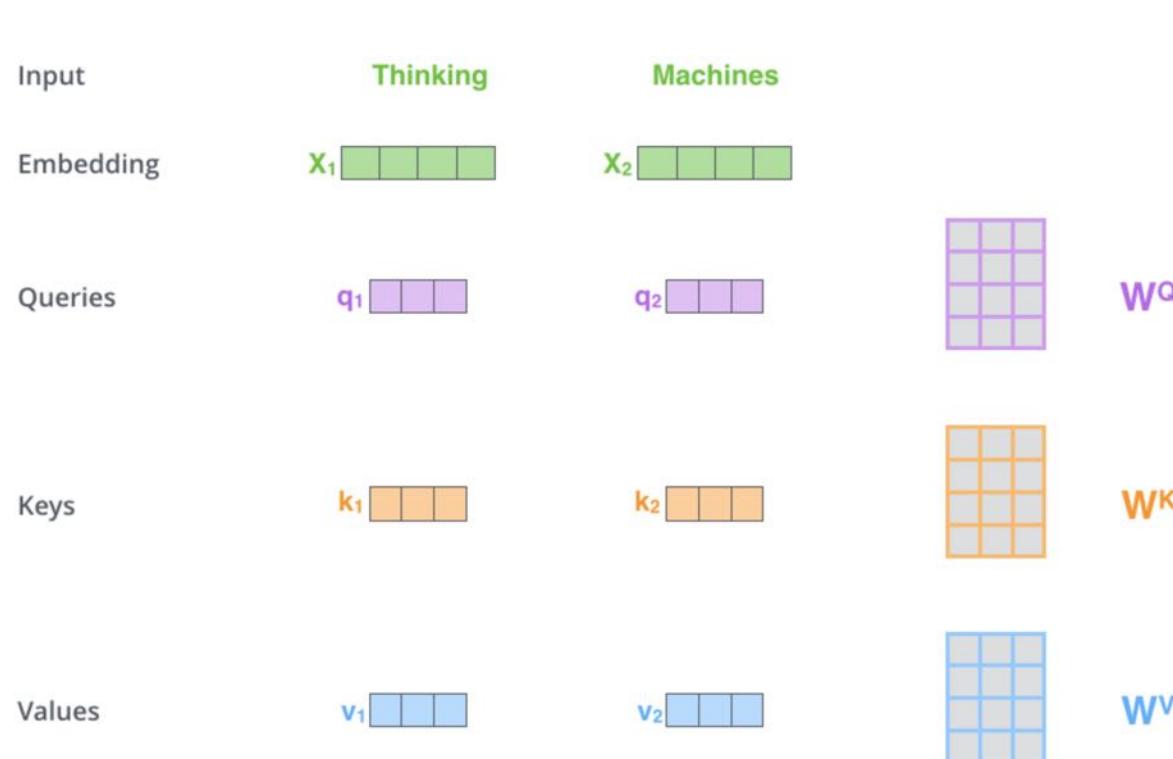
slide credit: Yongqin Xian



Self-Attention: Query, Key and Value



Self-Attention: Query, Key and Value



Attention is all you need. Vaswani et al. NeurIPS 2017

slide credit: Yongqin Xian

Self-Attention: Query, Key and Value

$$\begin{array}{ccc} \mathbf{X} & \mathbf{W^Q} & \mathbf{Q} \\ \text{Thinking} \\ \text{Machines} & \times & \begin{matrix} \text{purple grid} \end{matrix} \\ & & = \\ & & \begin{matrix} \text{purple grid} \end{matrix} \end{array}$$
$$\begin{array}{ccc} \mathbf{X} & \mathbf{W^K} & \mathbf{K} \\ \text{Thinking} \\ \text{Machines} & \times & \begin{matrix} \text{orange grid} \end{matrix} \\ & & = \\ & & \begin{matrix} \text{orange grid} \end{matrix} \end{array}$$
$$\begin{array}{ccc} \mathbf{X} & \mathbf{W^V} & \mathbf{V} \\ \text{Thinking} \\ \text{Machines} & \times & \begin{matrix} \text{blue grid} \end{matrix} \\ & & = \\ & & \begin{matrix} \text{blue grid} \end{matrix} \end{array}$$

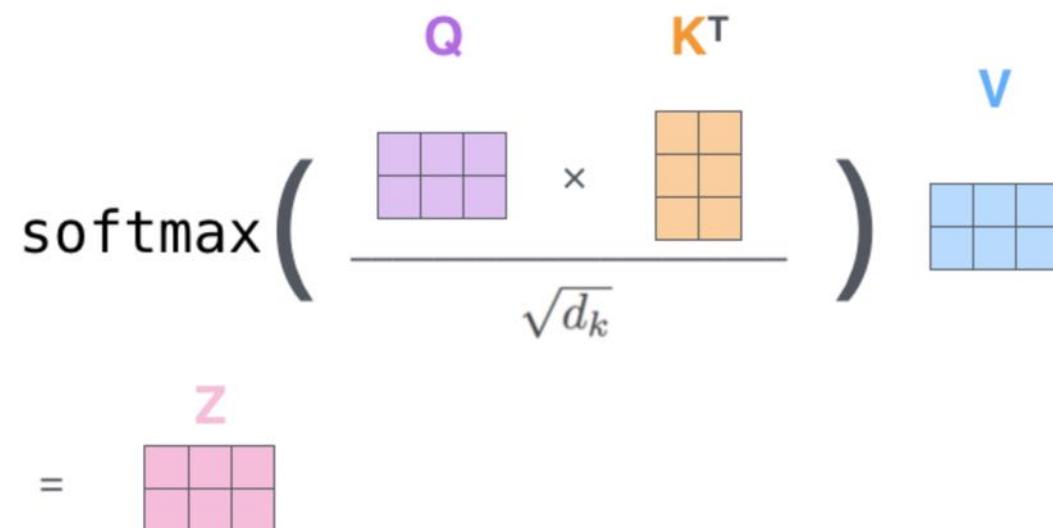
Attention is all you need. Vaswani et al. NeurIPS 2017

slide credit: Yongqin Xian

Self-Attention: Output

$$\text{softmax}\left(\frac{\text{Q} \times \text{K}^T}{\sqrt{d_k}}\right) \text{V}$$

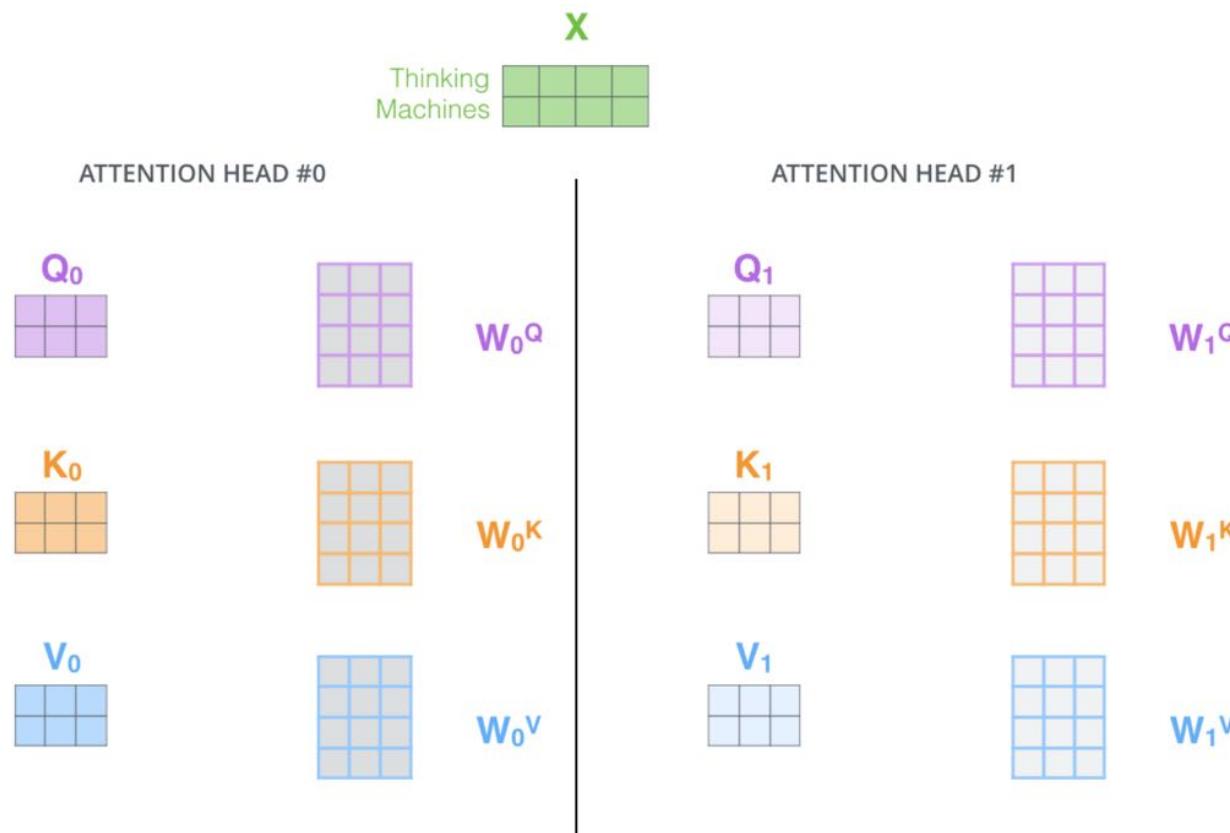
= Z



Attention is all you need. Vaswani et al. NeurIPS 2017

slide credit: Yongqin Xian

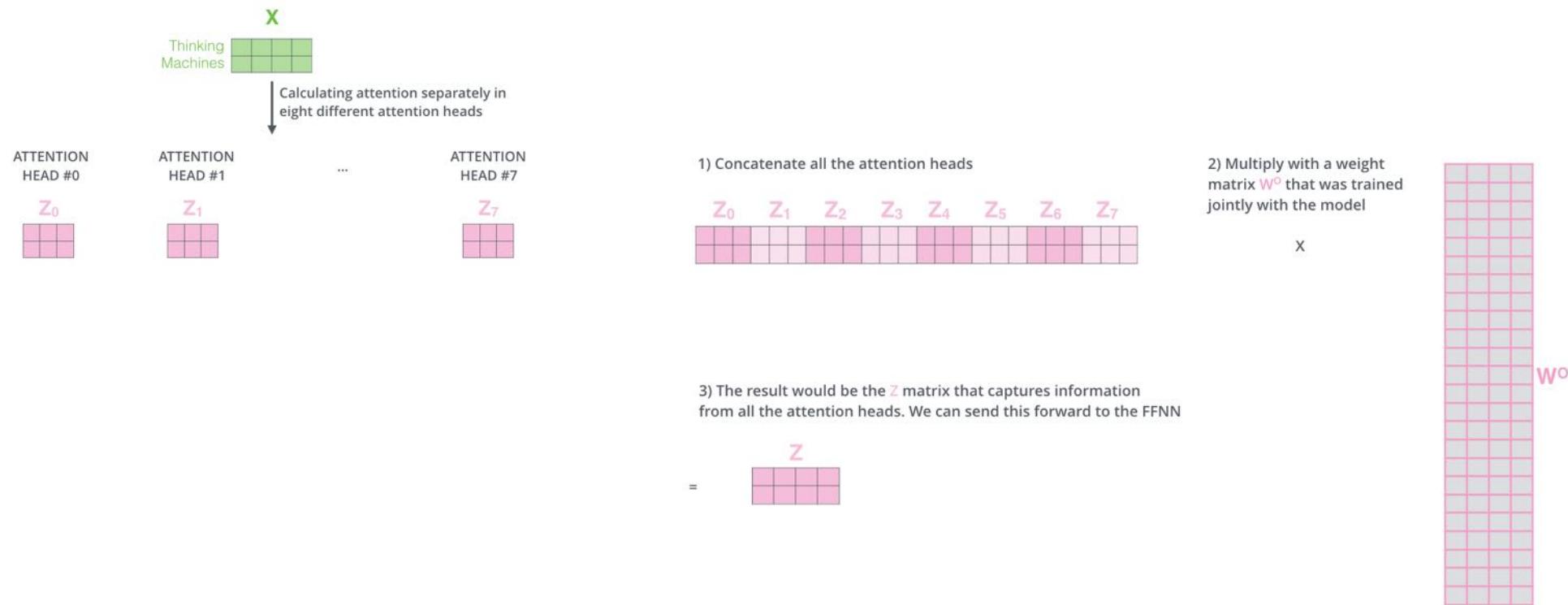
Multi-Head Self-Attention



Attention is all you need. Vaswani et al. NeurIPS 2017

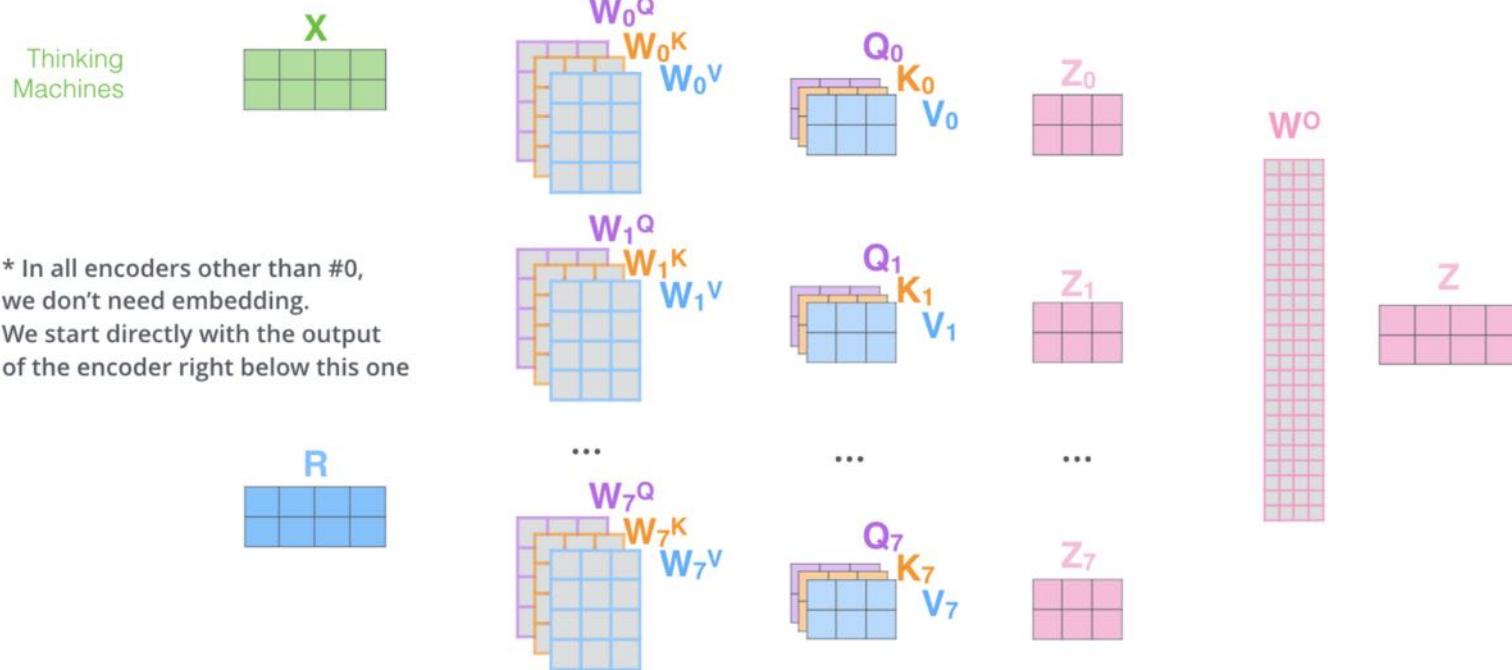
slide credit: Yongqin Xian

Multi-Head Self-Attention



Multi-Head Self-Attention

- 1) This is our input sentence*
- 2) We embed each word*
- 3) Split into 8 heads. We multiply X or R with weight matrices
- 4) Calculate attention using the resulting $Q/K/V$ matrices
- 5) Concatenate the resulting Z matrices, then multiply with weight matrix W^O to produce the output of the layer

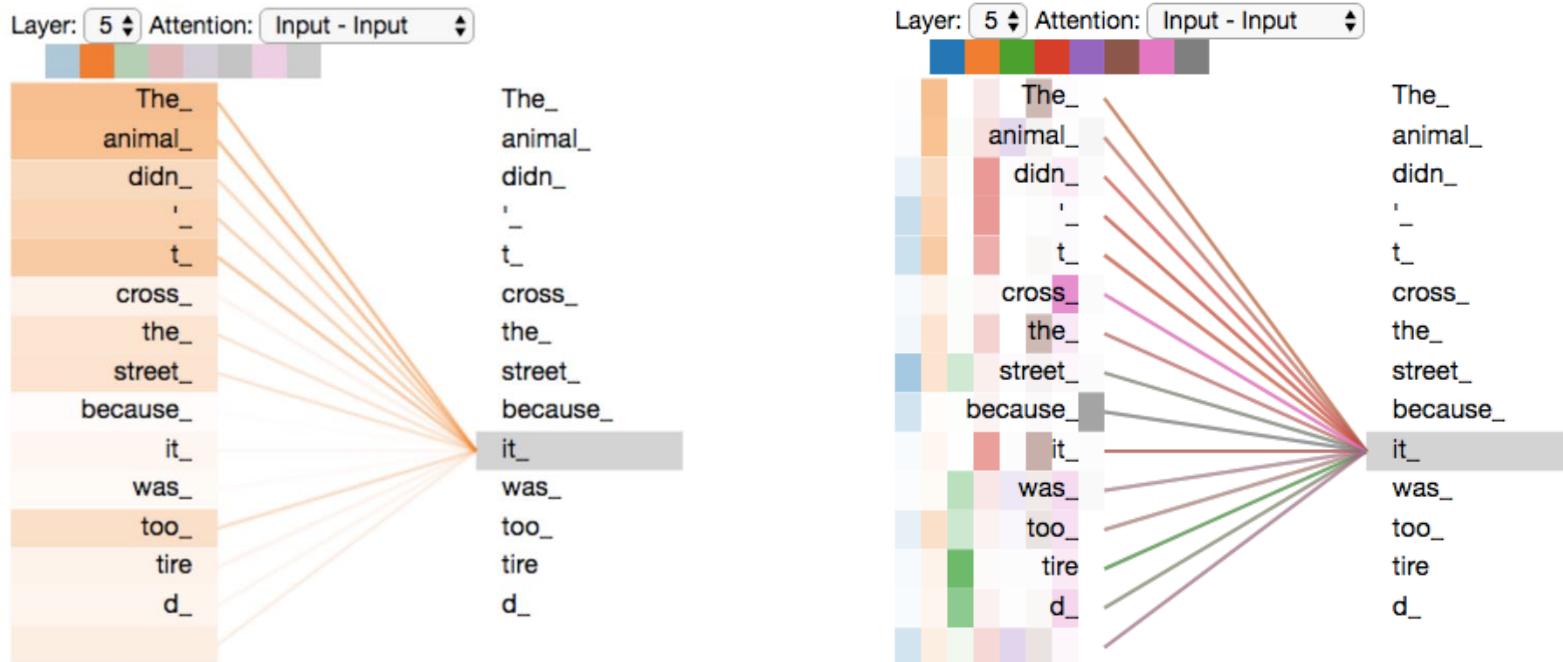


Attention is all you need. Vaswani et al. NeurIPS 2017

slide credit: Yongqin Xian

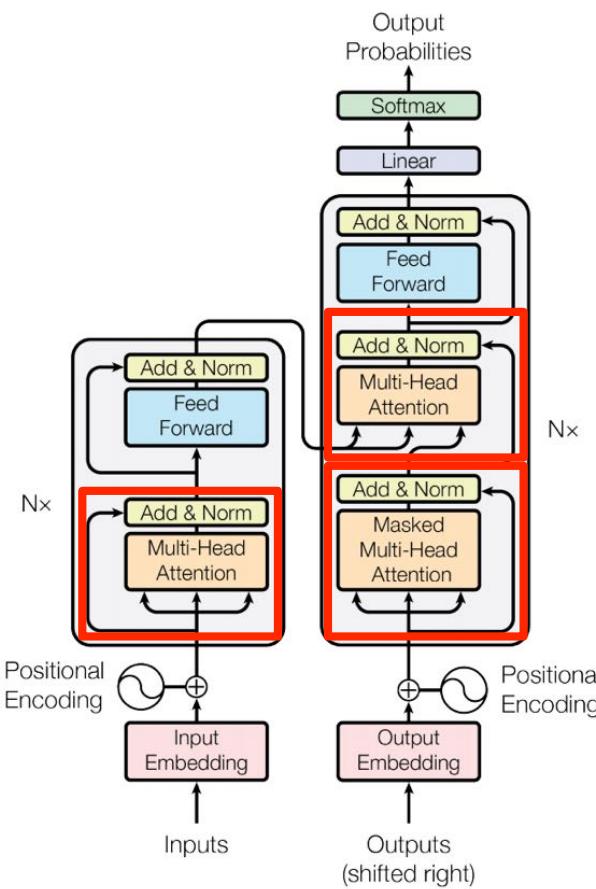
Multi-Head Self-Attention — How to Interpret?

- One head vs. multi-head... hard to interpret



Attention is all you need. Vaswani et al. NeurIPS 2017

Transformer Architecture — Aspects



- Residual connections (add & norm)
- Three Attention Mechanisms:
 - Encoder: Multi-Head (Self) Attention
 - see previous slides
 - (Encoder-Decoder) Multi-Head Attention
 - mechanism identical to the above
 - keys & values coming from encoder
 - queries coming from decoder
 - Decoder: Masked Multi-Head (Self) Attention
 - mechanism identical to the above
 - previous words are given as “queries”
 - future words are “masked”

Illustration of Masking on the Decoder Side

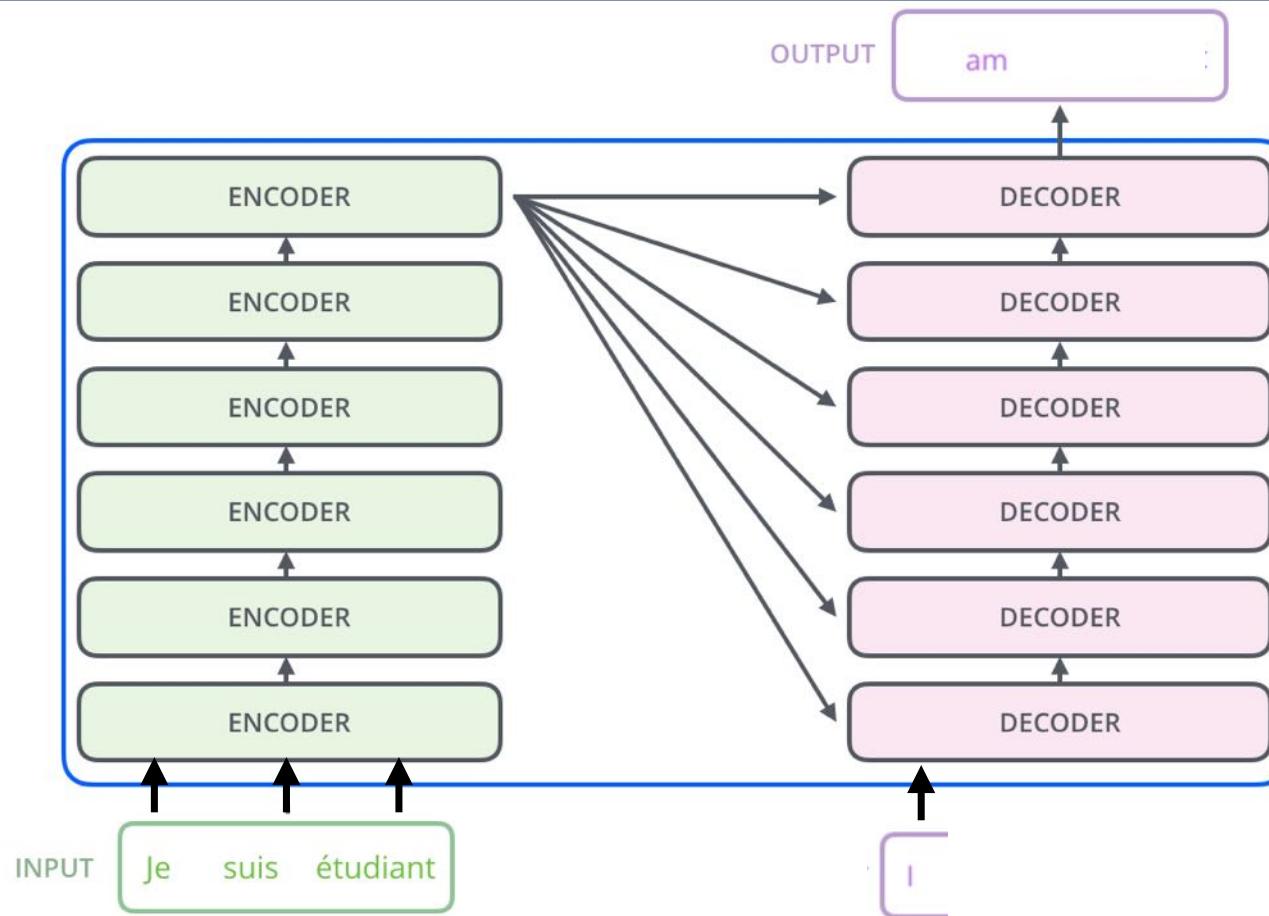


Illustration of Masking on the Decoder Side

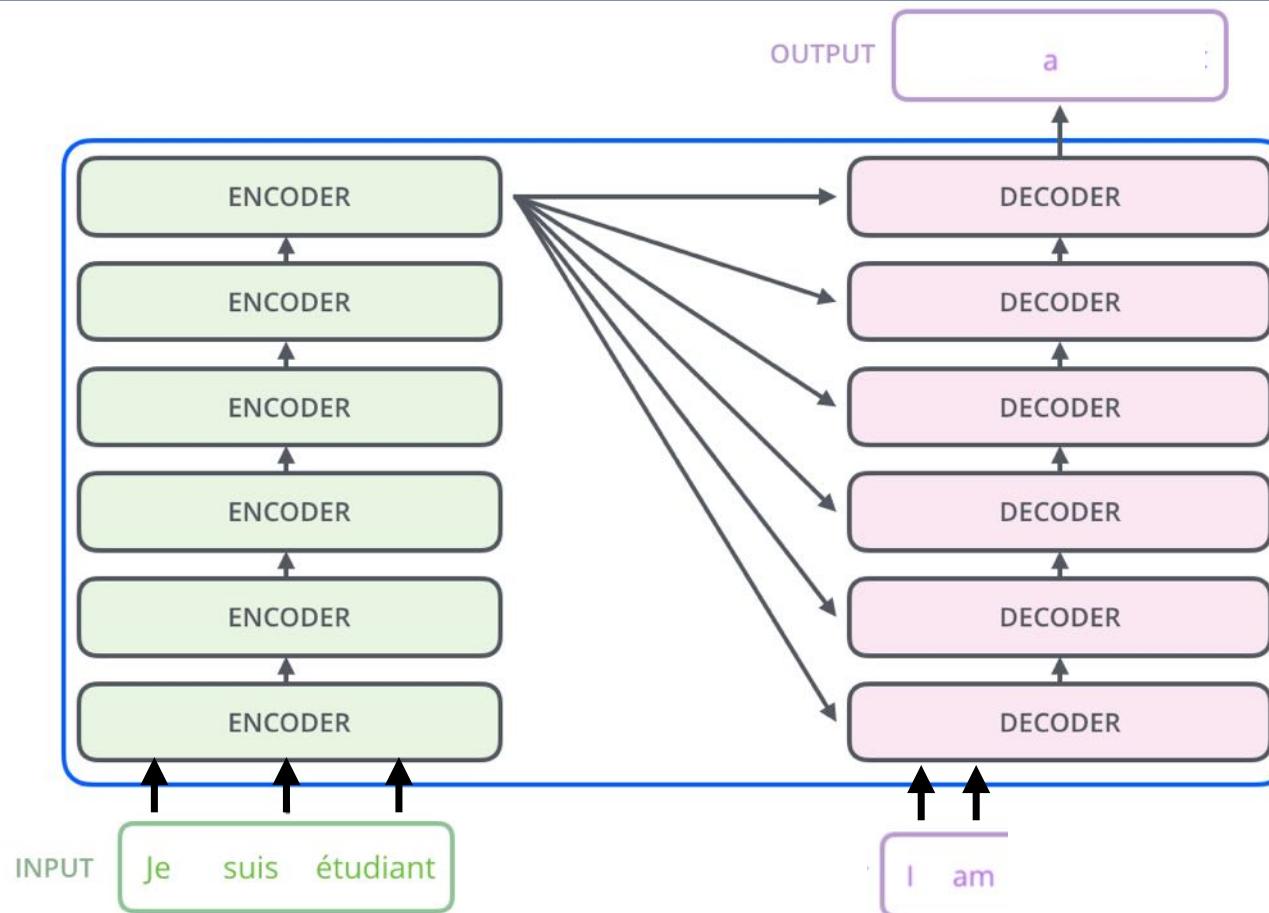
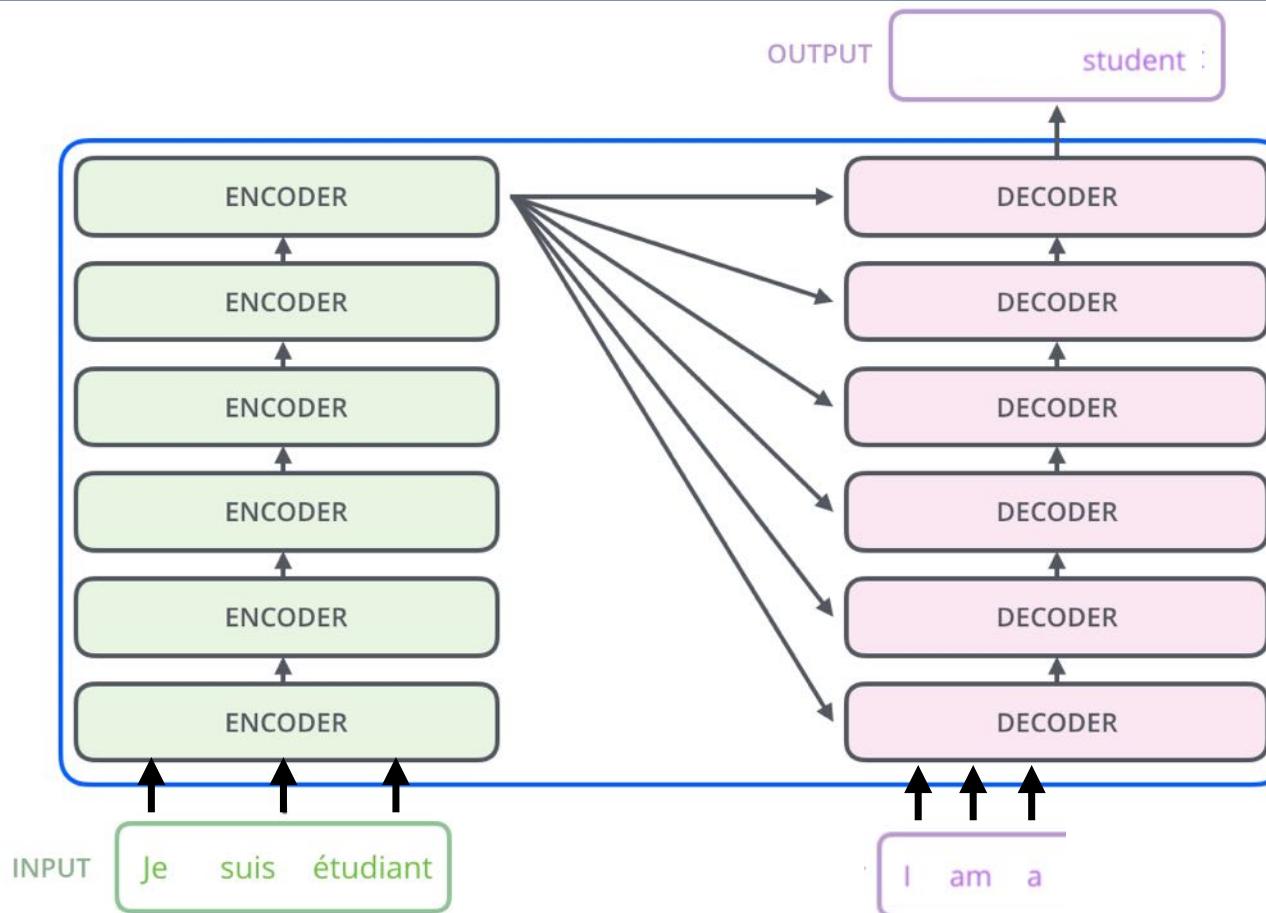
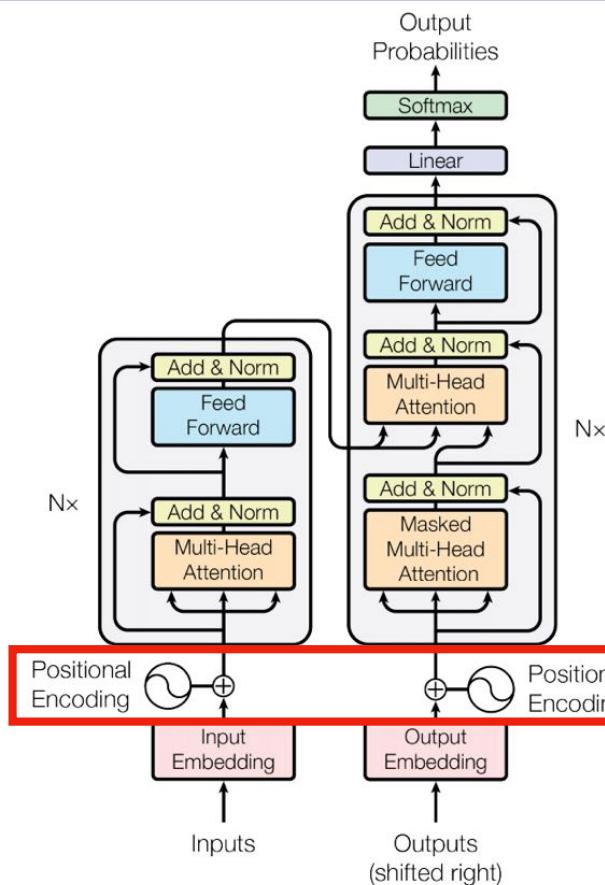


Illustration of Masking on the Decoder Side



Positional Encoding: Representing Order of Sequence

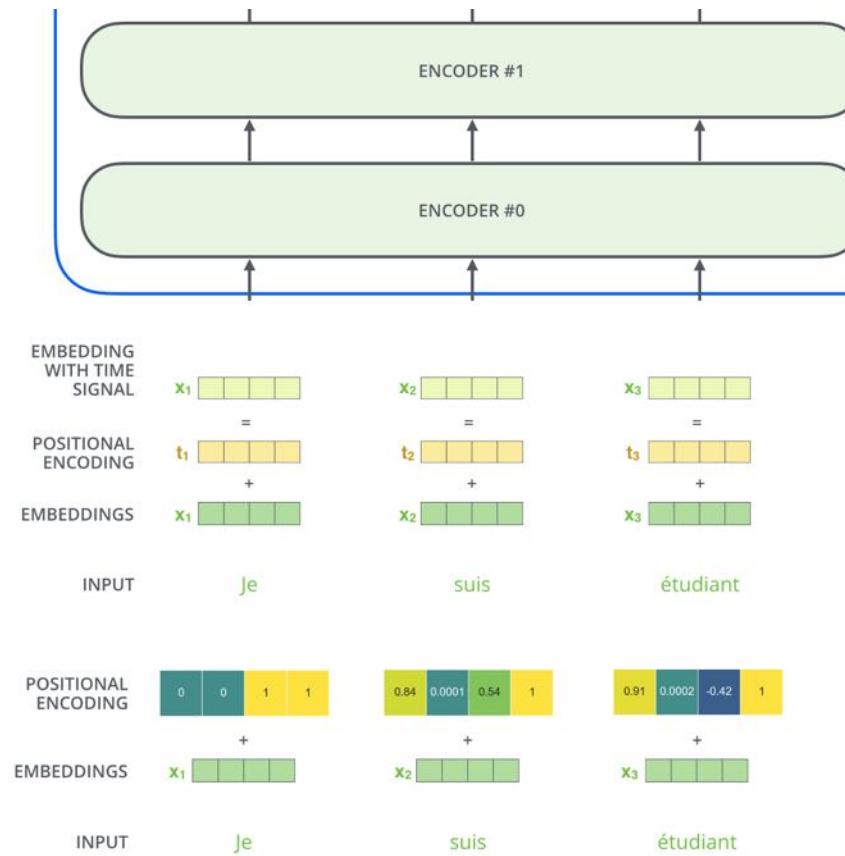


- unique encoding of each position
- consistent relative distance
- generalize to longer sequence

Attention is all you need. Vaswani et al. NeurIPS 2017

slide credit: Yongqin Xian

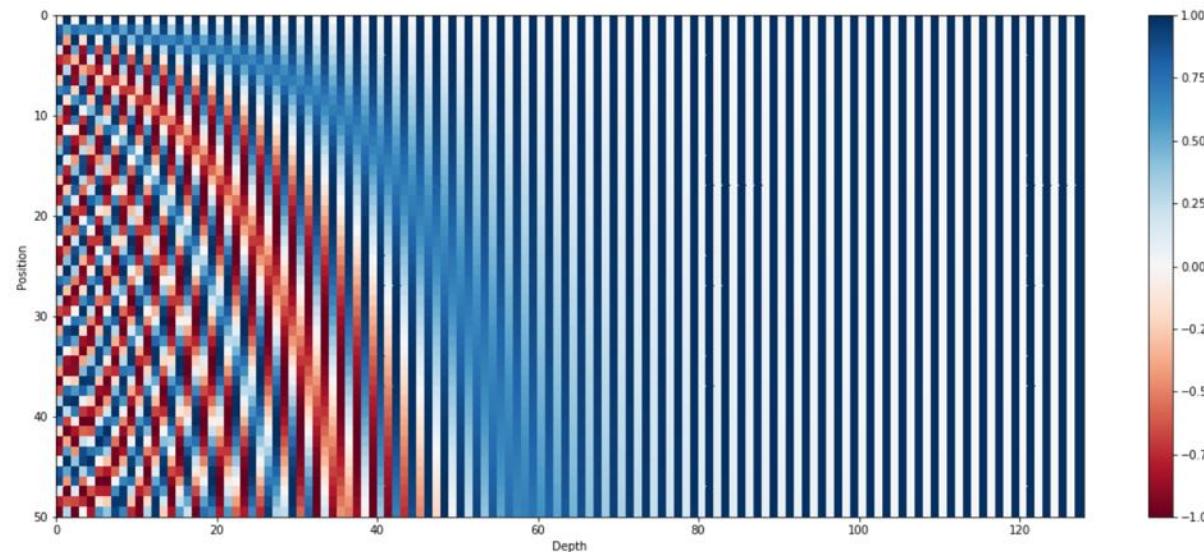
Positional Encoding: Representing Order of Sequence



Positional Encoding: Representing Order of Sequence

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

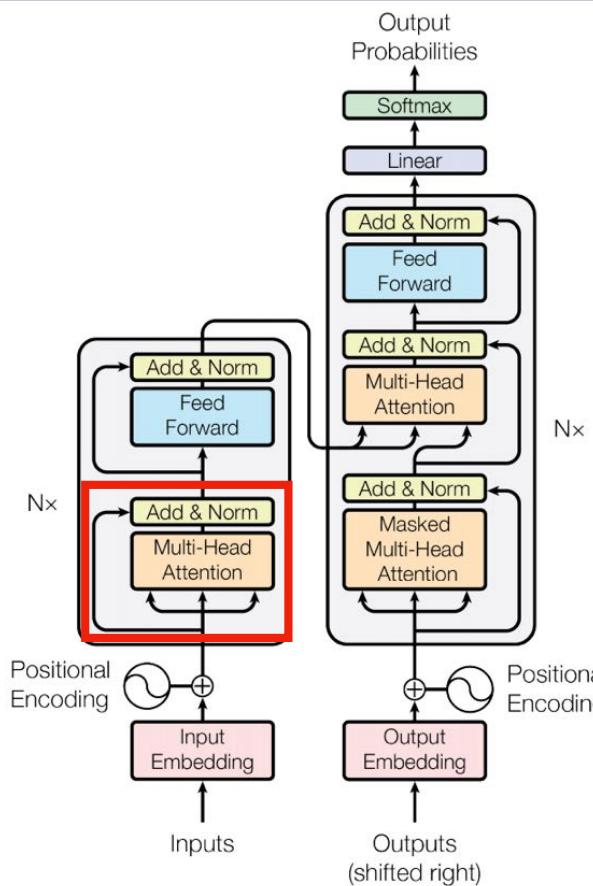
$$PE_{(pos,2i+1)} = \cos(pos/10000^{2i/d_{\text{model}}})$$



Attention is all you need. Vaswani et al. NeurIPS 2017

slide credit: Yongqin Xian

Add & Norm

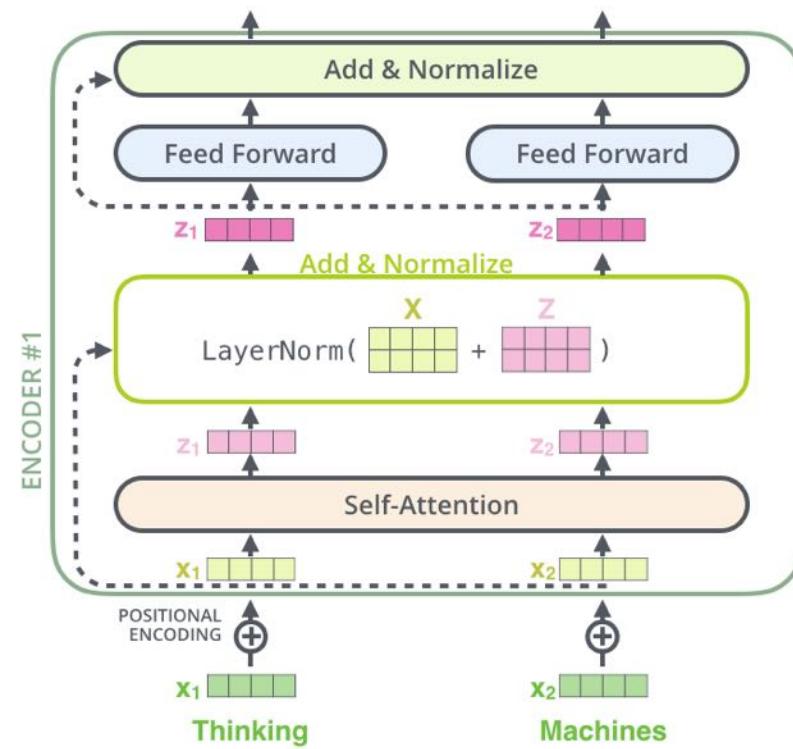
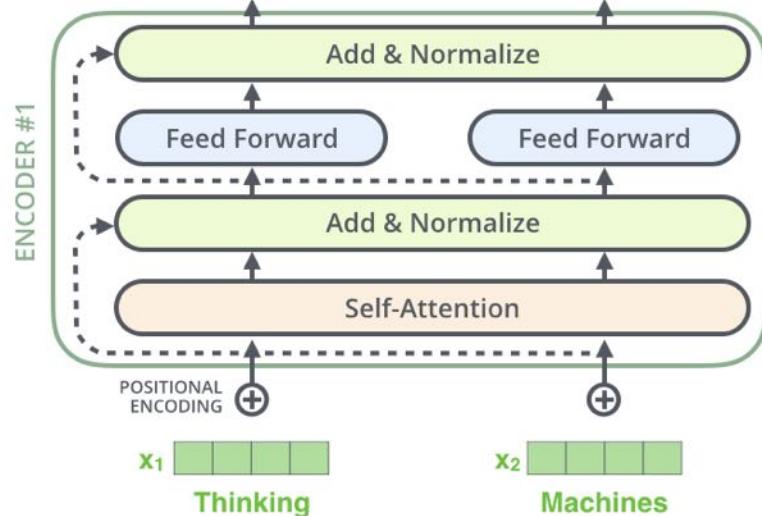


- residual connection
- normalization

Attention is all you need. Vaswani et al. NeurIPS 2017

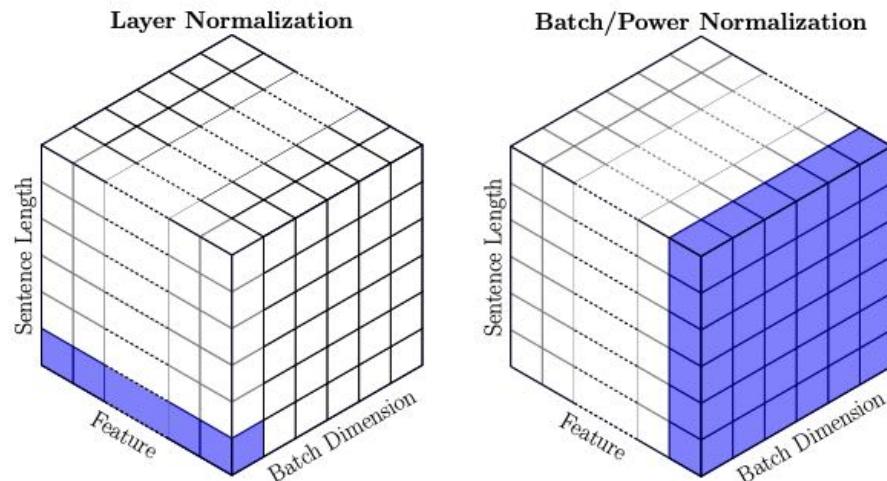
slide credit: Yongqin Xian

Add & Norm

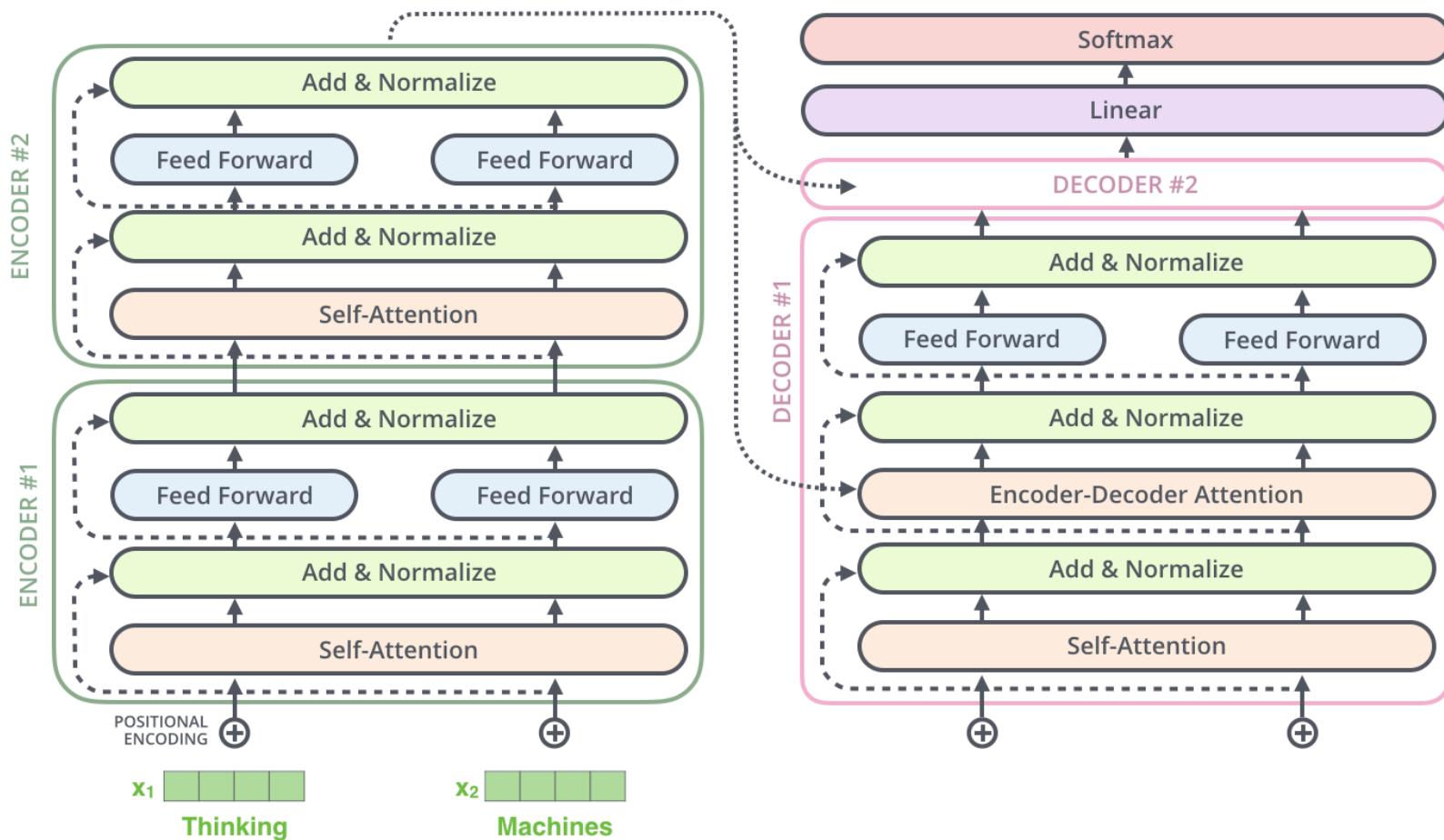


Layernorm vs. Batchnorm

- Layernorm:
 - ▶ normalize each instance (word) independently
- Batchnorm
 - ▶ normalize each feature dimension (channel) over the entire batch



“Complete Picture”



Overview of Today's Lecture

- Introduction of Transformer
 - Attention Is All You Need @ NeurIPS 2017 - <https://arxiv.org/abs/1706.03762>
- **Vision Transformer (ViT) for Image Classification**
 - An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale @ ICLR 2021
<https://arxiv.org/abs/2010.11929>
- Object Detection with Transformers (DETR)
 - DETR: End-to-End Object Detection with Transformers @ ECCV 2020
<https://arxiv.org/abs/2005.12872>
 - Deformable DETR: Deformable Transformers for End-to-End Object Detection @ ICLR 2021
<https://arxiv.org/abs/2010.04159>
- Swin Transformer
 - Swin Transformer: Hierarchical Vision Transformer using Shifted Windows @ ICCV 2021
<https://arxiv.org/abs/2103.14030>

Contributions

- Propose Vision Transformer (ViT) for image classification
 - ▶ without need for any CNNs
- ViT achieves state-of-art results for image classification
 - ▶ with lower computational costs

Vision Transformer (ViT)

- What are sensible tokens for images?



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

Vision Transformer (ViT)

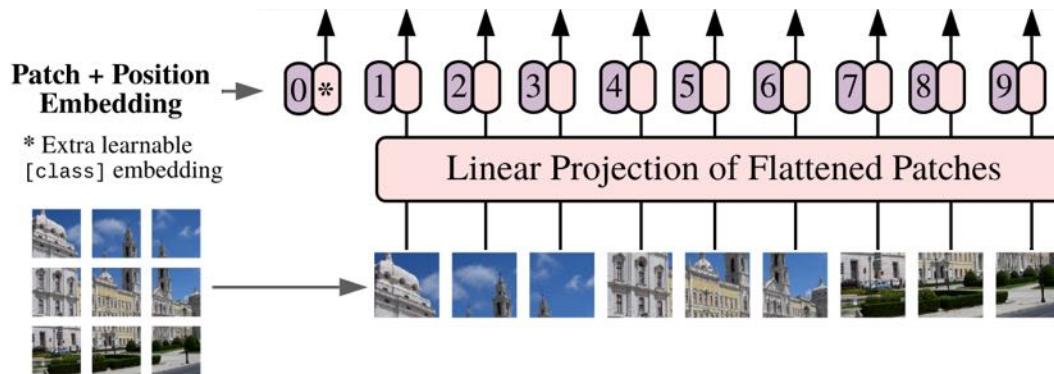
- What are sensible tokens for images?
 - ▶ Image Patches...
 - ▶ ... here: each image patch of size 16x16



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

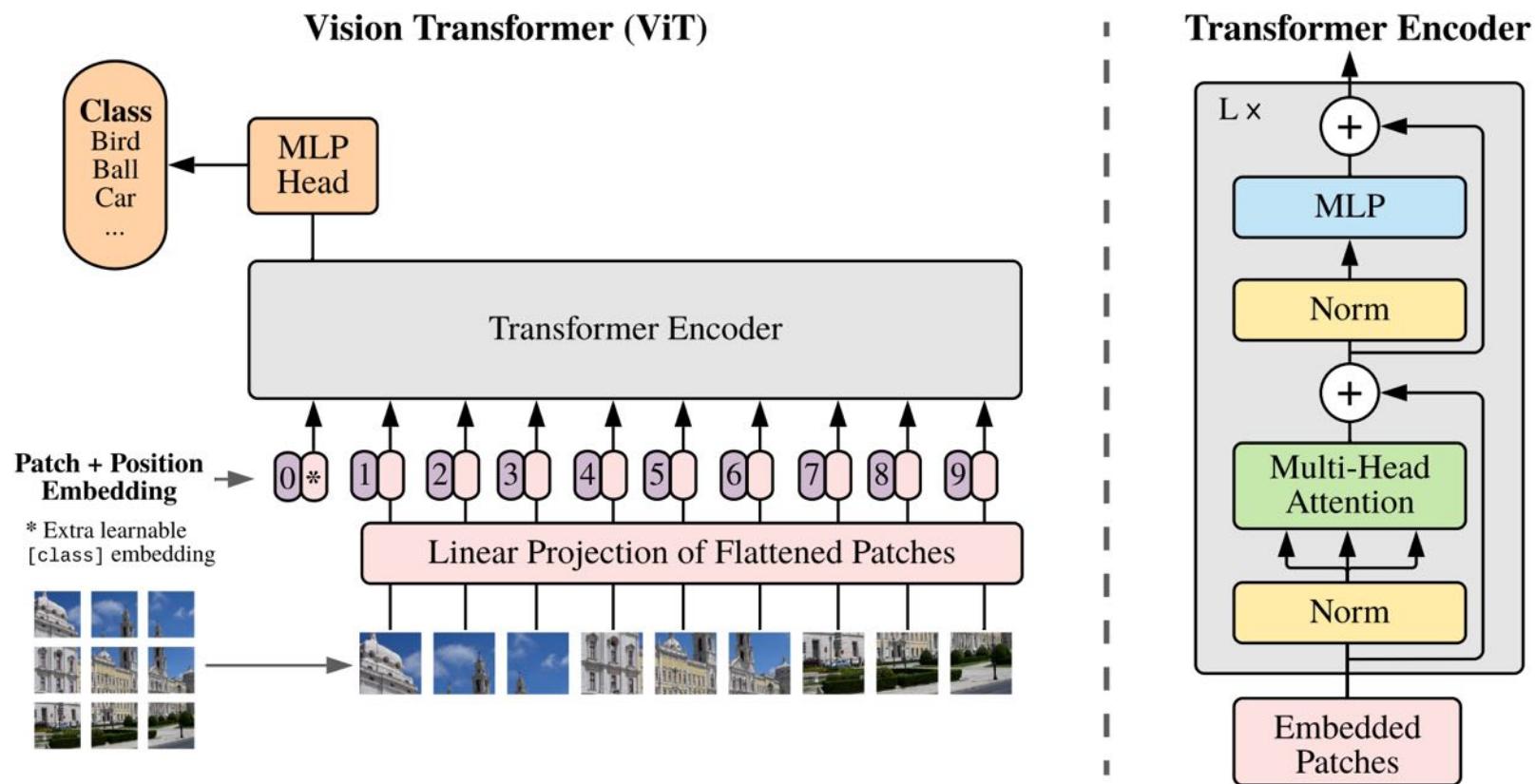
Vision Transformer (ViT)

- Positional encoding
 - ▶ learnable embedding for each patch position (sometimes also fixed positional encoding)
- Additional token for “class embedding”
 - ▶ learnable, its output embedding used for global image representation (and image classification — see next slide)



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

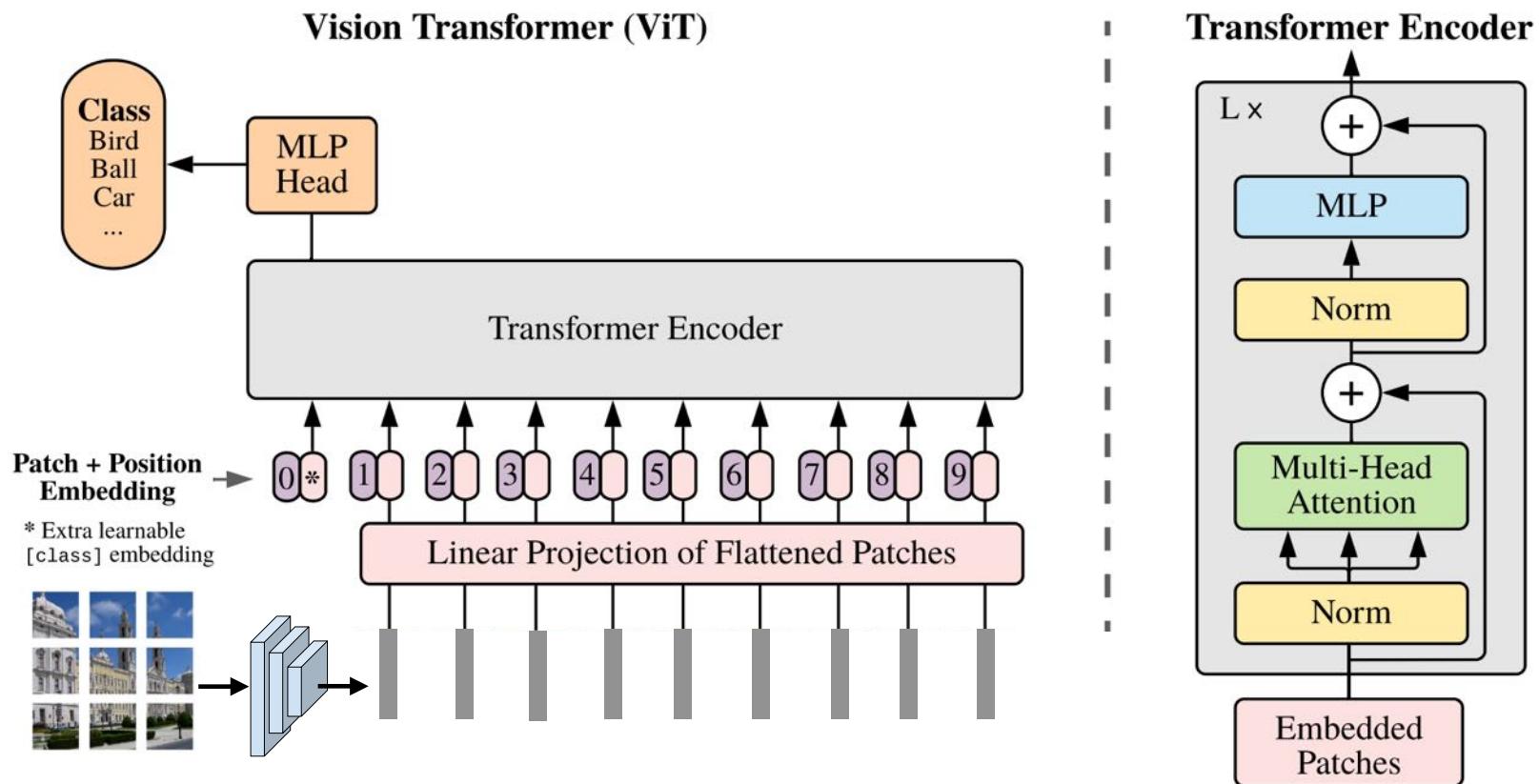
Vision Transformer (ViT)



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



Hybrid ViT Architecture (with CNN-embedding of image patches)



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

Experimental Setup

- Pretrained on large datasets with labels
 - ▶ ImageNet (1.4M images, 1K classes)
 - ▶ ImageNet 21K (14M images, 21K classes) [Deng et al., 2009]
 - ▶ JFT (303M images, 18K classes) [SUN et al., 2017]
- Finetuning for downstream tasks
 - ▶ replace the classification head with random weights
 - ▶ fine-tune the whole network
- Downstream tasks
 - ▶ ImageNet, CIFAR10/100, Oxford Pet/Flower, ...
 - ▶ 19-task VTAB classification suite: 1000 samples per task [Zhai et al., 2019]

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



Model Variants

- ViT-variants: Base, Large, Huge

Model	Layers	Hidden size D	MLP size	Heads	Params
ViT-Base	12	768	3072	12	86M
ViT-Large	24	1024	4096	16	307M
ViT-Huge	32	1280	5120	16	632M

- For comparison:
 - ResNet-50: 25M
 - ResNet-152: 60M
 - VGG16: 197M
- Notation in the following slide:
 - ▶ “ViT-L/16” means: ViT-Large model (see table, with 16x16 input patches)

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

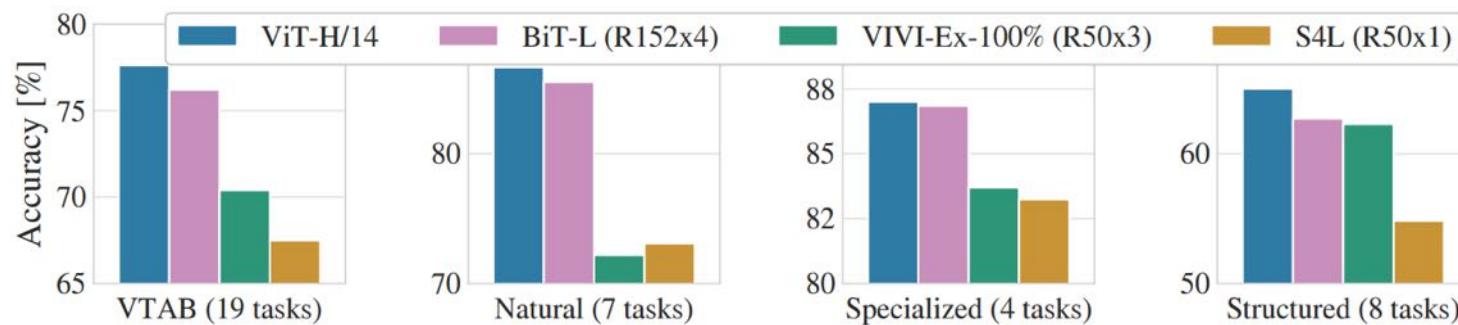
Results (top-1 accuracies)

	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

- Comparisons
 - ▶ BiT-L (ResNet152x4): SOTA on other datasets, pretrained on JFT [Kolesnikov et al., 2020]
 - ▶ Noisy student: SOTA on ImageNet, pretrained with ImageNet + unlabeled JFT [Xie et al., 2020]

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

Results on “Low-Shot” Transfer (1000 Examples per Task)

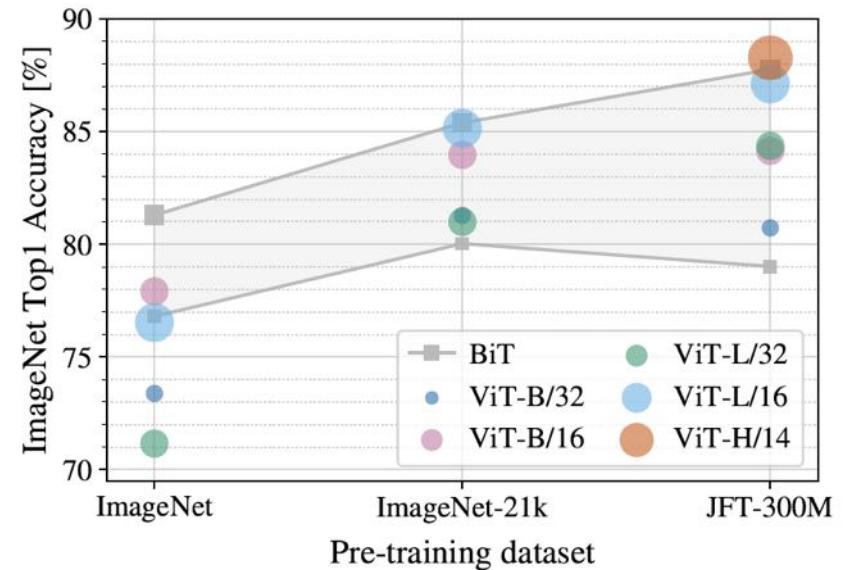
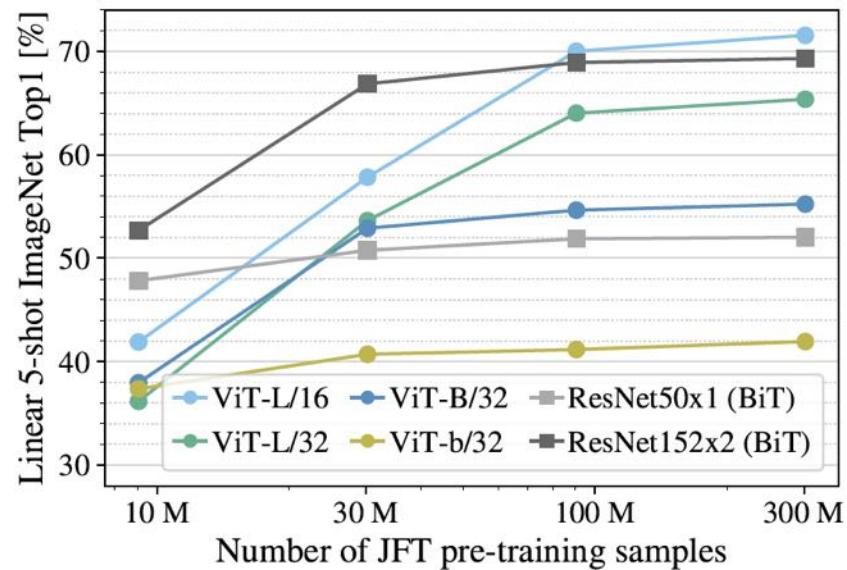


- Comparisons
 - ▶ BiT-L: SOTA, pretrained on JFT [Kolesnikov et al., 2020]
 - ▶ VIVI: ResNet, pretrained with ImageNet + youtube [Tschanne et al., 2020]
 - ▶ S4L: ResNet, semisupervised on ImageNet [Zhai et al., 2019]

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

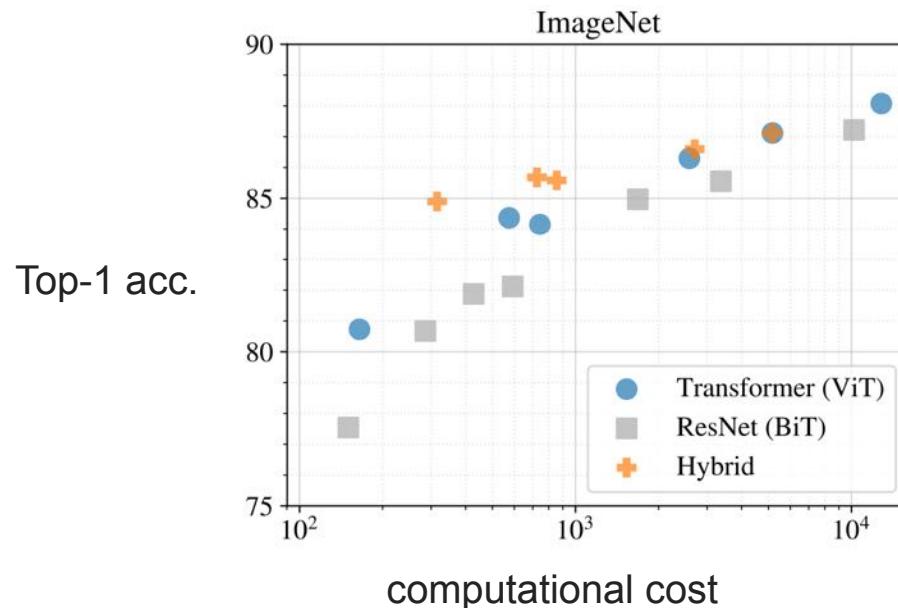
Ablation on Dataset Size

- ViT requires large dataset to outperform CNNs



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

Performance vs. Cost for Different Architectures

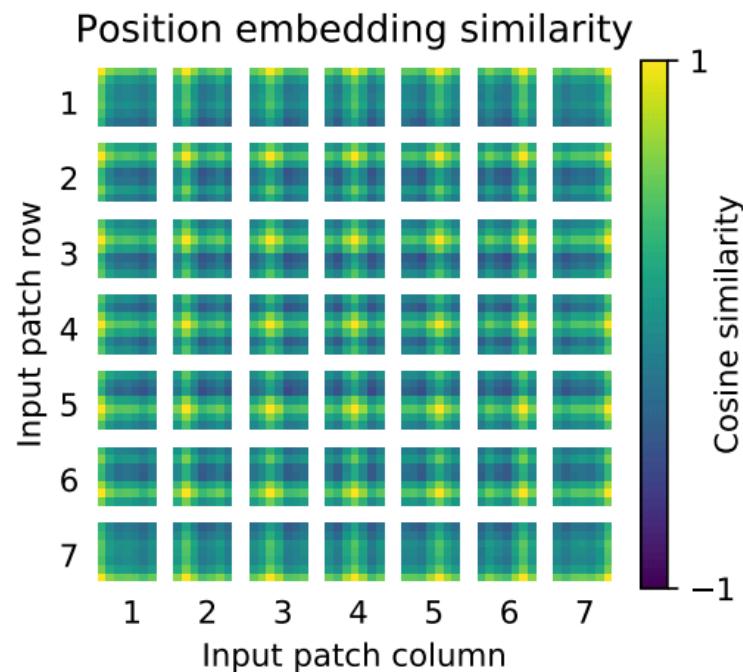


- ▶ ViT dominates ResNet on the performance-compute trade-off
- ▶ Hybrid outperforms ViT at small computational budget

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

Analysis of Positional Embedding

- ▶ Closer patches tend to have more similar positional embedding

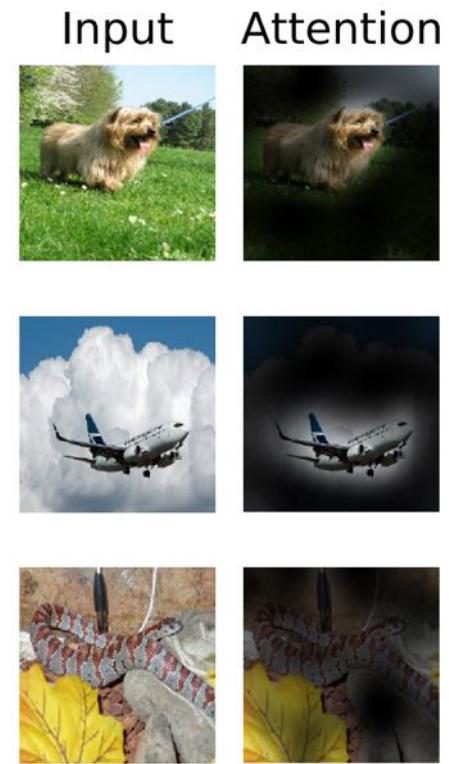


Pos. Emb.	Default/Stem
No Pos. Emb.	0.61382
1-D Pos. Emb.	0.64206
2-D Pos. Emb.	0.64001

5-shot learning top-1 acc. on ImageNet

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

Visualizing Attention



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

Main Conclusion

- A pure transformer can achieve SOTA (state of the art) performance for image classification
 - ▶ without convolutional layers
 - ▶ with a more scalable and simpler architecture

Overview of Today's Lecture

- Introduction of Transformer
 - Attention Is All You Need @ NeurIPS 2017 - <https://arxiv.org/abs/1706.03762>
- Vision Transformer (ViT) for Image Classification
 - An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale @ ICLR 2021
<https://arxiv.org/abs/2010.11929>
- **Object Detection with Transformers (DETR)**
 - DETR: End-to-End Object Detection with Transformers @ ECCV 2020
<https://arxiv.org/abs/2005.12872>
 - Deformable DETR: Deformable Transformers for End-to-End Object Detection @ ICLR 2021
<https://arxiv.org/abs/2010.04159>
- Swin Transformer
 - Swin Transformer: Hierarchical Vision Transformer using Shifted Windows @ ICCV 2021
<https://arxiv.org/abs/2103.14030>

DETR: End-to-End Object Detection with Transformers

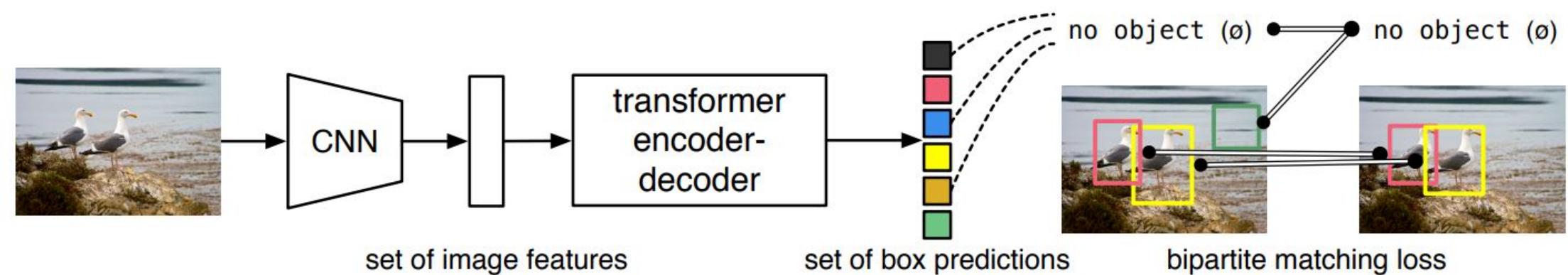
DEtection + TRansformer

Nicolas Carion*, Francisco Massa*, Gabriel Synnaeve, Nicolas Usunier,
Alexander Kirillov, and Sergey Zagoruyko

Facebook AI

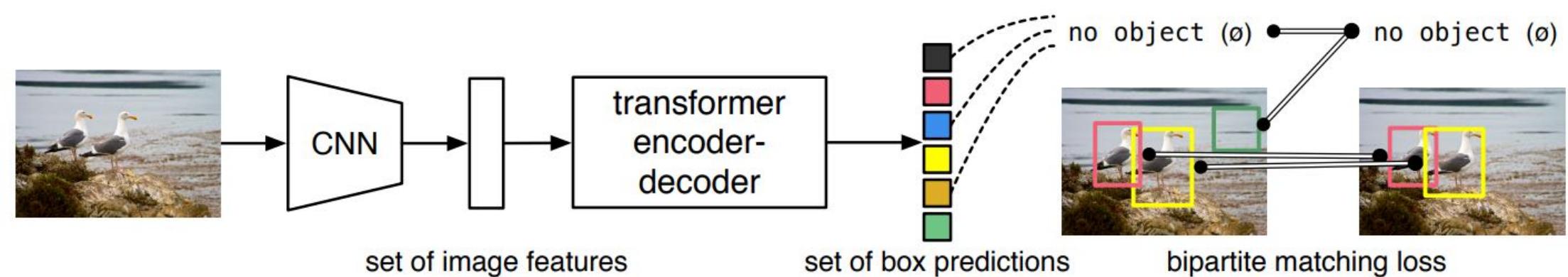
Introduction

- CNN-based Object Detectors (such as Yolo, Faster-RCNN) :
 - ▶ use proposal regions, anchors, window centers,... as bounding box candidates
 - ▶ these object detectors are not really trained end-to-end because of necessary postprocessing such non-maximum suppression (standard NMS, Soft NMS, ...)
- Object Detection with Transformers (DETR)
 - ▶ object detection pipeline that directly outputs a set of boxes from a Transformer
 - ▶ combination of a common CNN with a transformer architecture

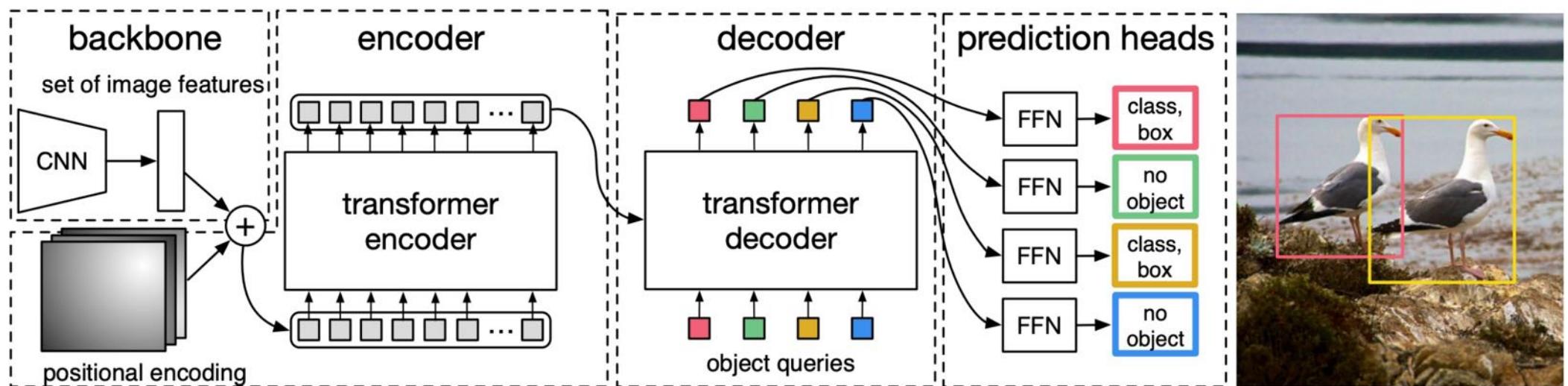
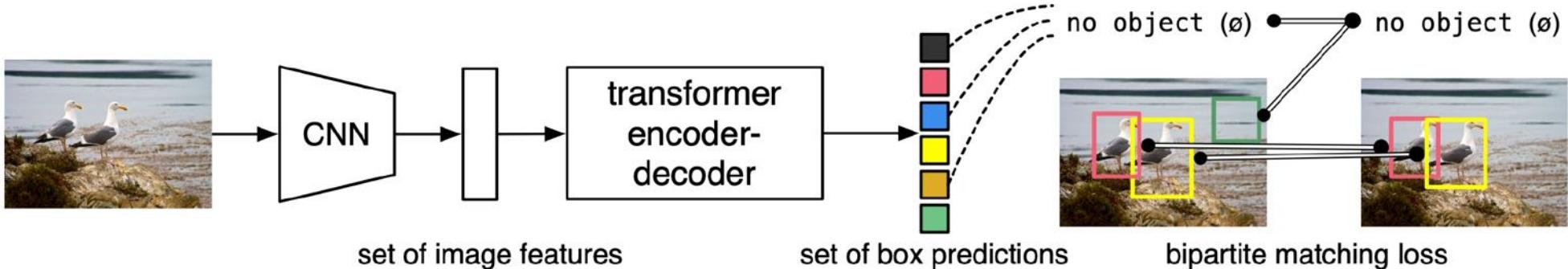


DEtection with TRansformers (DETR)

- Two core aspects:
 - ▶ 1. Architecture that directly **predicts** the final **set of all detections** (in parallel & single pass)
 - ▶ 2. **Set prediction loss** that results in a unique matching of object predictions and ground truth bounding boxes
 - bipartite matching loss to uniquely assign predictions with ground truth boxes
 - predictions with no match should yield a “no object” class prediction (see figure)



Object Detection with Transformers: DETR



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

slide credit: Justin Johnson



What are Object Queries?

- Object queries are the input of the decoder.
 - ▶ learnt embeddings
 - ▶ they use 100 (or 100s) of them
- Many think of them as “querying specific object classes, at specific locations, of a particular size, and of a particular aspect ratio” ...
 - ▶ but at the end “just learnt”

Table 1: Comparison with Faster R-CNN with a ResNet-50 and ResNet-101 backbones on the COCO validation set. The top section shows results for Faster R-CNN models in Detectron2 [50], the middle section shows results for Faster R-CNN models with GIoU [38], random crops train-time augmentation, and the long 9x training schedule. DETR models achieve comparable results to heavily tuned Faster R-CNN baselines, having lower AP_S but greatly improved AP_L. We use torchscript Faster R-CNN and DETR models to measure FLOPS and FPS. Results without R101 in the name correspond to ResNet-50.

Model	GFLOPS/FPS	#params	AP	AP ₅₀	AP ₇₅	AP _S	AP _M	AP _L
Faster RCNN-DC5	320/16	166M	39.0	60.5	42.3	21.4	43.5	52.5
Faster RCNN-FPN	180/26	42M	40.2	61.0	43.8	24.2	43.5	52.0
Faster RCNN-R101-FPN	246/20	60M	42.0	62.5	45.9	25.2	45.6	54.6
Faster RCNN-DC5+	320/16	166M	41.1	61.4	44.3	22.9	45.9	55.0
Faster RCNN-FPN+	180/26	42M	42.0	62.1	45.5	26.6	45.4	53.4
Faster RCNN-R101-FPN+	246/20	60M	44.0	63.9	47.8	27.2	48.1	56.0
DETR	86/28	41M	42.0	62.4	44.2	20.5	45.8	61.1
DETR-DC5	187/12	41M	43.3	63.1	45.9	22.5	47.3	61.1
DETR-R101	152/20	60M	43.5	63.8	46.4	21.9	48.0	61.8
DETR-DC5-R101	253/10	60M	44.9	64.7	47.7	23.7	49.5	62.3

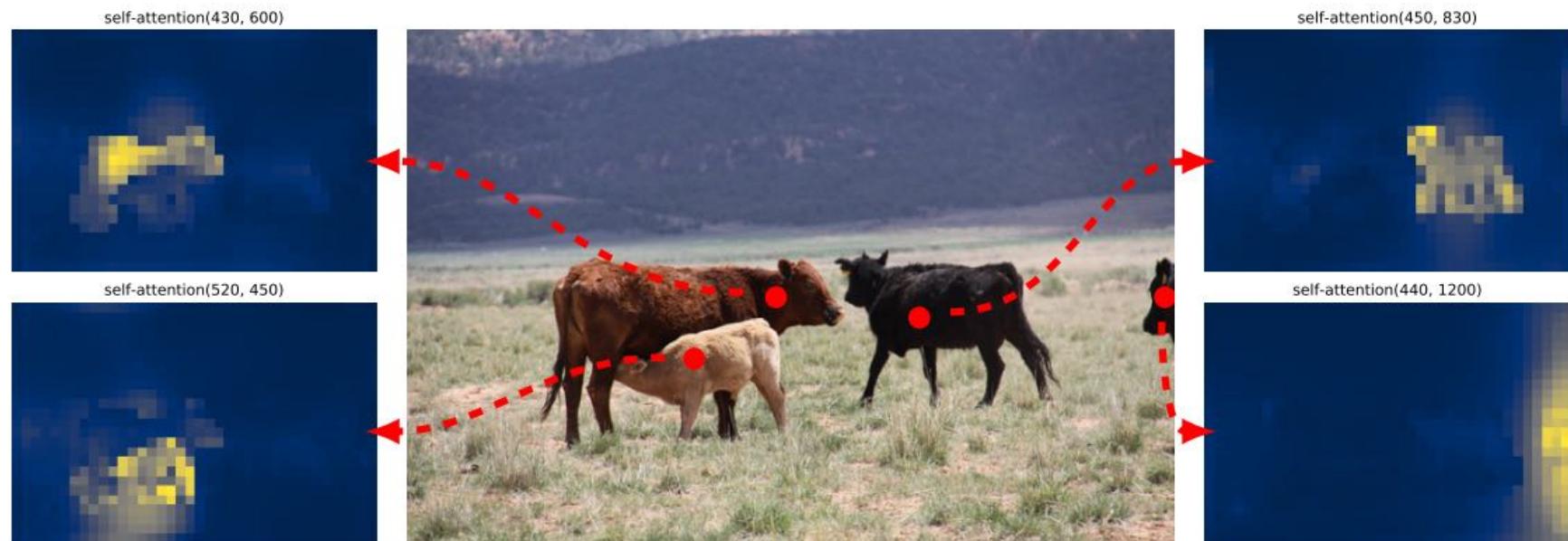
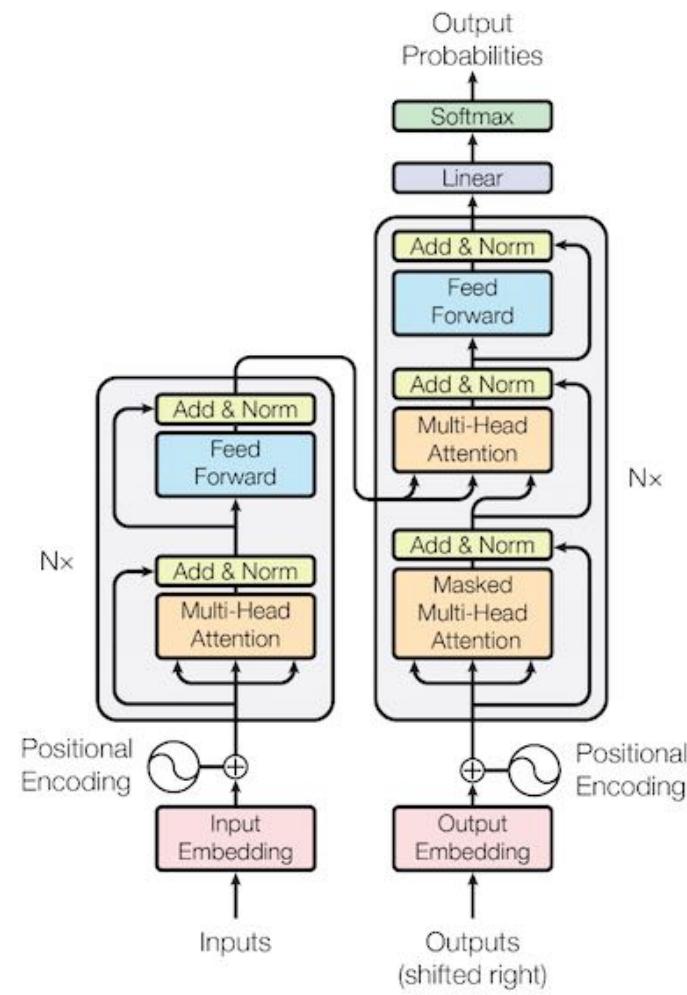
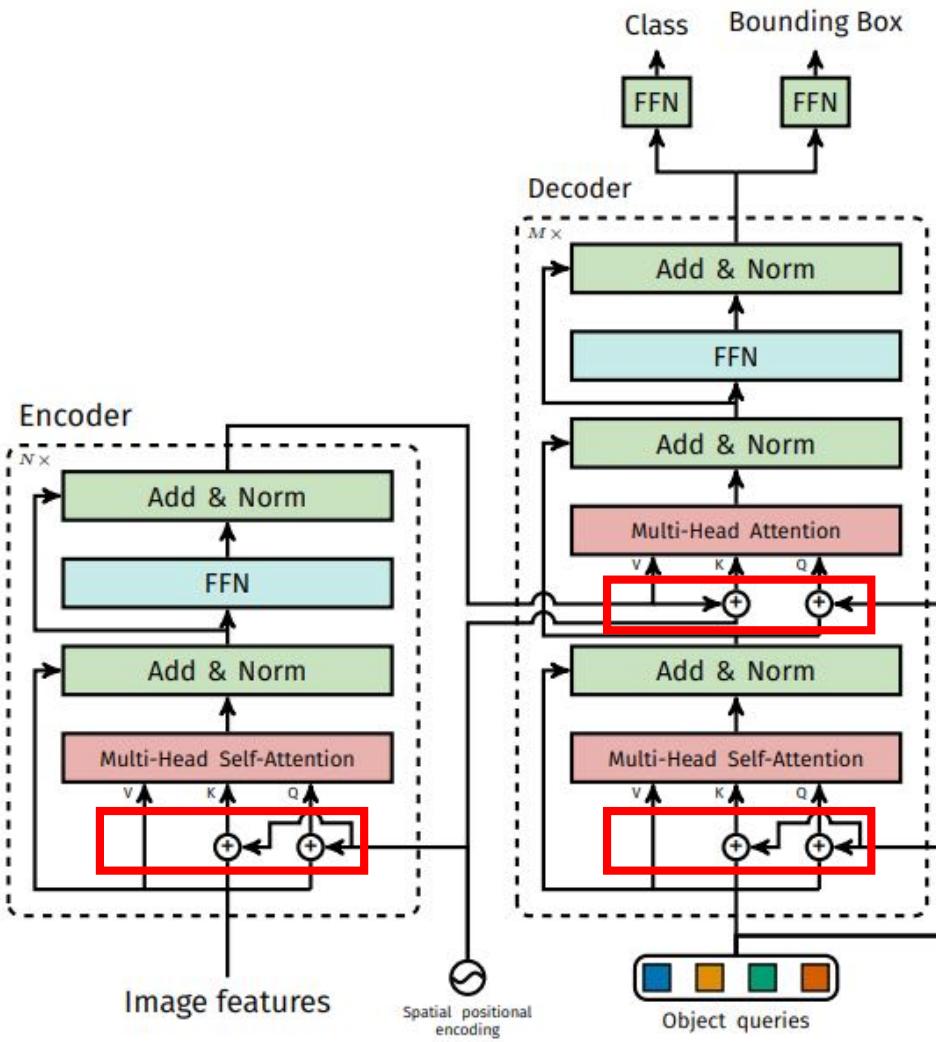


Fig. 3: Encoder self-attention for a set of reference points. The encoder is able to separate individual instances. Predictions are made with baseline DETR model on a validation set image.



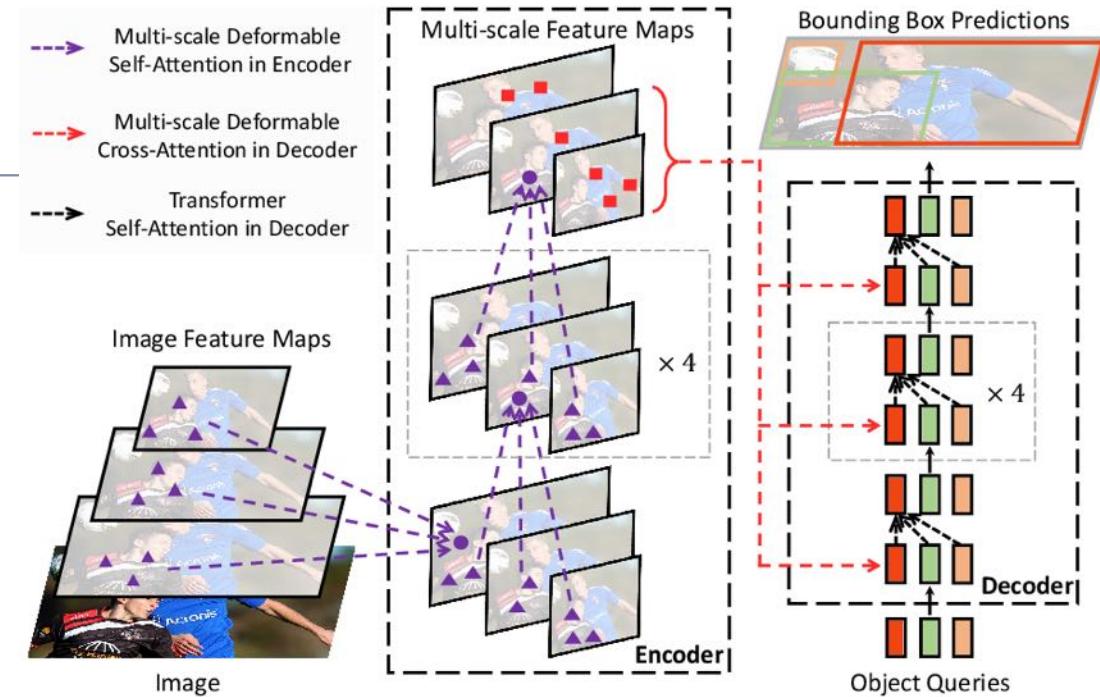
Importance of positional encodings

Table 3: Results for different positional encodings compared to the baseline (last row), which has fixed sine pos. encodings passed at every attention layer in both the encoder and the decoder. Learned embeddings are shared between all layers. Not using spatial positional encodings leads to a significant drop in AP. Interestingly, passing them in decoder only leads to a minor AP drop. All these models use learned output positional encodings.

encoder	decoder		AP	Δ	AP ₅₀	Δ
		decoder				
none	none	learned at input	32.8	-7.8	55.2	-6.5
sine at input	sine at input	learned at input	39.2	-1.4	60.0	-1.6
learned at attn.	learned at attn.	learned at attn.	39.6	-1.0	60.7	-0.9
none	sine at attn.	learned at attn.	39.3	-1.3	60.3	-1.4
sine at attn.	sine at attn.	learned at attn.	40.6	-	61.6	-

sine : fixed sine pos
learned : learnable

Deformable DETR



Method	Epochs	AP	AP ₅₀	AP ₇₅	AP _S	AP _M	AP _L	params	FLOPs	FPS
Faster R-CNN + FPN	109	42.0	62.1	45.5	26.6	45.4	53.4	42M	180G	26
DETR	500	42.0	62.4	44.2	20.5	45.8	61.1	41M	86G	28
DETR-DC5	500	43.3	63.1	45.9	22.5	47.3	61.1	41M	187G	12
DETR-DC5	50	35.3	55.7	36.8	15.2	37.5	53.6	41M	187G	12
DETR-DC5 ⁺	50	36.2	57.0	37.4	16.3	39.2	53.9	41M	187G	12
Deformable DETR	50	43.8	62.6	47.7	26.4	47.1	58.0	40M	173G	19
+ iterative bounding box refinement	50	45.4	64.7	49.0	26.8	48.3	61.7	40M	173G	19
++ two-stage Deformable DETR	50	46.2	65.2	50.0	28.8	49.2	61.7	40M	173G	19

Overview of Today's Lecture

- Introduction of Transformer
 - ▶ Attention Is All You Need @ NeurIPS 2017 - <https://arxiv.org/abs/1706.03762>
- Vision Transformer (ViT) for Image Classification
 - ▶ An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale @ ICLR 2021
<https://arxiv.org/abs/2010.11929>
- Object Detection with Transformers (DETR)
 - ▶ DETR: End-to-End Object Detection with Transformers @ ECCV 2020
<https://arxiv.org/abs/2005.12872>
 - ▶ Deformable DETR: Deformable Transformers for End-to-End Object Detection @ ICLR 2021
<https://arxiv.org/abs/2010.04159>
- **Swin Transformer**
 - ▶ Swin Transformer: Hierarchical Vision Transformer using Shifted Windows @ ICCV 2021
<https://arxiv.org/abs/2103.14030>



Swin Transformer: Hierarchical Vision Transformer using Shifted Windows

Ze Liu^{1,2†*} Yutong Lin^{1,3†*} Yue Cao^{1*} Han Hu^{1*‡} Yixuan Wei^{1,4†}
Zheng Zhang¹ Stephen Lin¹ Baining Guo¹

¹Microsoft Research Asia ²University of Science and Technology of China

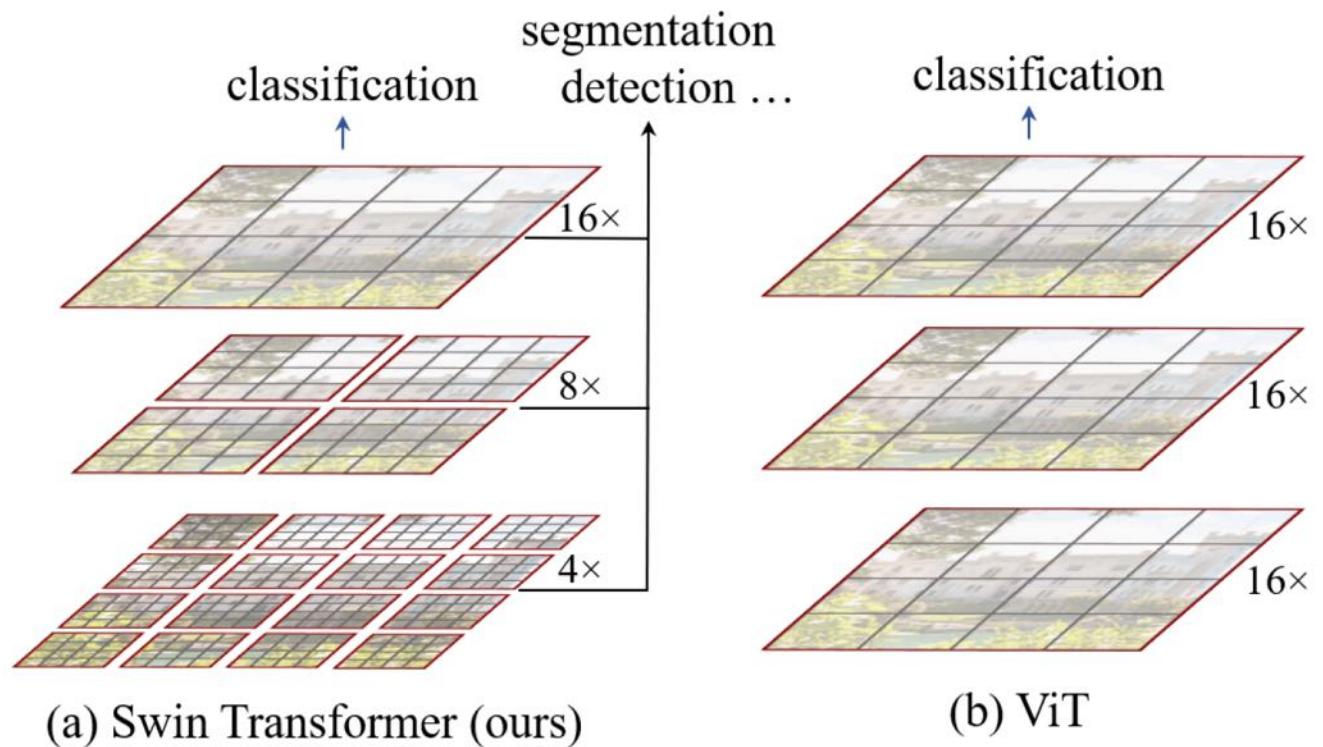
³Xian Jiaotong University ⁴Tsinghua University

{v-zeliu1, v-yutlin, yuecao, hanhu, v-yixwe, zhez, stevelin, bainguo}@microsoft.com

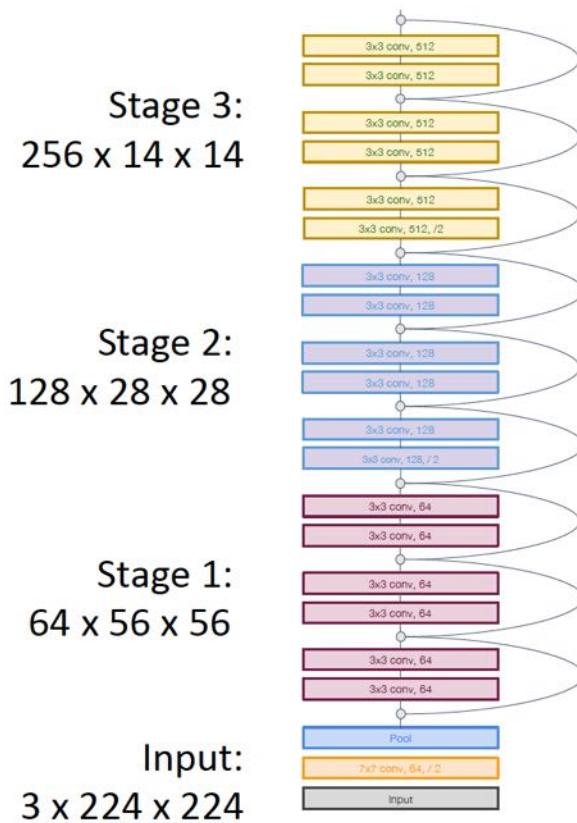
ICCV 2021 — Best Paper Award (Marr Prize)

Swin Transformer vs. standard ViT (Vision Transformer)

- Swin Transformer
 - ▶ hierarchical: token size increases from 4x4 to 16x16 pixels
 - ▶ architecture can be used for
 - image classification, but also
 - semantic segmentation and
 - object detection
 - instance segmentation
 - ▶ computationally more efficient than ViT



ViT vs CNN

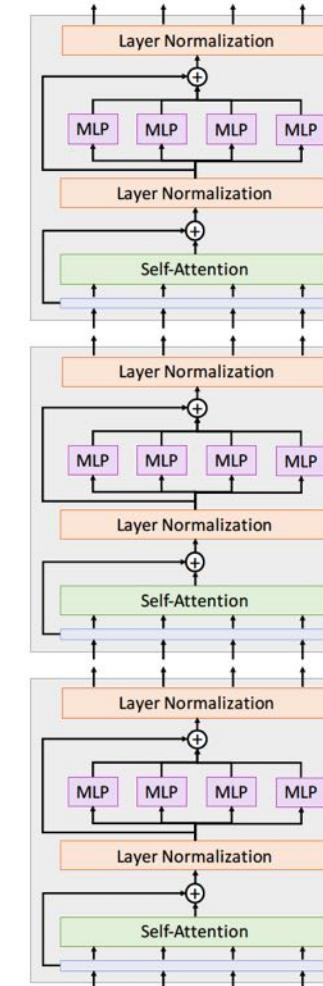


In most CNNs (including ResNets), **decrease** resolution and **increase** channels as you go deeper in the network (Hierarchical architecture)

Useful since objects in images can occur at various scales

In a ViT, all blocks have
same resolution and
number of channels
(Isotropic architecture)

Can we build a **hierarchical** ViT model?



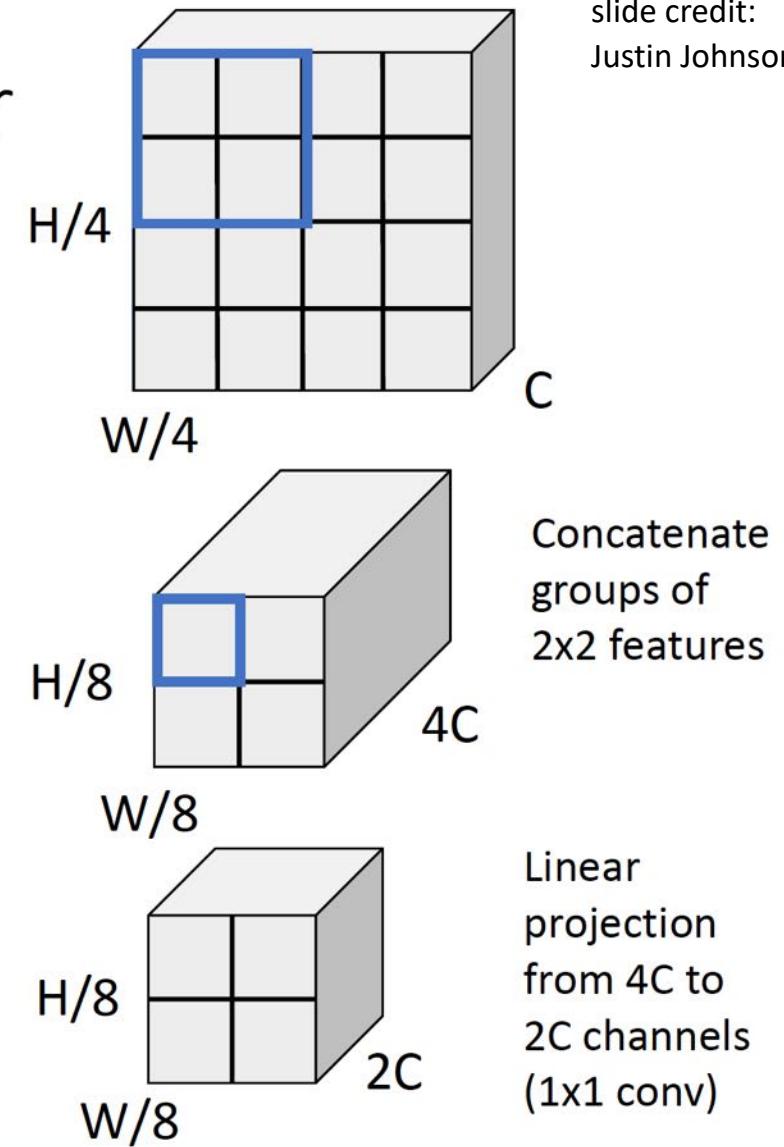
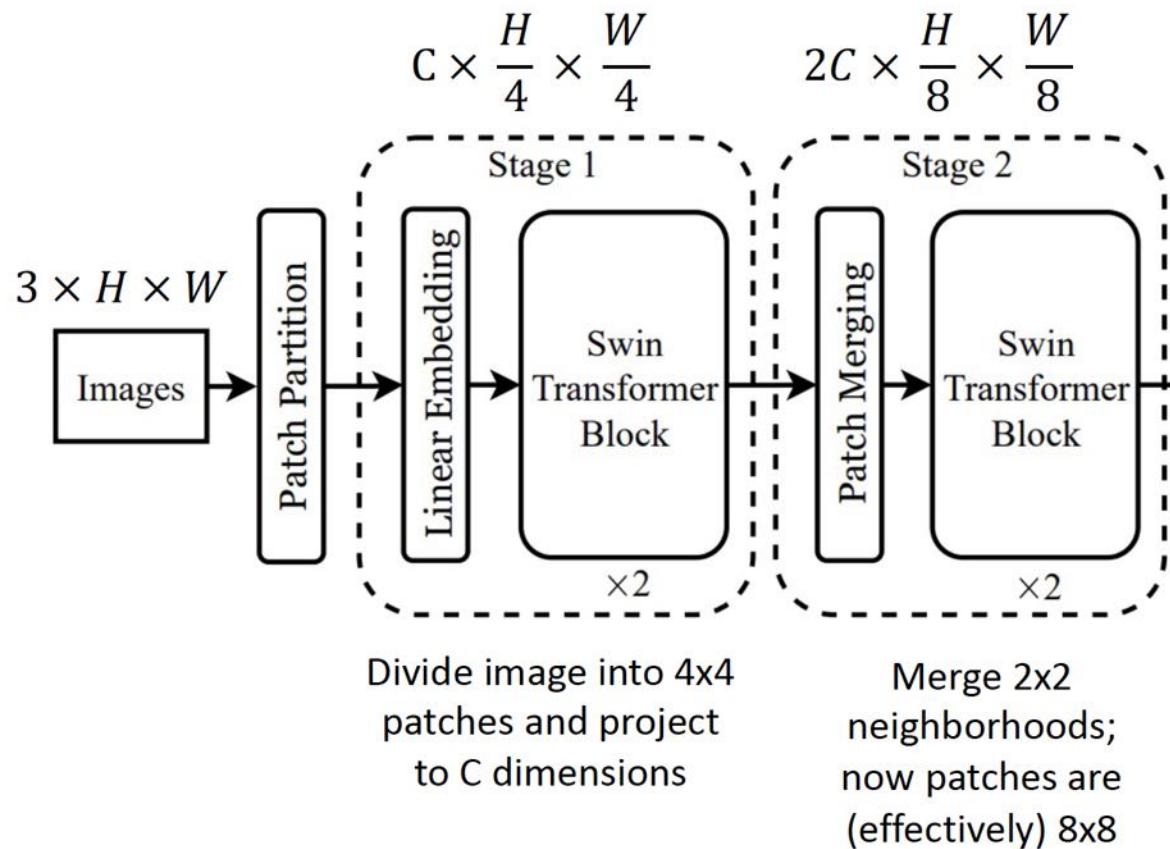
3rd block:
 $768 \times 14 \times 14$

2nd block:
768 x 14 x 14

1st block:
768 x 14 x 14

Input:
 $3 \times 224 \times 224$

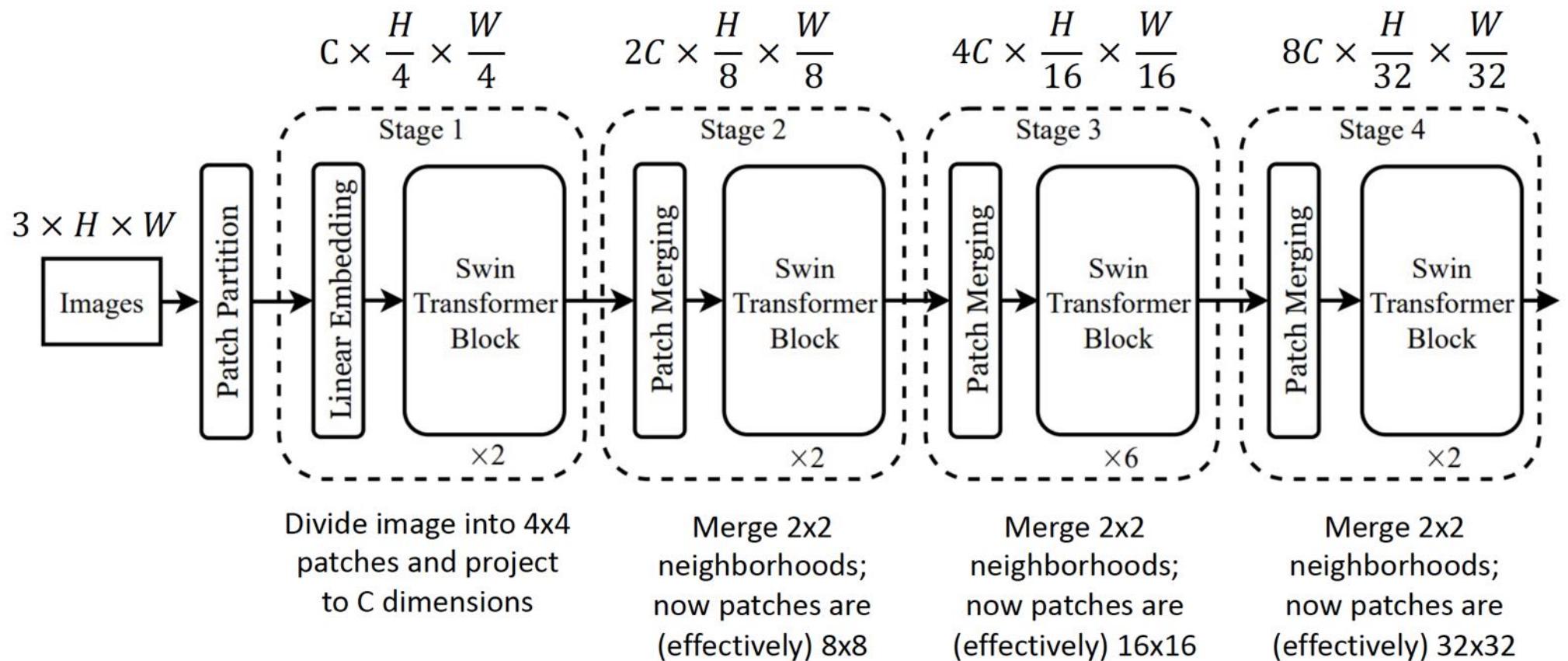
Hierarchical ViT: Swin Transformer



Liu et al, "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows", ICCV 2021



Hierarchical ViT: Swin Transformer



Liu et al, "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows", ICCV 2021

slide credit: Justin Johnson

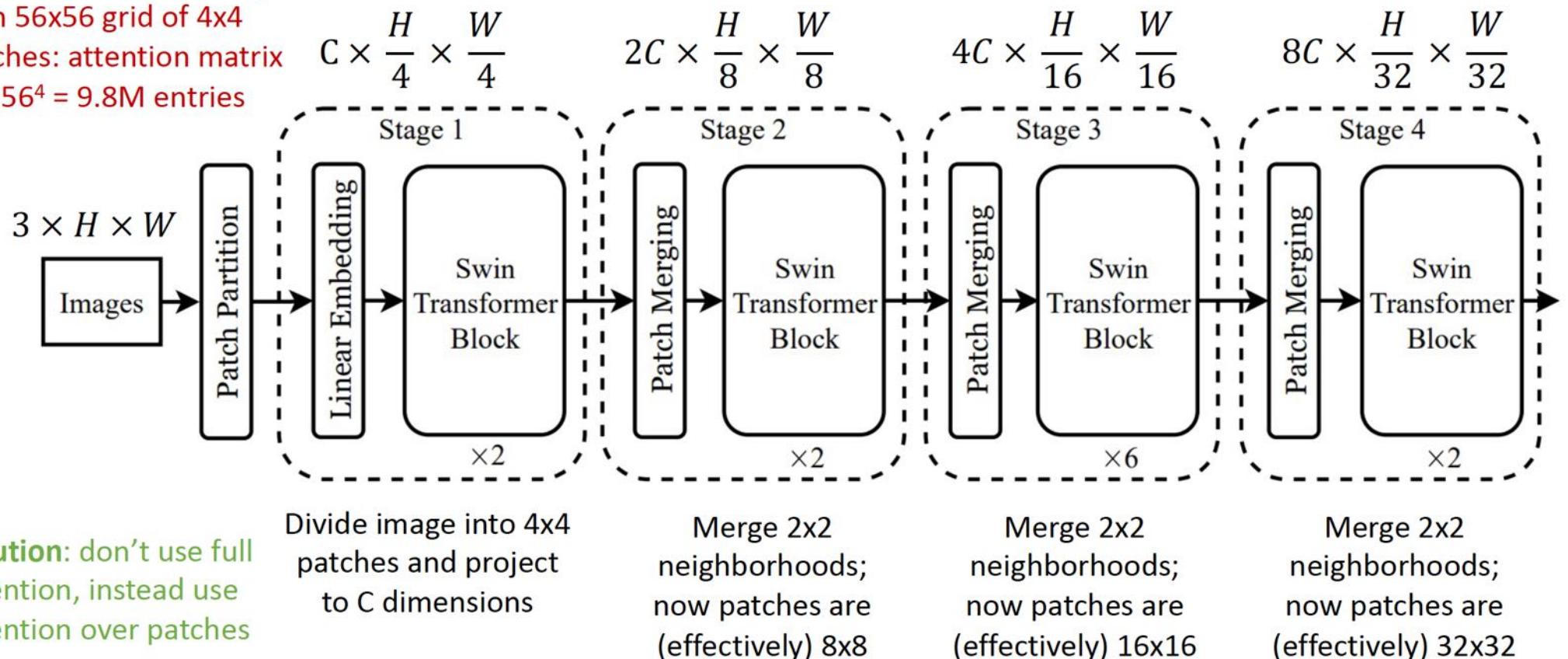


Hierarchical ViT: Swin Transformer

Problem: 224x224 image

with 56x56 grid of 4x4

patches: attention matrix
has $56^4 = 9.8\text{M}$ entries



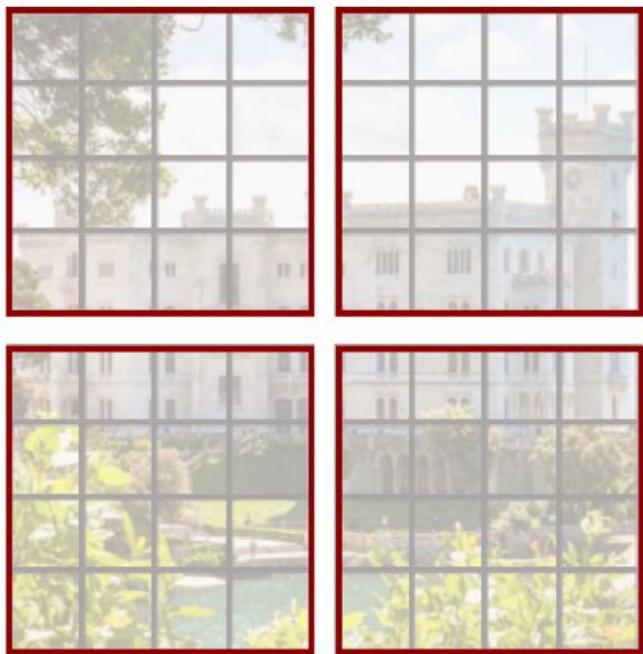
Solution: don't use full attention, instead use attention over patches

Liu et al, "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows", ICCV 2021

slide credit: Justin Johnson



Swin Transformer: Window Attention



With $H \times W$ grid of **tokens**, each attention matrix is H^2W^2 – **quadratic** in image size

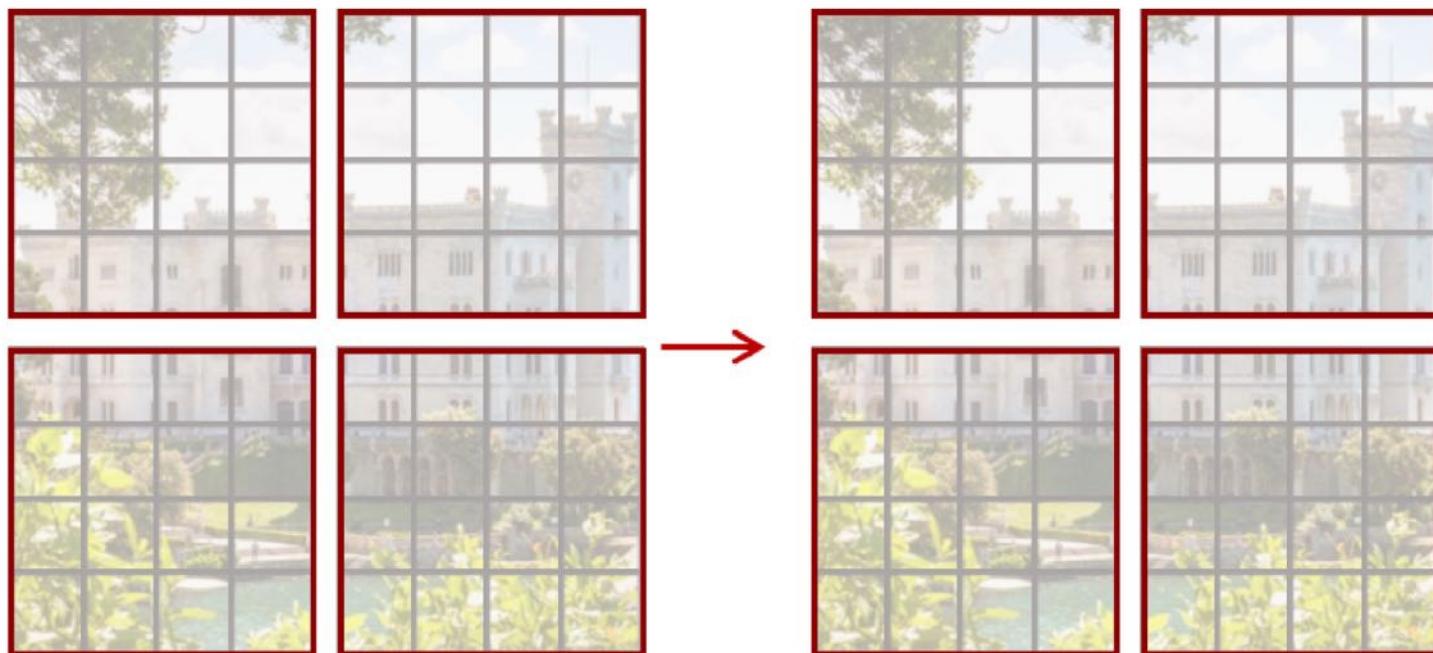
Rather than allowing each **token** to attend to all other tokens, instead divide into **windows** of $M \times M$ tokens (here $M=4$); only compute attention within each window

Total size of all attention matrices is now:
 $M^4(H/M)(W/M) = M^2HW$

Linear in image size for fixed M !
Swin uses $M=7$ throughout the network

Swin Transformer: Window Attention

Problem: tokens only interact with other tokens within the same window; no communication across windows



Swin Transformer: Shifted Window Attention

Solution: Alternate between normal windows and shifted windows in successive Transformer blocks



Ugly detail:
Non-square
windows at
edges and
corners

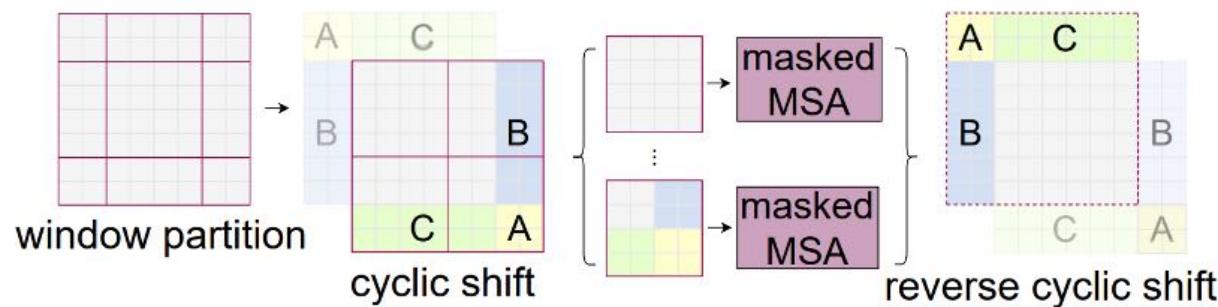
Liu et al, "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows", ICCV 2021

slide credit: Justin Johnson



Efficient batch computation for shifted configuration

- First idea
 - ▶ add padding... works, but increased computation
- Their idea: cyclic-shift:



Swin Transformer: Shifted Window Attention

Solution: Alternate between normal windows and shifted windows in successive Transformer blocks



Detail: Relative Positional Bias

ViT adds positional embedding to input tokens, encodes *absolute position* of each token in the image

Swin does not use positional embeddings, instead encodes *relative position* between patches when computing attention:

Standard Attention:

$$A = \text{Softmax} \left(\frac{QK^T}{\sqrt{D}} \right) V$$

$Q, K, V: M^2 \times D$ (Query, Key, Value)

Swin Transformer: Shifted Window Attention

Solution: Alternate between normal windows and shifted windows in successive Transformer blocks



Block L: Normal windows

Block L+1: Shifted Windows

Detail: Relative Positional Bias

ViT adds positional embedding to input tokens, encodes *absolute position* of each token in the image

Swin does not use positional embeddings, instead encodes *relative position* between patches when computing attention:

Attention with relative bias:

$$A = \text{Softmax} \left(\frac{QK^T}{\sqrt{D}} + B \right) V$$

$Q, K, V: M^2 \times D$ (Query, Key, Value)
 $B: M^2 \times M^2$ (learned biases)

slide credit: Justin Johnson

Application to Various Tasks...

- General idea: think of (hierarchical tokens) as feature maps
 - allows to apply standard techniques for detection (e.g. Faster-RNN), instance segmentation (e.g. Mask-RCNN), semantic segmentation

	ImageNet		COCO		ADE20k
	top-1	top-5	AP ^{box}	AP ^{mask}	mIoU
w/o shifting	80.2	95.1	47.7	41.5	43.3
shifted windows	81.3	95.6	50.5	43.7	46.1
no pos.	80.1	94.9	49.2	42.6	43.8
abs. pos.	80.5	95.2	49.0	42.4	43.2
abs.+rel. pos.	81.3	95.6	50.2	43.4	44.0
rel. pos. w/o app.	79.3	94.7	48.2	41.9	44.1
rel. pos.	81.3	95.6	50.5	43.7	46.1

