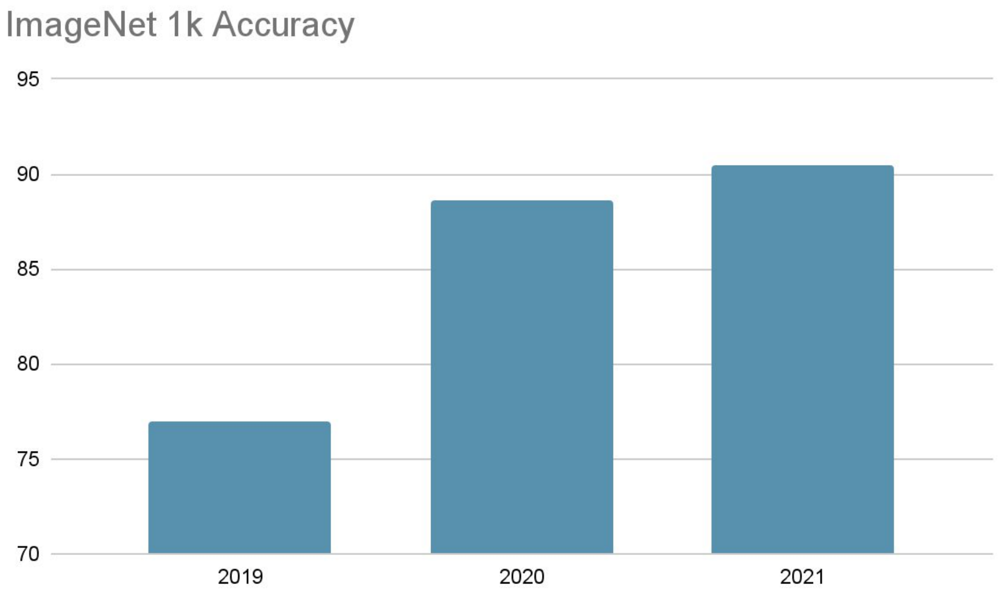
Self-Attention for Vision

Ashish Vaswani[1], Prajit Ramachandran[1], and Aravind Srinivas[2]

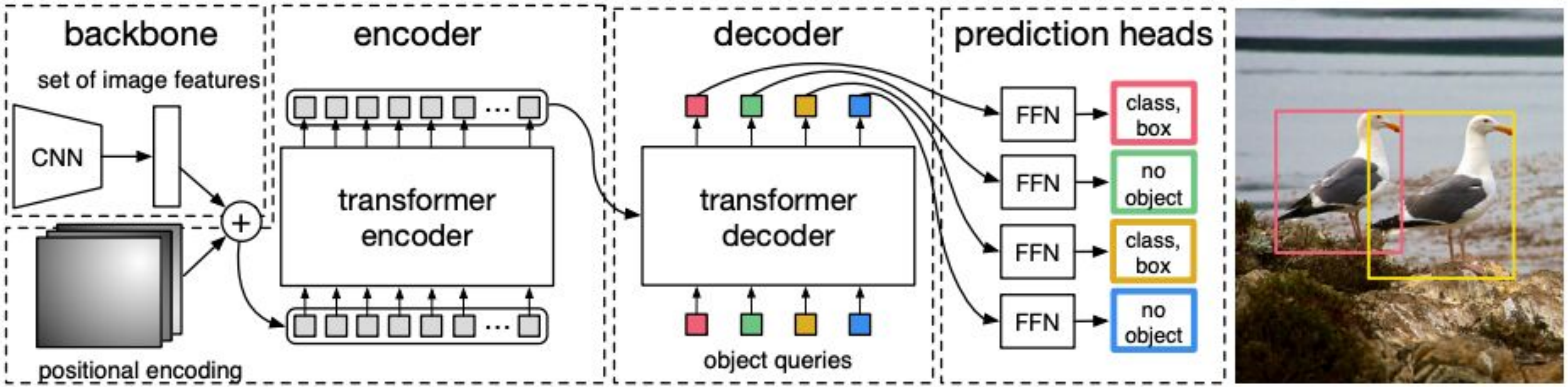
[1] Google Research, [2] UC Berkeley

Self-Attention's moment in Vision has arrived

Image Classification

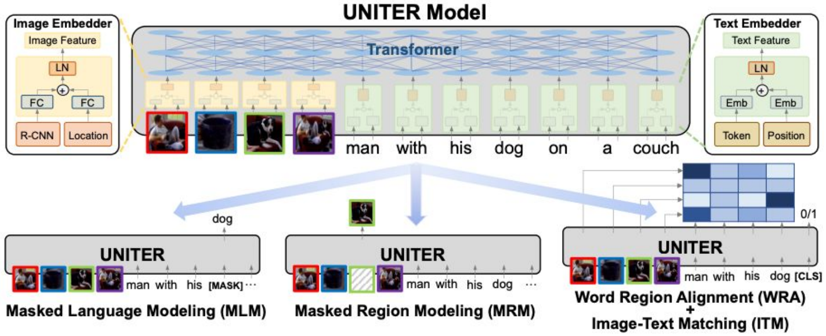


Object detection

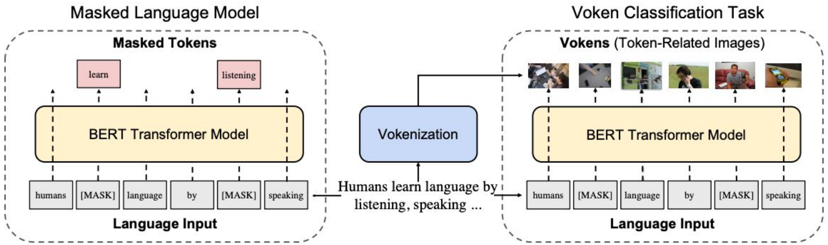


DETR, Carion et al.

Multimodal models



UNITER, Chen et al.



Vokenization, Tan et al.

Emergent localization

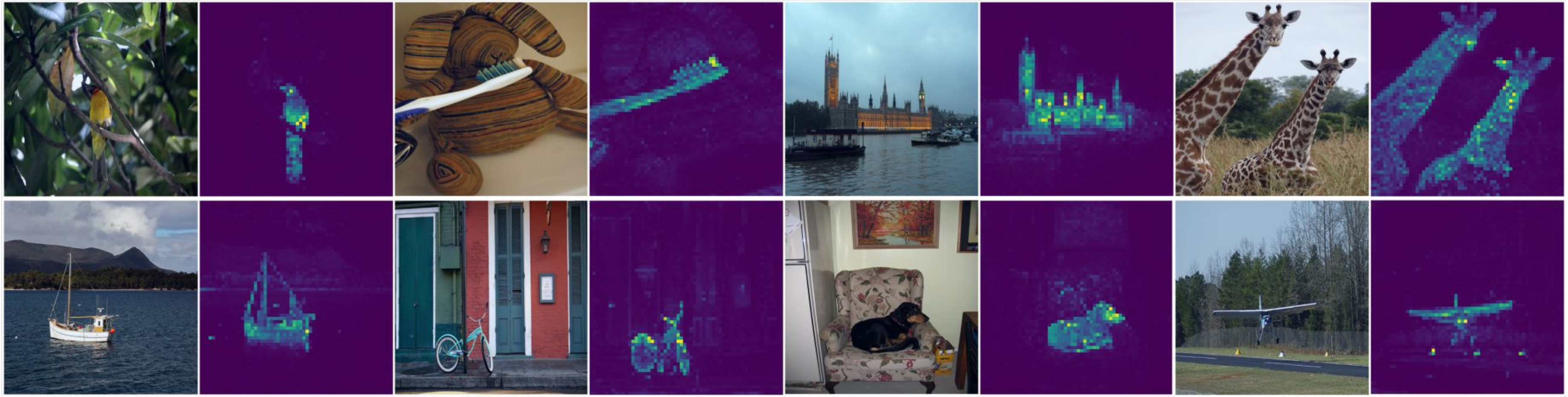
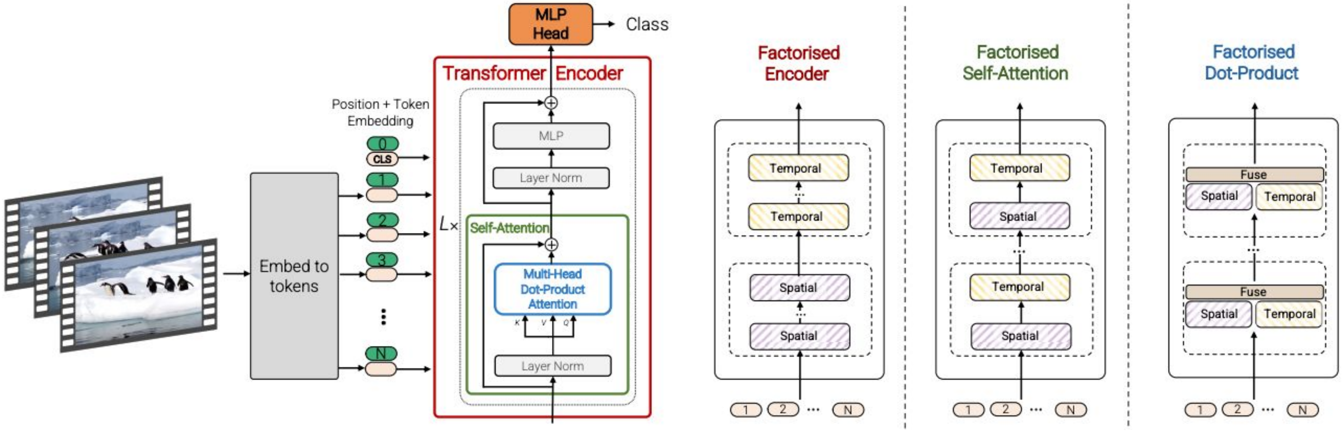


Figure 1: **Self-attention from a Vision Transformer with 8×8 patches trained with no supervision.** We look at the self-attention of the [CLS] token on the heads of the last layer. This token is not attached to any label nor supervision. These maps show that the model automatically learns class-specific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

DINO, Caron et al.

Video



[ViVIT, Arnab et al.]

Outline

Motivation (30-45 minutes)

10 min Break

Designing self-attention models for vision (30-45 minutes)

10 minute break

Brief survey of self-attention in Vision

Universality in deep learning

Universality: developing
components that work
across
all possible settings

Modern deep learning is only partly universal

Universal	Not Universal
• Matrix-vector multiplication	• Mixing primitive
• ReLU	• Data preprocessing
• Residual connections	• Input format
• Maximum likelihood estimation	• Output format
• Parameter initialization	• Data augmentation
• Optimizer	• Feature normalization
• Regularizations	• Hyperparameters

Universality has several core benefits

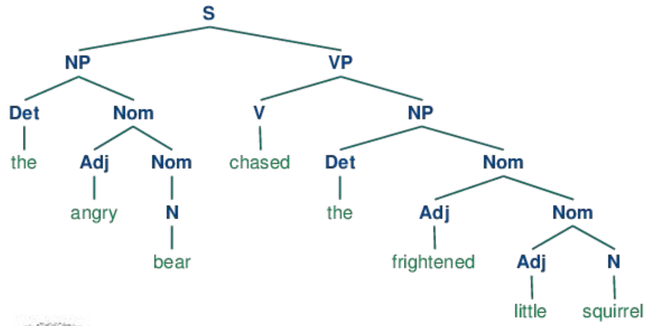
- **Generalization** to new settings
- **Simplicity** of building models
- **Minimizes explicit constraints**, instead preferring to learn from data
- **Large impact** even from small improvements

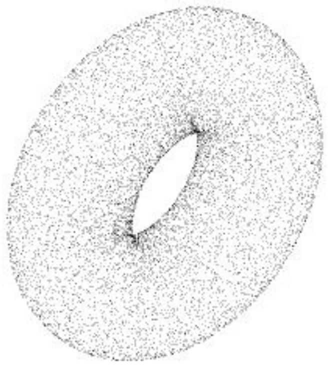
Modern deep learning is only partly universal

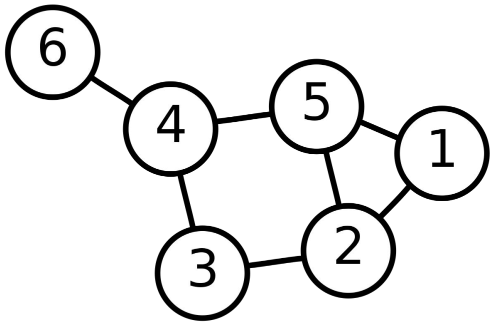
Universal	Not Universal
• Matrix-vector multiplication	• **Mixing primitive**
• ReLU	• Data preprocessing
• Residual connections	• Input format
• Maximum likelihood estimation	• Output format
• Parameter initialization	• Data augmentation
• Optimizer	• Feature normalization
• Regularizations	• Hyperparameters

Our focus: build a universal mixing primitive

- Operations that integrate information across **entities** with **relationships**
- Examples of entities:
  - Words
  - Pixels
  - Points in a cloud
  - Graph vertices
- Examples of relationships:
  - Geometric locality
  - Elements of the same set
  - Graph edges
- Critical for deep learning







Attention is a promising candidate for universality

- **Theoretical:** flexibility to handle many types of data

- **Practical:** efficient mapping to modern hardware

- **Empirical:** scales well to large models and data

Expanding the universe of self-attention

- **Attention** dominates **language**
- **Convolution** dominates (dominated?) **vision**
- Can we bring **attention** to **vision**?

Self-attention: A perspective from langauge

The Deep Learning transformation in language

Learning continuous representations of variable length sequences

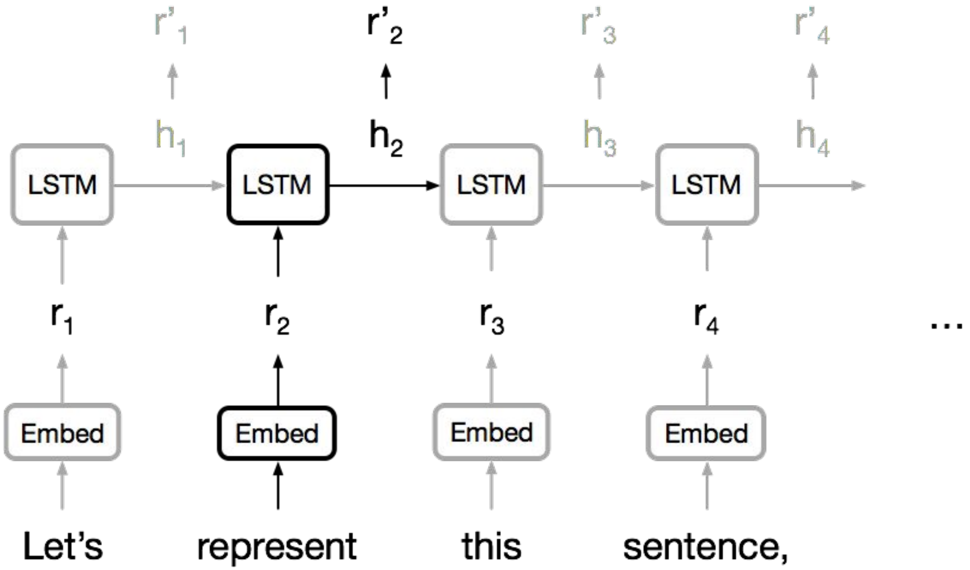
Machine translation, language modeling, summarization, question answering...

RNNs: Sequential models for representation learning

LSTMs, GRUs, Quasi-RNNs...

Advanced state-of-the-art in several NLP tasks.

Recurrent Neural Networks

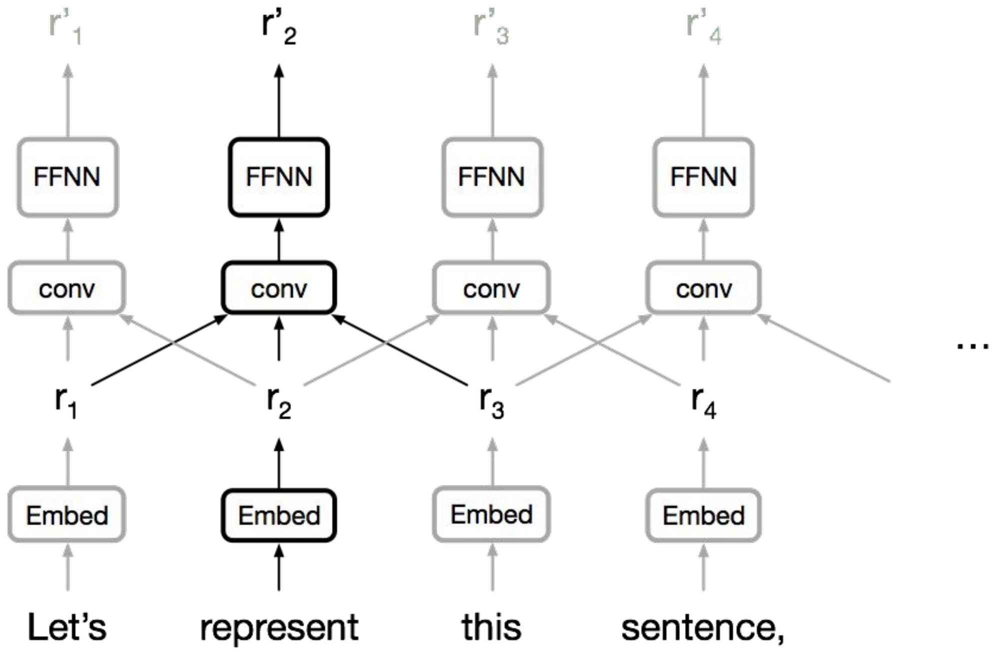


Limitations of RNNs

Computations for over positions cannot be parallelized

Long-range interactions are bottlenecked by a fixed size memory

Convolutional Neural Networks?



Convolutional neural networks for language

Each position can compute representations in parallel per layer

Exploits local dependencies

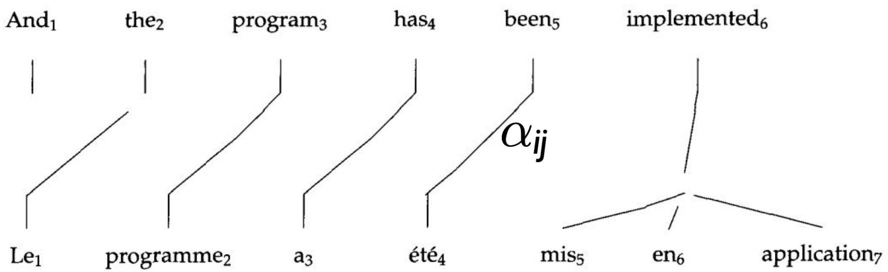
Long-range interactions in linear or logarithmic number of layers.

Attention

Encoder-decoder attention (Bahdanau et al., 2014): Content-based interactions between input and output words

Attention mimics alignments

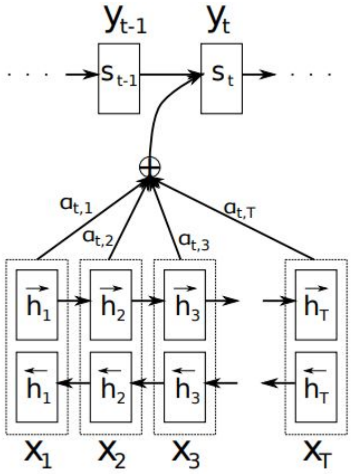
Discrete



$$P(\alpha_{ij} \mid \{e_1, \ldots, e_n\}, \{f_1, \ldots, f_n\})$$

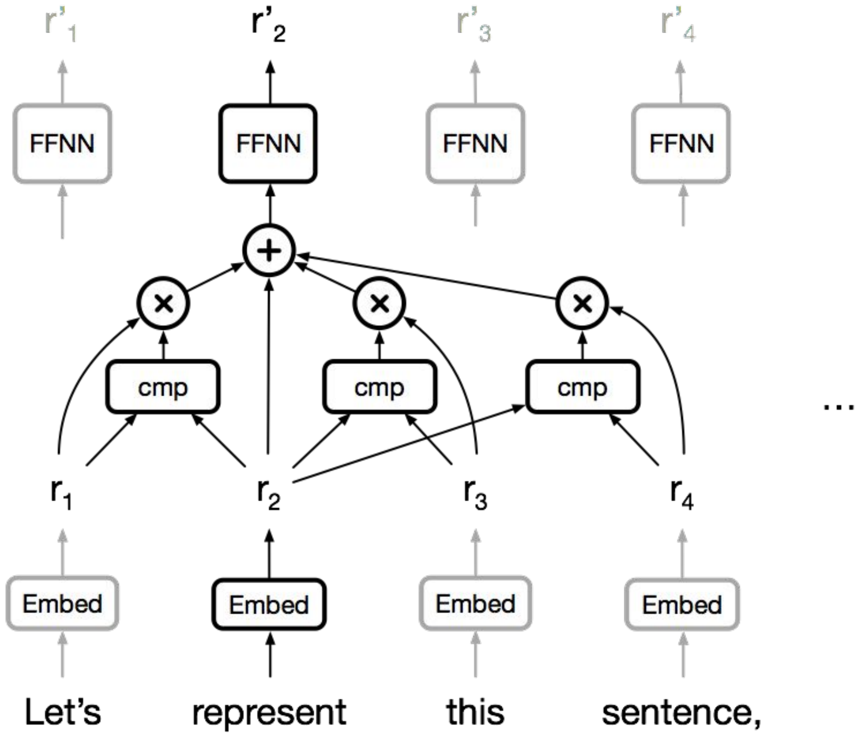
[Brown et al., 1993]()

Continuous



$$\alpha_{ij} = \frac{\exp(s(h_i^x, h_j^y))}{\sum_{i=1}^{T} \exp(s(h_i^x, h_j^y))}$$

Self-Attention



Self-Attention

Single-shot interaction between all-pairs of words

Gating/multiplicative interactions.

Trivial to parallelize (per layer).

Previous work

Classification & regression with self-attention:

Parikh et al. (2016), Lin et al. (2016)

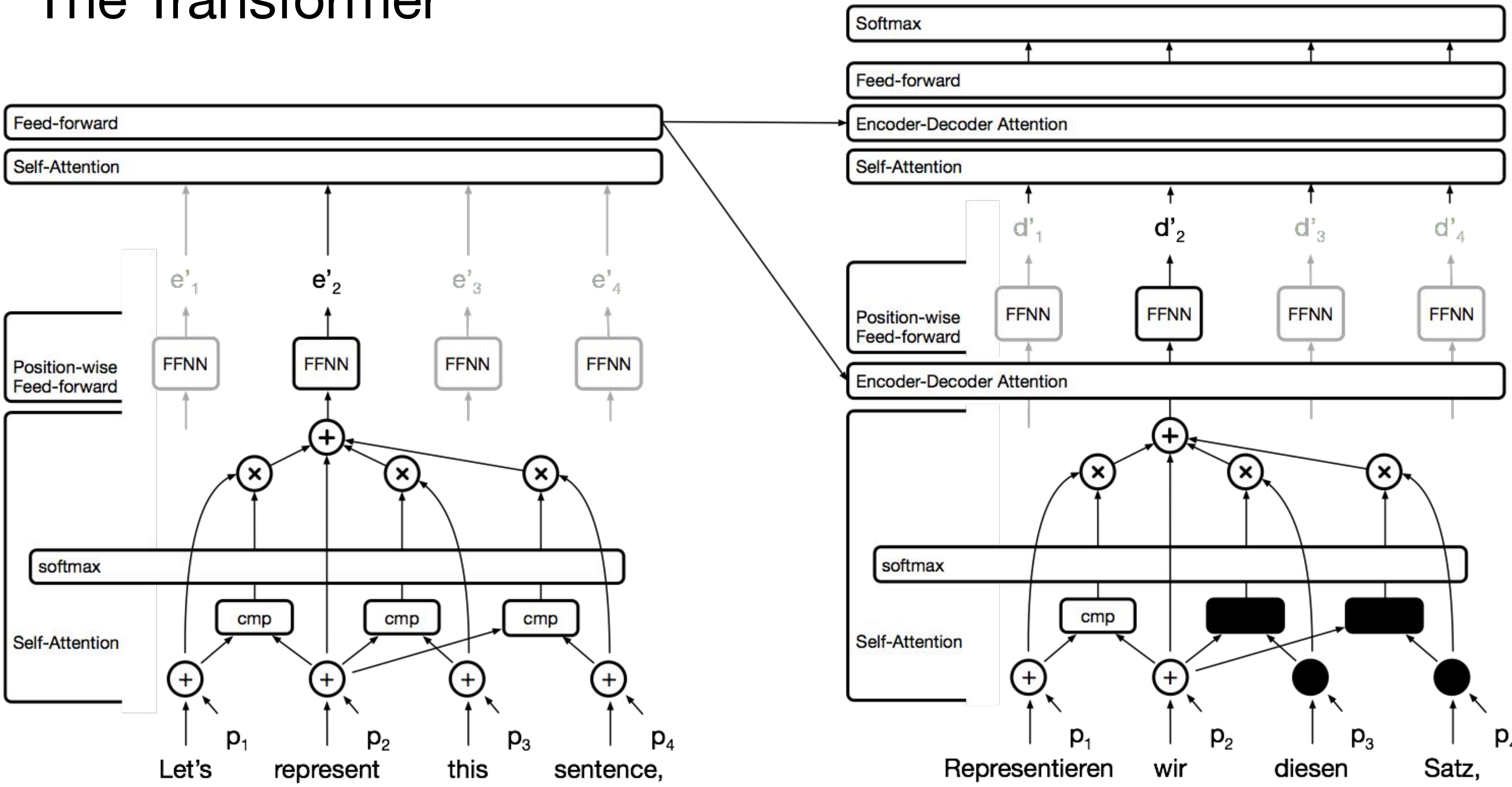
Self-attention with RNNs:

Long et al. (2016), Shao, Gouws et al. (2017)

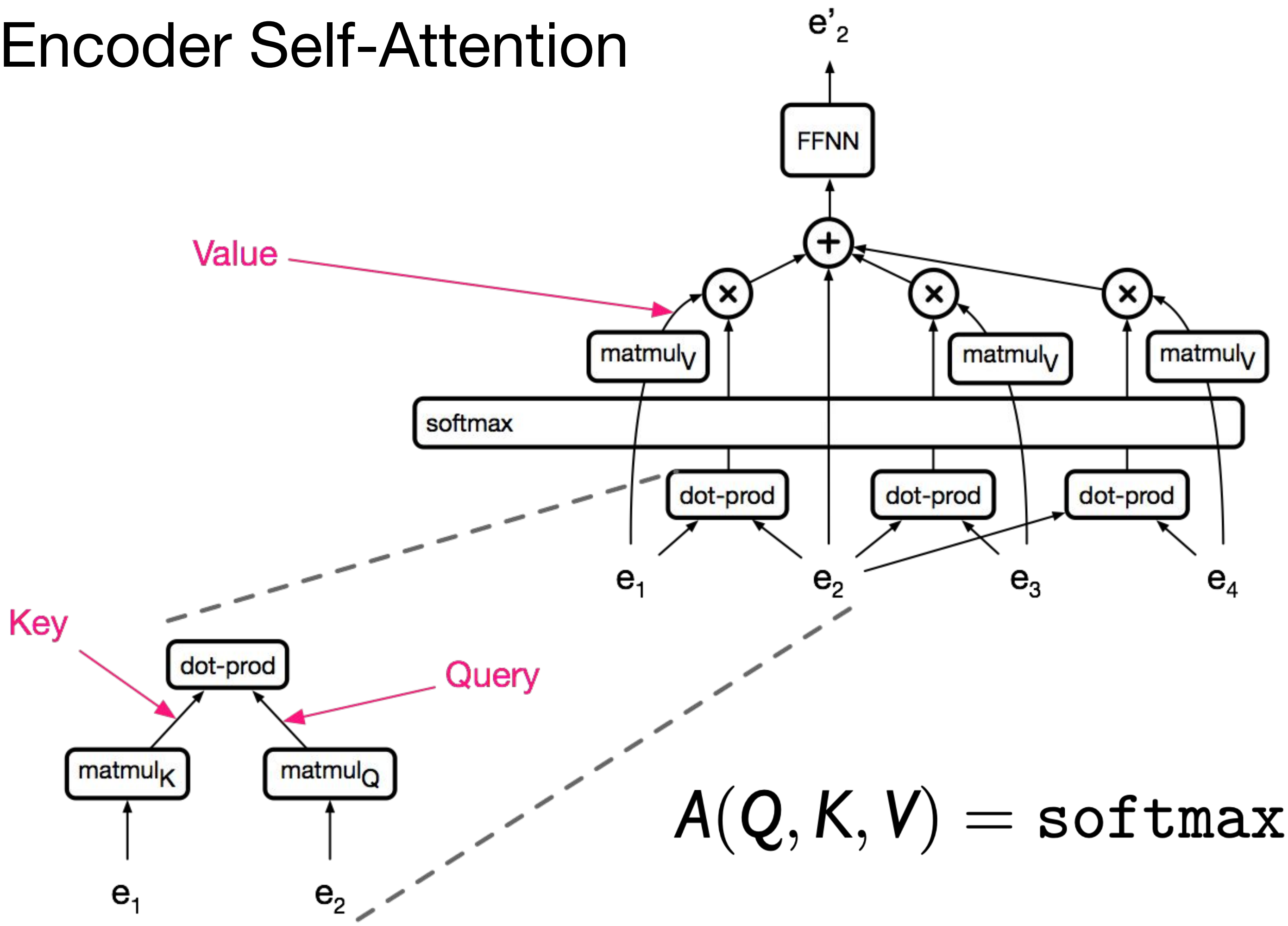
Recurrent attention:

Sukhbaatar et al. (2015)

The Transformer

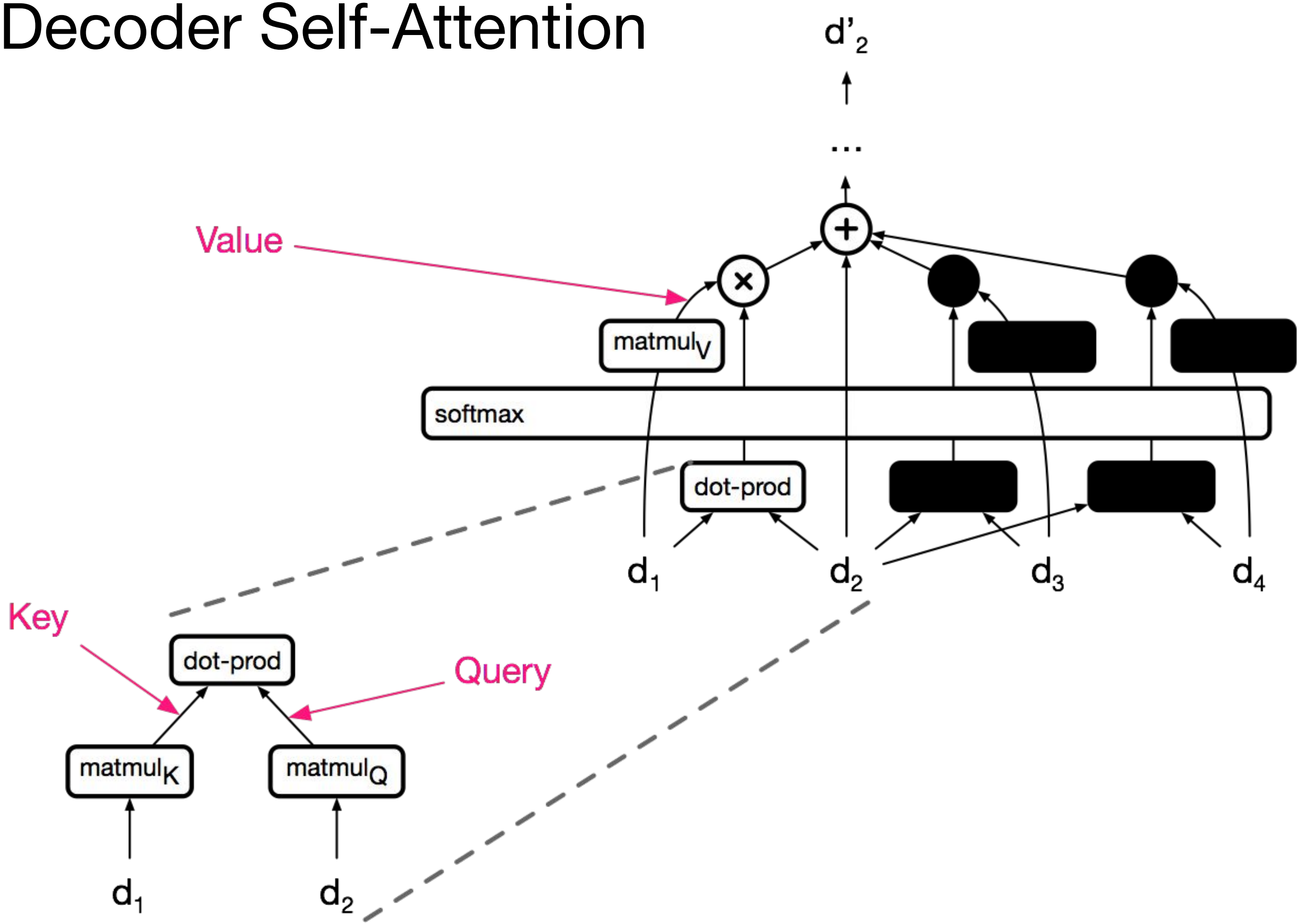


Encoder Self-Attention



$$A(Q, K, V) = \mathtt{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V$$

Decoder Self-Attention



Attention is Cheap!

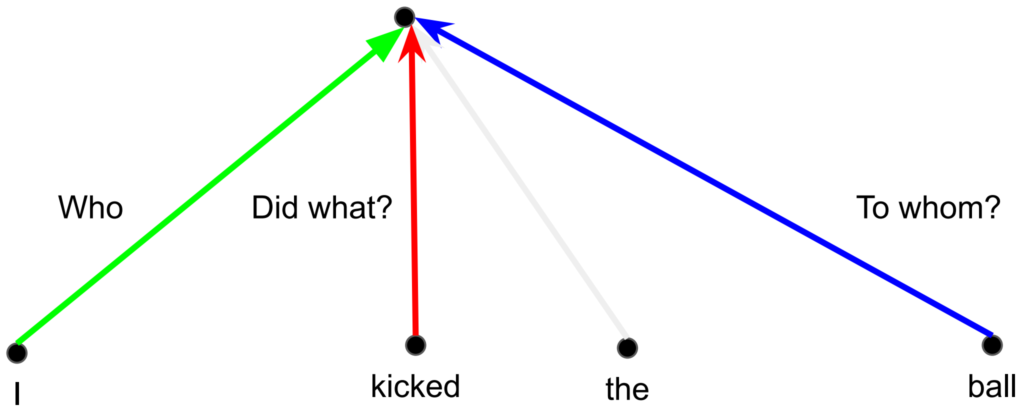
	FLOPs
Self-Attention	$O(\text{length}^2 \cdot \text{dim})$
RNN (LSTM)	$O(\text{length} \cdot \text{dim}^2)$
Convolution	$O(\text{length} \cdot \text{dim}^2 \cdot \text{kernel\_width})$

Attention is Cheap!

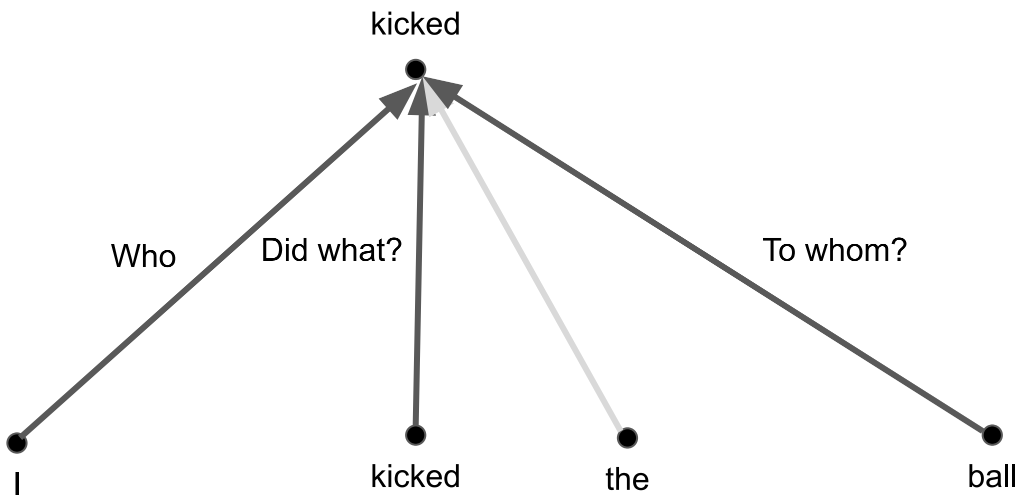
	FLOPs
Self-Attention	$O(\text{length}^2 \cdot \text{dim}) = 4 \cdot 10^9$
RNN (LSTM)	$O(\text{length} \cdot \text{dim}^2) = 16 \cdot 10^9$
Convolution	$O(\text{length} \cdot \text{dim}^2 \cdot \text{kernel\_width}) = 6 \cdot 10^9$

length=1000 dim=1000 kernel_width=3

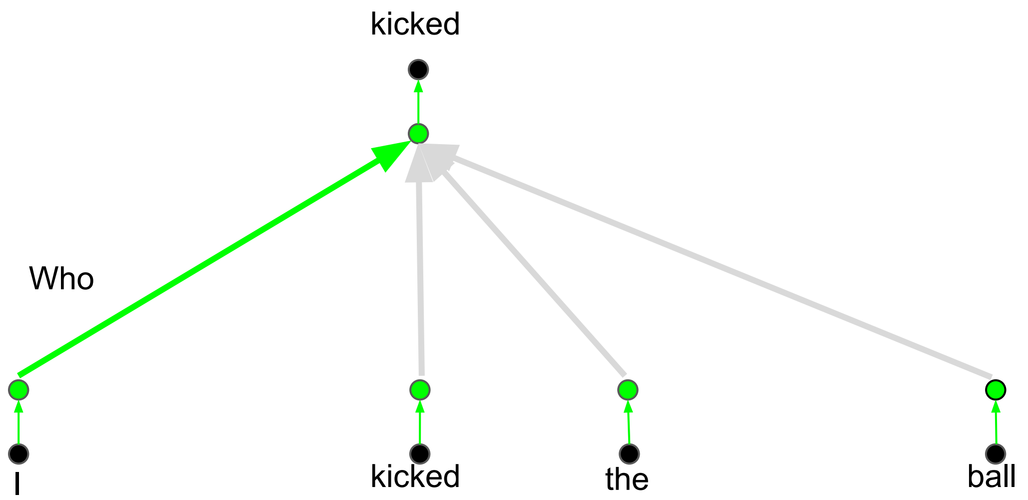
Convolutions



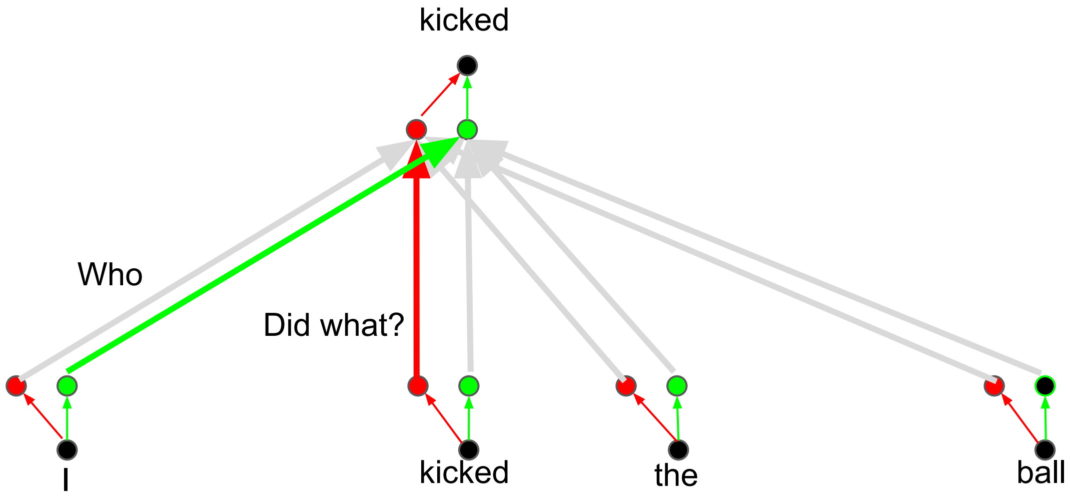
Self-Attention: Averaging



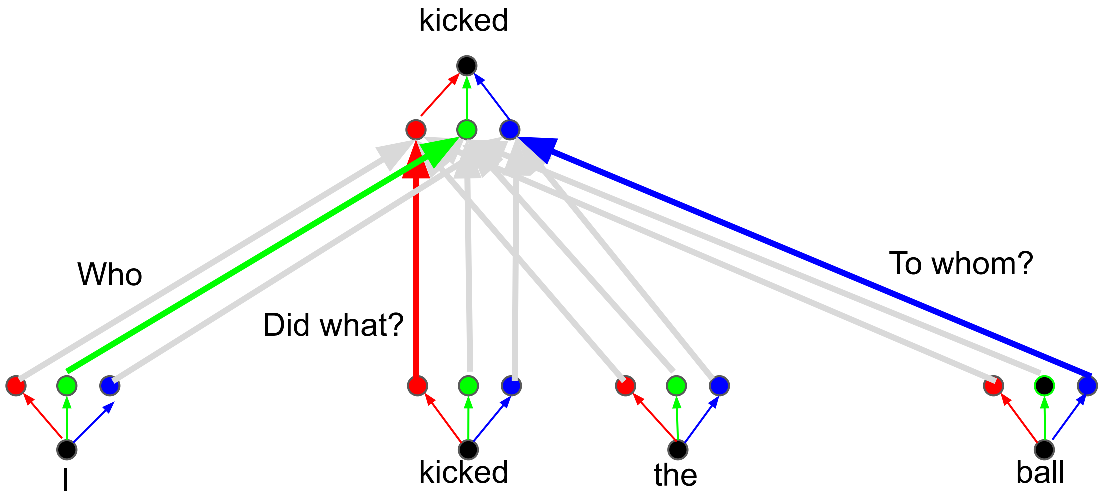
Attention head: Who



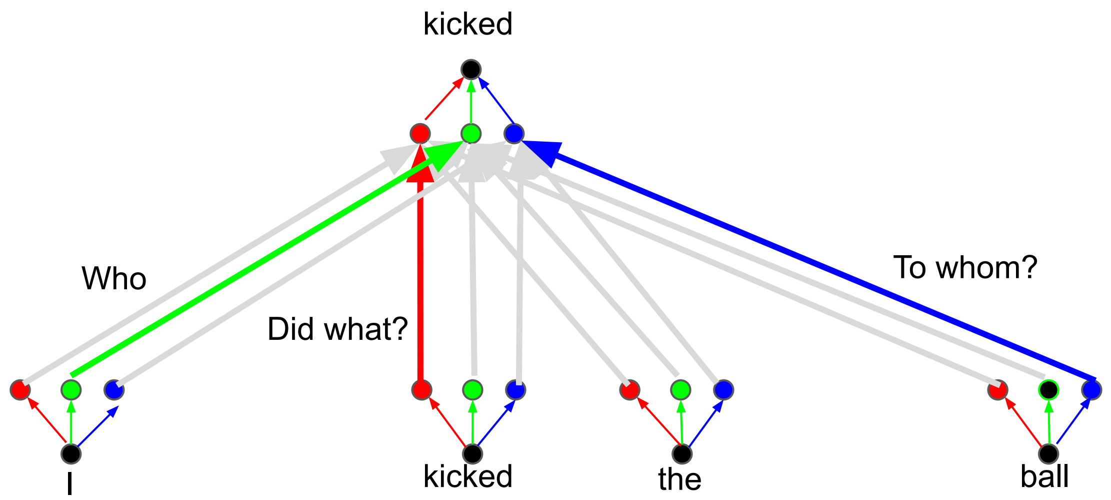
Attention head: Did What?



Attention head: To Whom?



Multihead Attention



Why self-attention for vision?

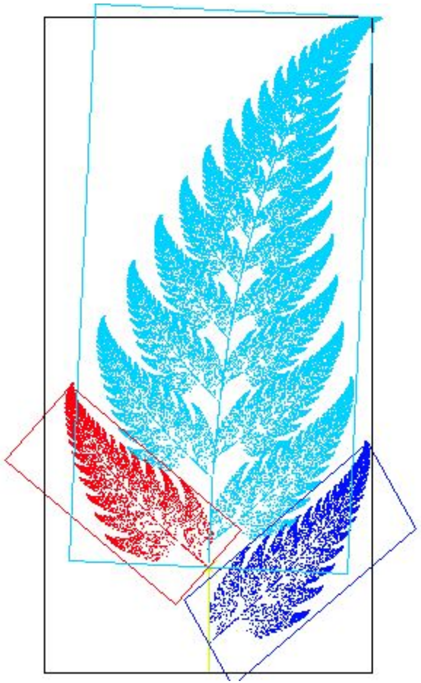
Recap

Modeling long-range interactions between words (pixels).

Useful for longer sentences (images).

Different heads can model different kinds of interactions between words (pixels)

Self-similarity in images



Source

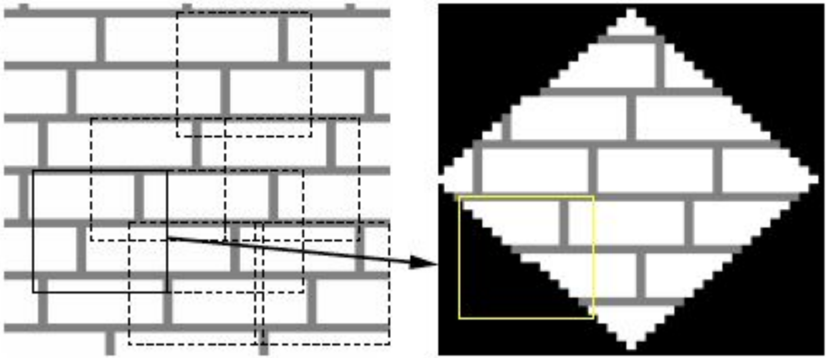
Self-Similarity in Images

Starry Night (Van Gogh, June 1889)

Self-similarity in segmentation

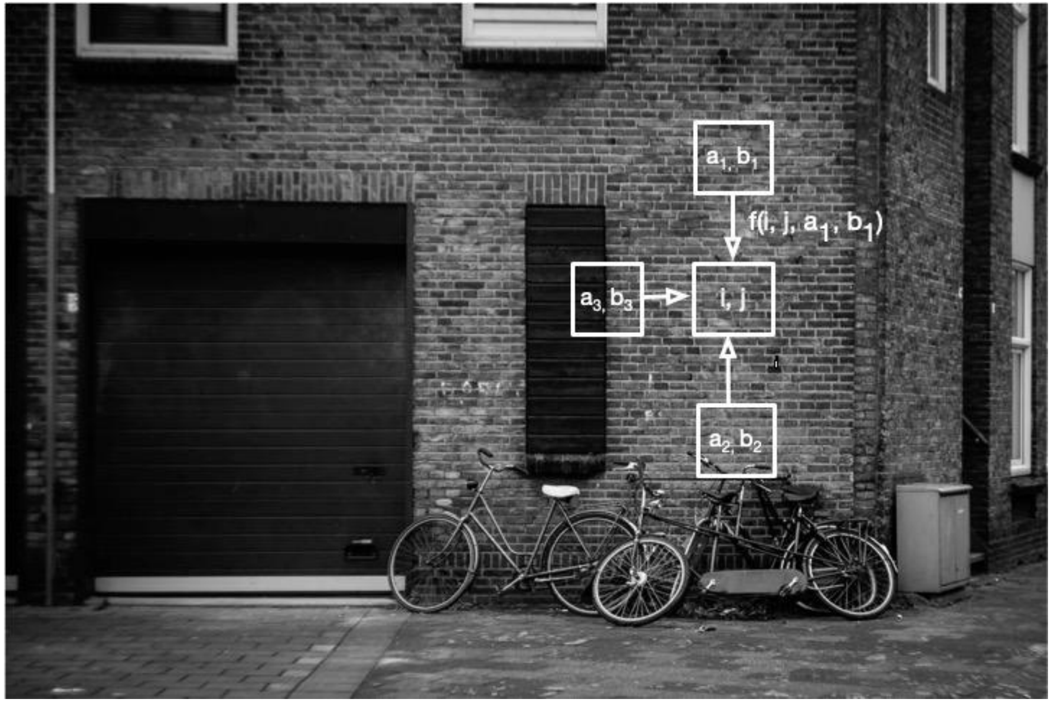
Source

Texture Synthesis with Self-Similarity



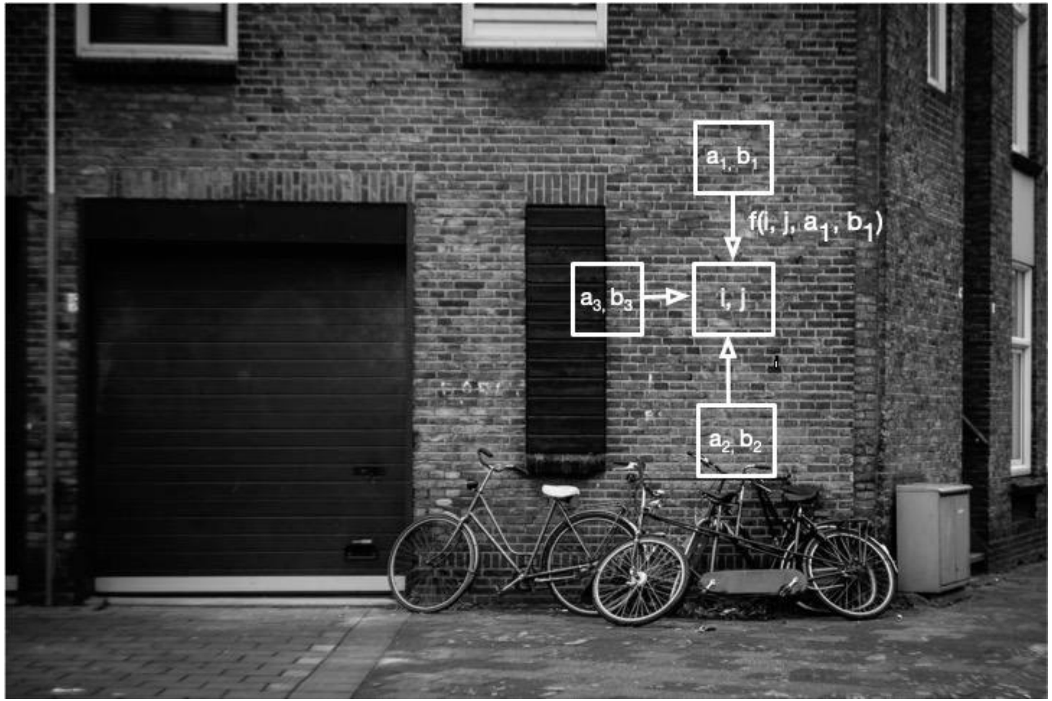
Texture Synthesis by Non-parametric Sampling (Efros and Leung, 1999)

Non-local Means



BCM 2005,
Wang et al., 2018

Non-local Means

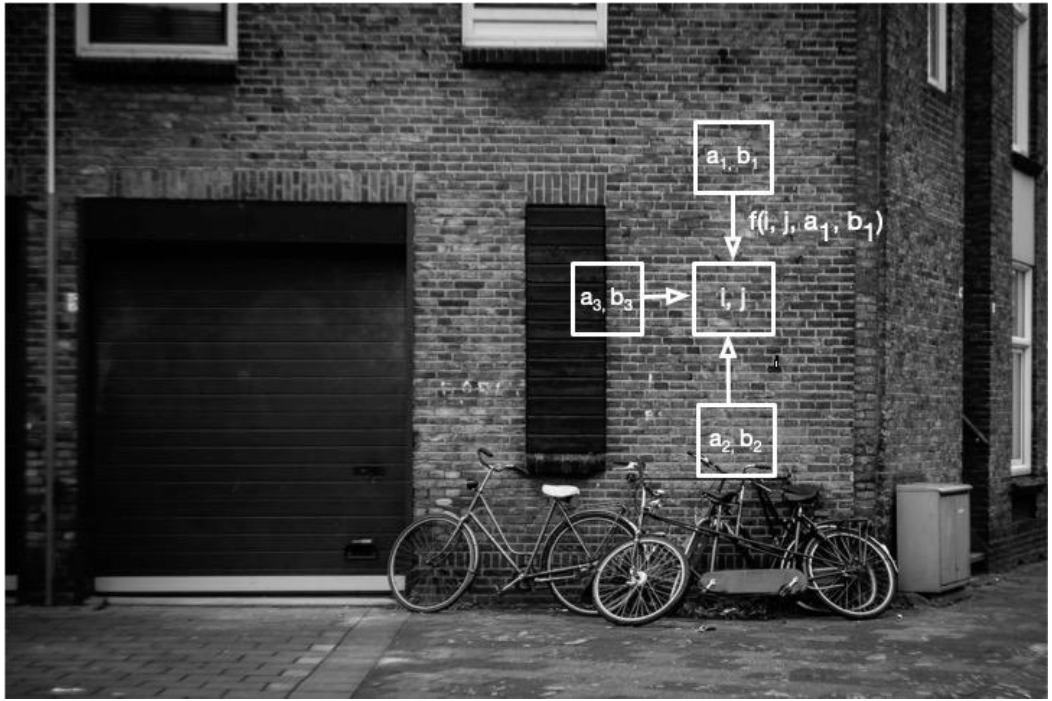


$$y_{ij} = \sum_{a,b \in \mathcal{N}(i,j)} f(i,j,a,b) x_{ab},$$

$$f(i,j,a,b) = \frac{1}{Z(i,j)} e^{-\frac{||x_{ij} - x_{ab}||_2^2}{h^2}}$$

BCM 2005,
Wang et al., 2018

Bilateral filters

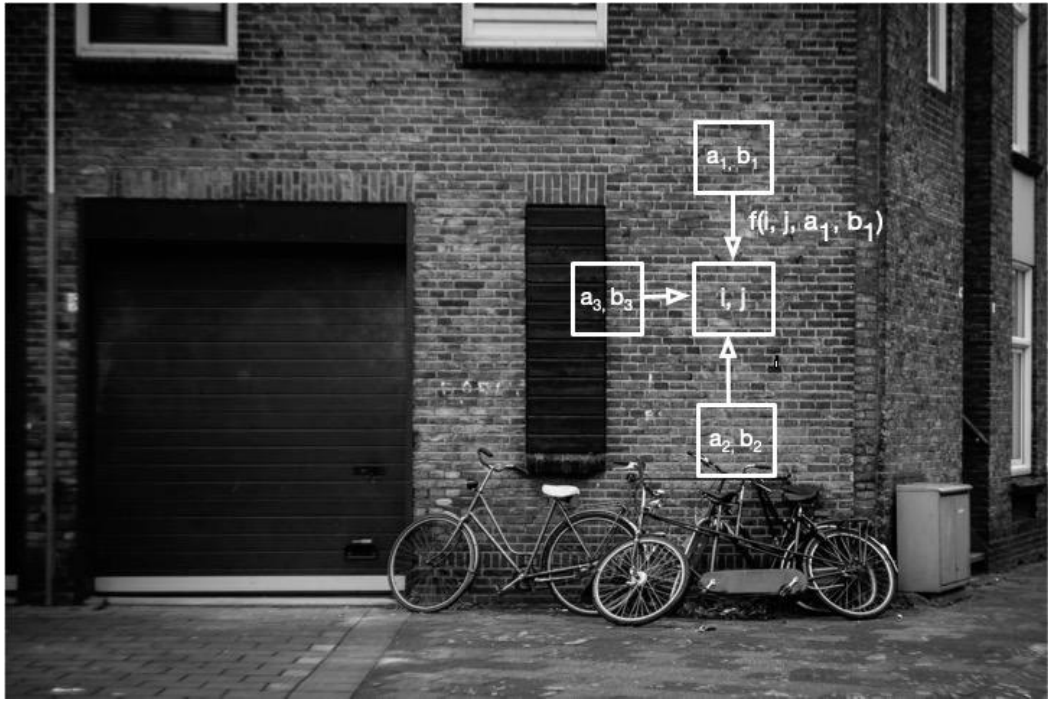


$$y_{ij} = \sum_{a,b \in \mathcal{N}(i,j)} f(i,j,a,b) x_{ab},$$

$$f(i,j,a,b) = \frac{1}{Z(i,j)} e^{-\frac{(i-a)^2 + (j-b)^2}{2\sigma_d^2} - \frac{||x_{ij} - x_{ab}||_2^2}{2\sigma_r^2}}$$

Tomasi and Manduchi, 1998

Self-Attention



$$y_{ij} = \sum_{a,b \in \mathcal{N}(i,j)} f(i,j,a,b) g(x_{ab}),$$

$$f(i,j,a,b) = \frac{1}{Z(i,j)} e^{\left(x_{ij}^\top W_q^\top W_k x_{ab}\right)}$$

$$g(x_{ab}) = W_v x_{ab}$$

Vaswani et al., 2017

Self-attention as a data dependent convolution

$$y_{ij} = \sum_{a,b \in \mathcal{N}(i,j)} f(i,j,a,b) x_{ab},$$

Convolution:

$$f(i,j,a,b) = W_{a-i,j-b}$$

Self-Attention:

$$f(i,j,a,b) = \frac{1}{Z(i,j)} e^{x_{ij}^\top W_q^\top W_k x_{ab}} W_v$$

Takeaways

Self-attention can model long-range interactions between pixels in an image

Self-attention can model the self-similarity within images

Self-attention (without distance information) can be seen a data dependent convolution.

Guidelines for developing an attention-based vision model

- **Build** fully attentional models

- **Reuse** as many vision-designed components as possible
    - Already verified to work for vision
    - Ensures attention is a general operator

- **Replace** all the spatial mixing convolutions with attention

Guidelines for developing an attention-based vision model

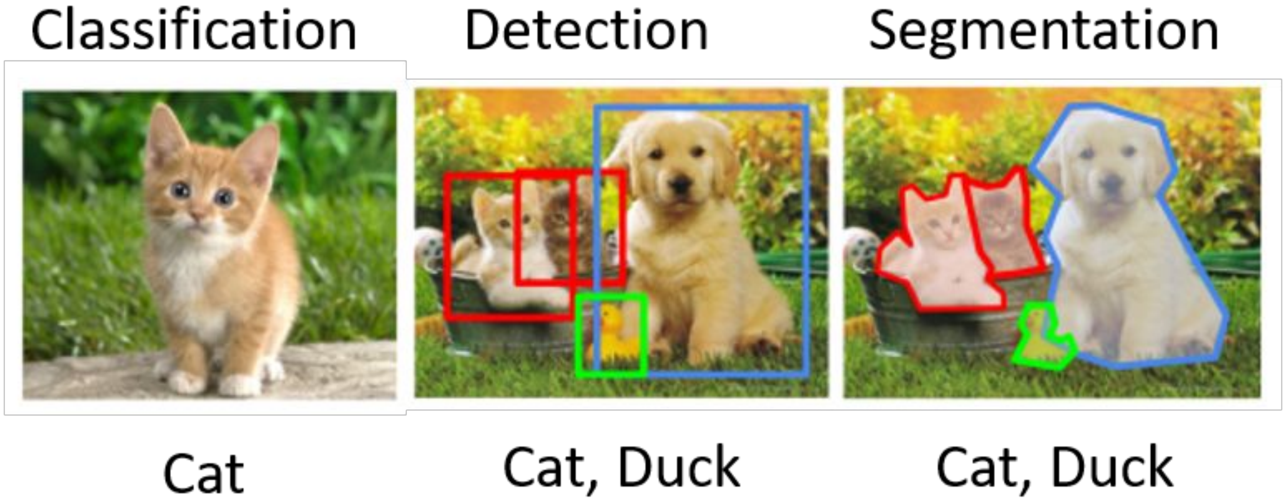
- **Build** fully attentional models

- **Reuse** as many language-designed components as possible
  - Already verified to work for language
  - Ensures attention is a general operator

- **ViT** (Dosovitsky et al.)

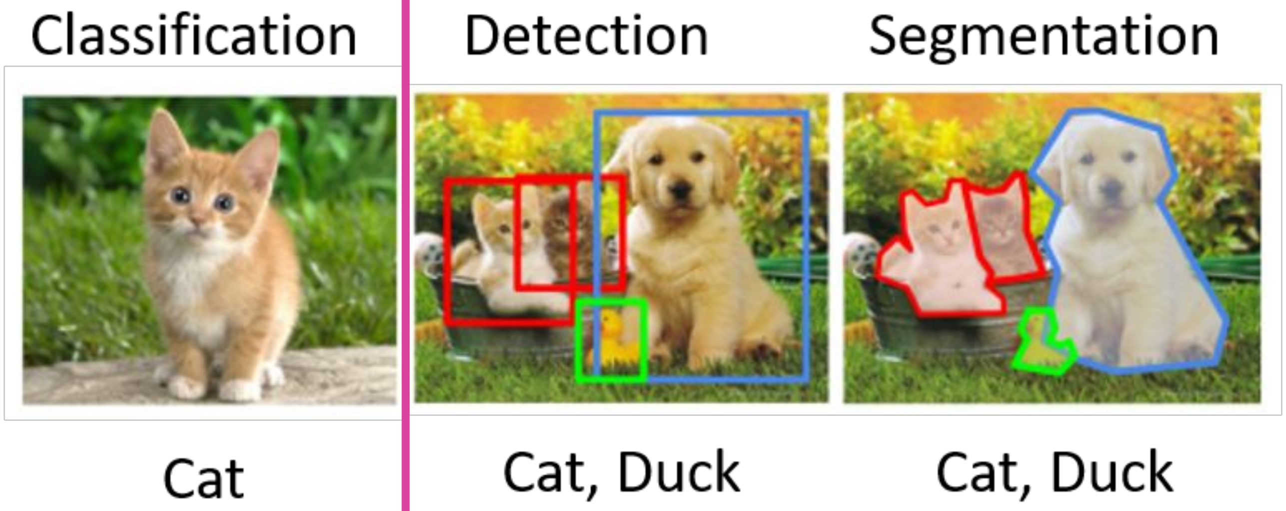
Designing attention models for vision

Vision tasks



Source

Let’s focus on classification for now



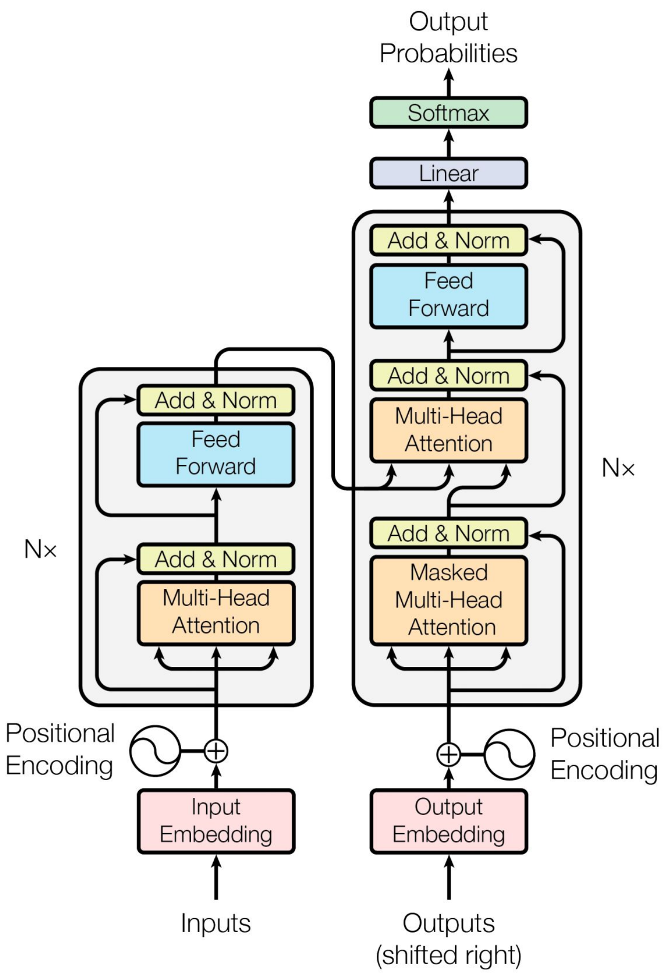
Source

How do we design vision attention models?

ML design philosophy:

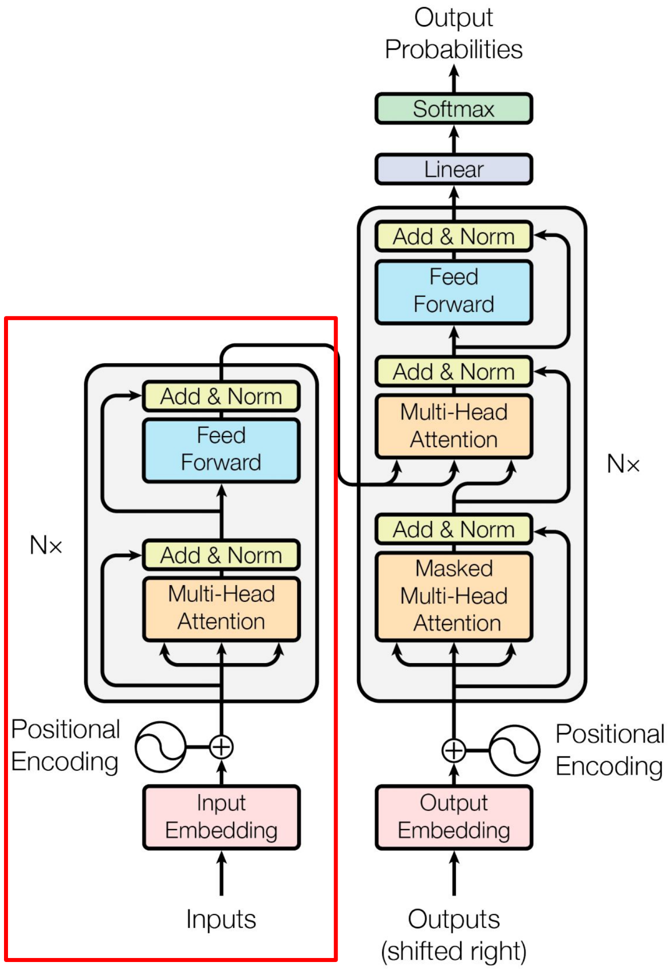
Adapt a pre-existing model.

Transformer



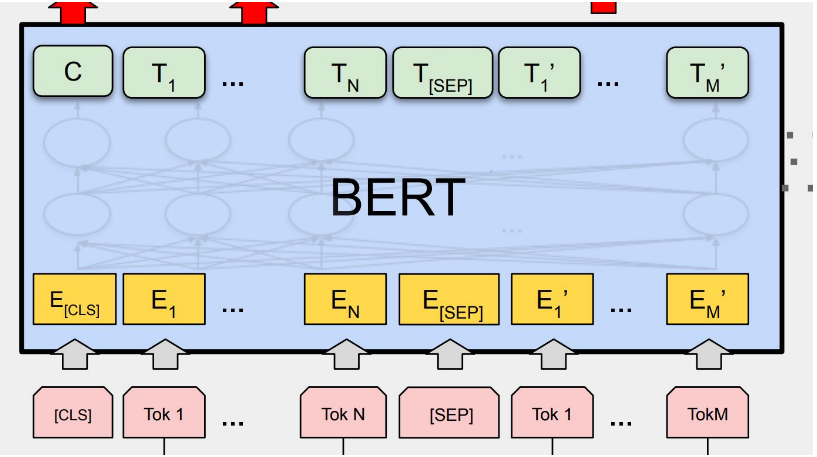
[Vaswani et al. 2017]

Transformer Encoder



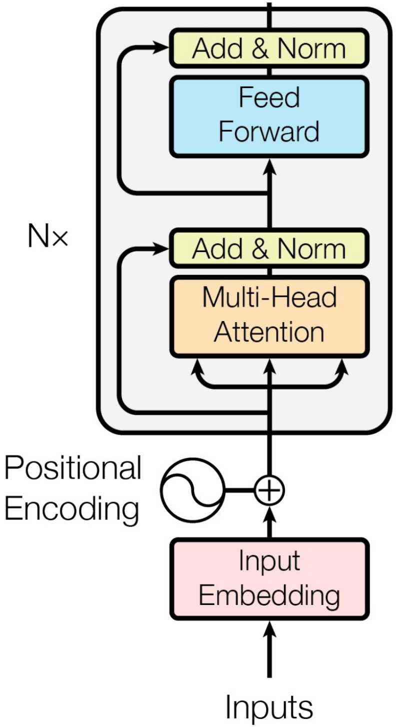
Vaswani et al. 2017

Transformers for NLP



Devlin et al. 2019

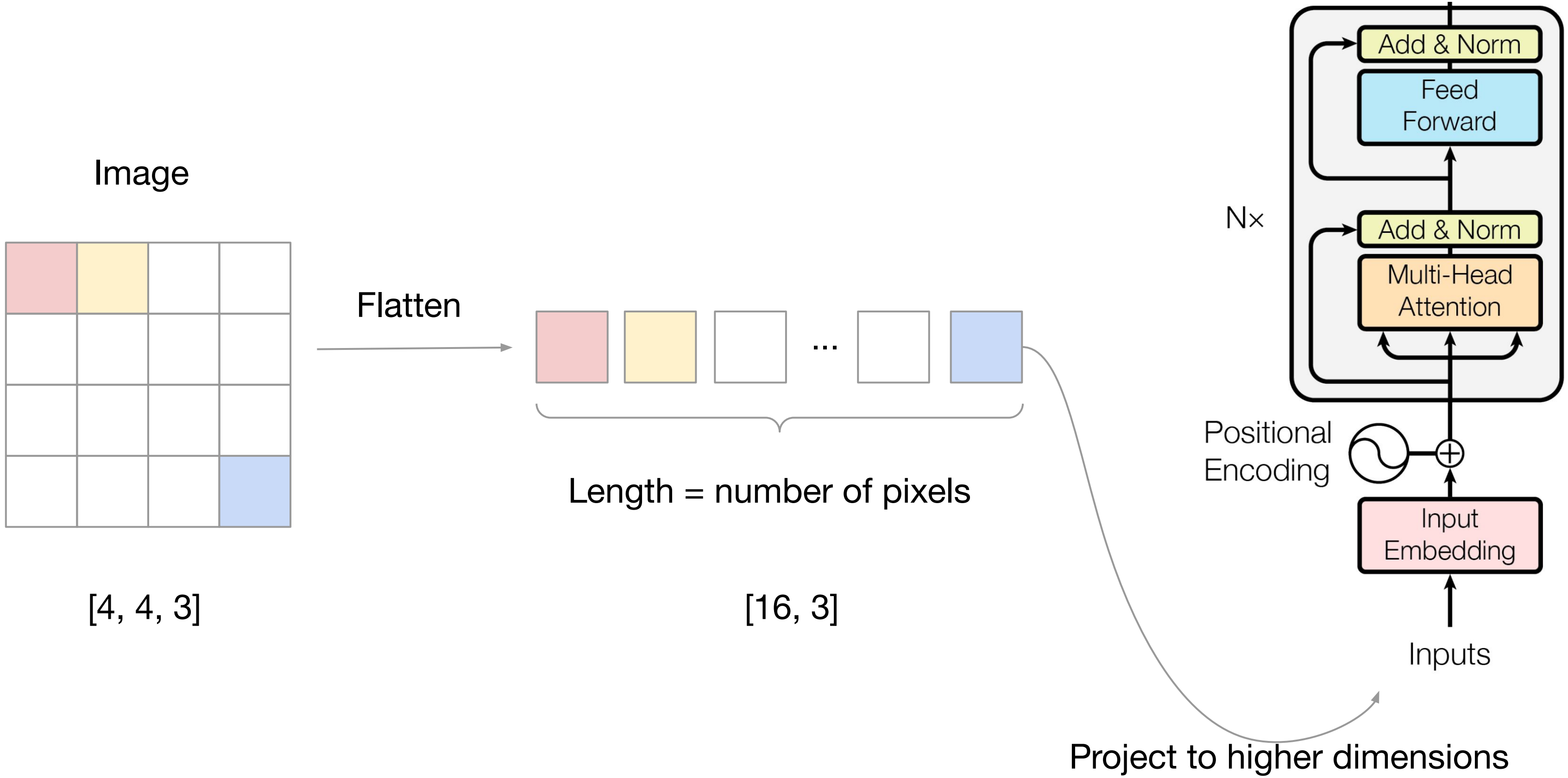
Computation in Transformer



Vaswani et al. 2017

Idea: treat each pixel as a token, and pass to a Transformer

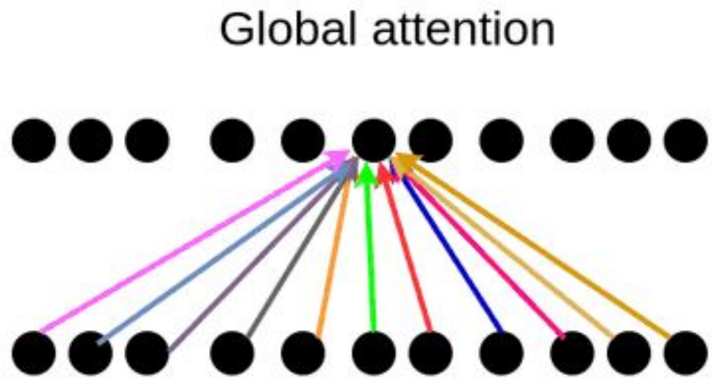
Idea: treat each pixel as a token, and pass to Transformer



Problem: it’s too expensive!

- For a 224x224 images, there ~50K pixels
- Attention cost scales quadratically with the input length

Controlling cost is the perennial problem of attention



n^2 time and memory cost!

Source

Problem: it's too expensive!

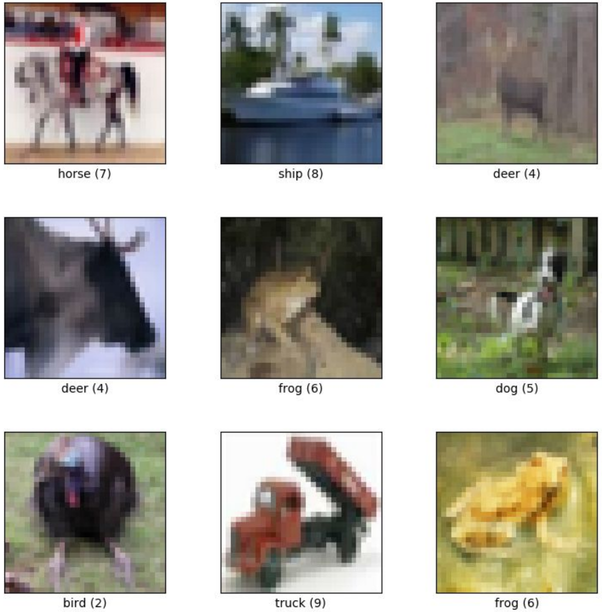
- For a 224x224 images, there ~50K pixels
- Attention cost scales quadratically with the input length
    - $50000^2 \rightarrow$ too large

Idea: use a smaller image size

- Will reduce the input length, which makes the Transformer cheaper

Source

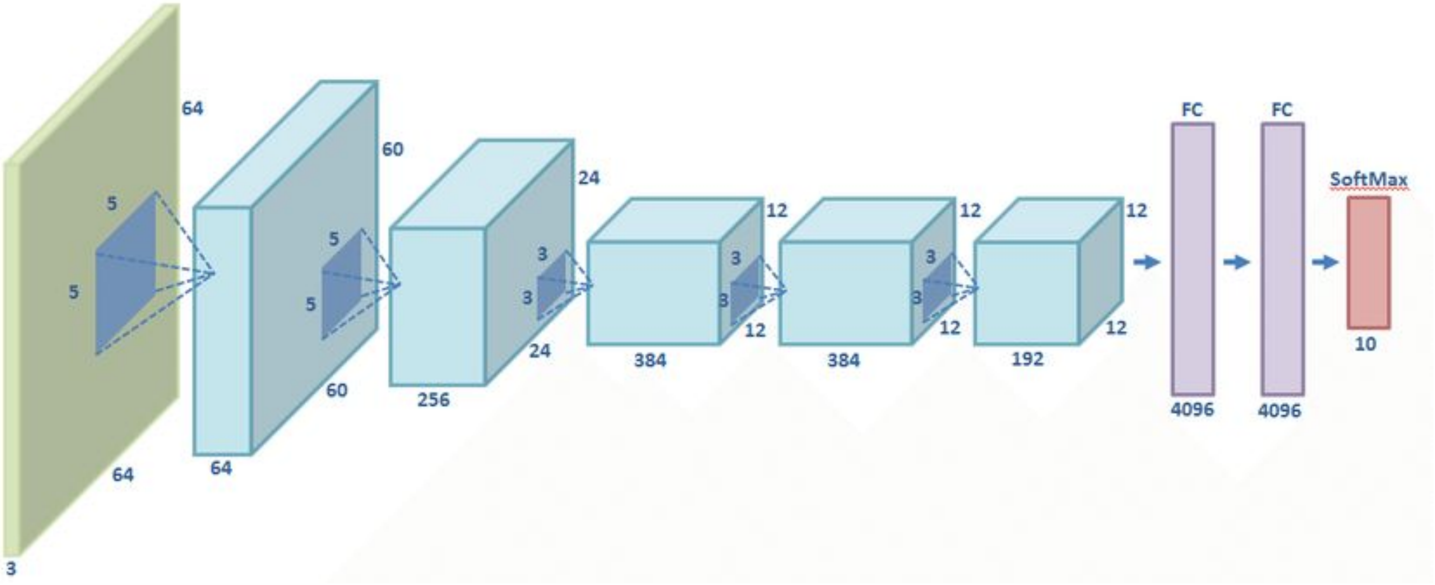
Problem: loses a lot of detail



Source

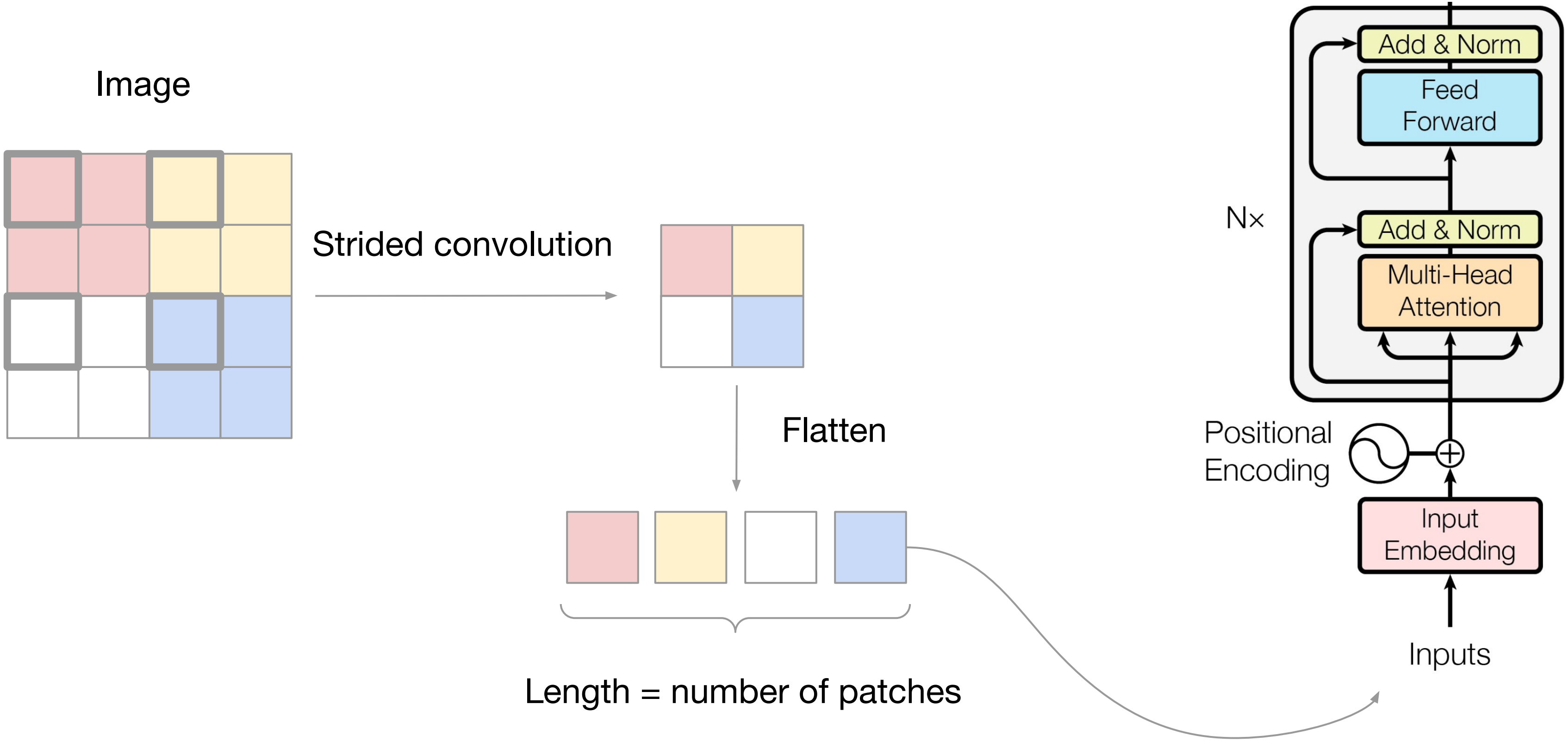
Idea: learnable downsampling of the image

- Model can learn to store important information in the features
- Similar to CNNs:

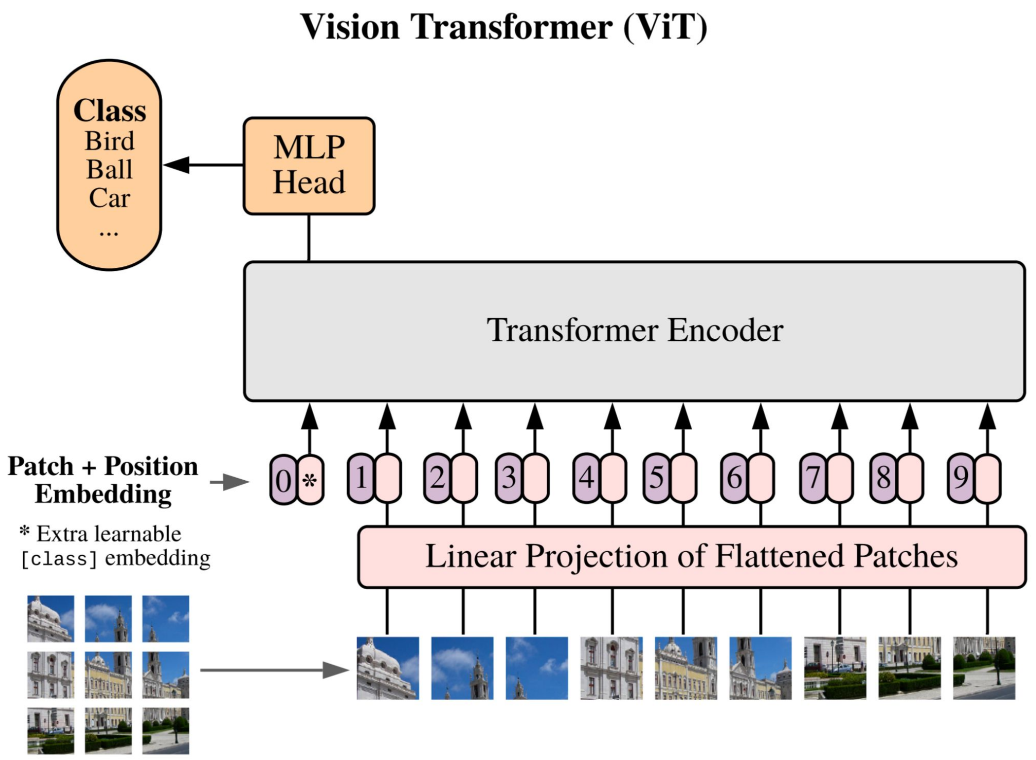


Source

Idea: learnable downsampling

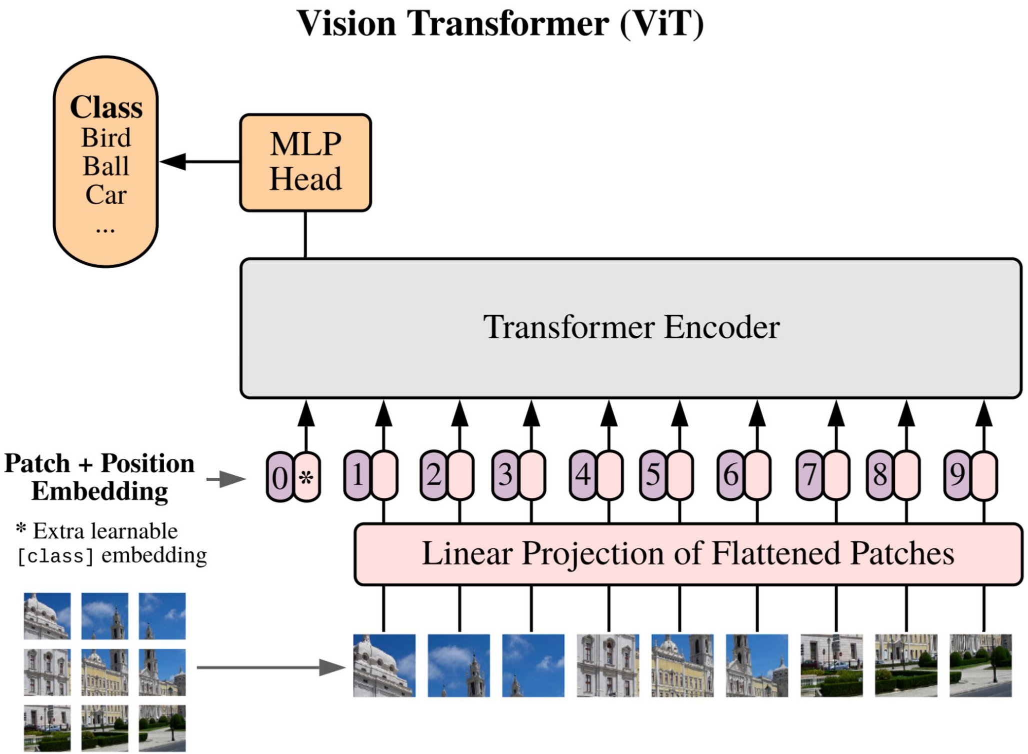


Vision Transformer



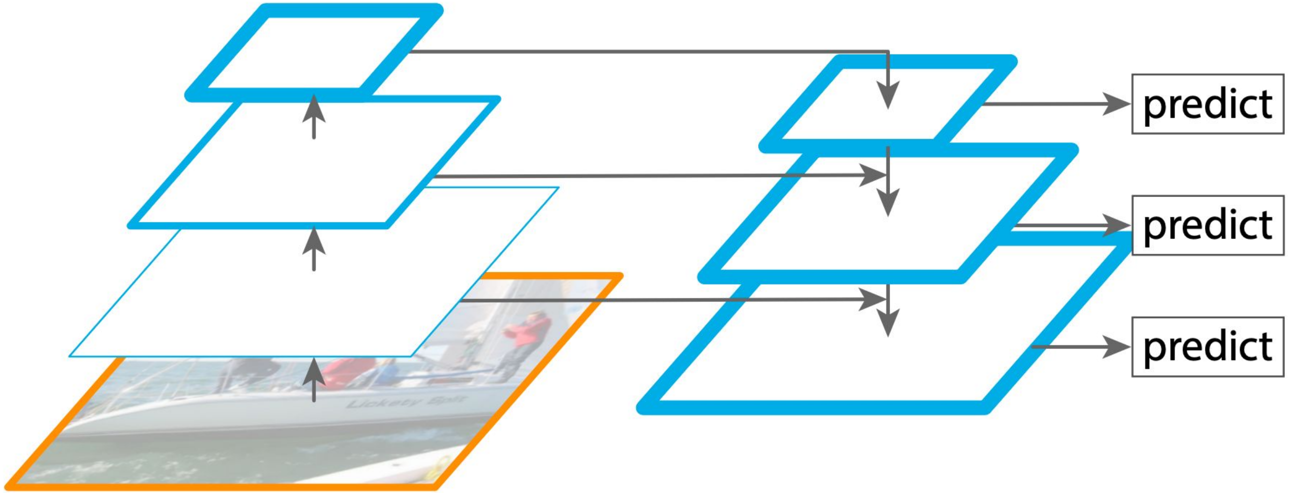
[Dosovitskiy et al. 2020]

Only a single scale achievable



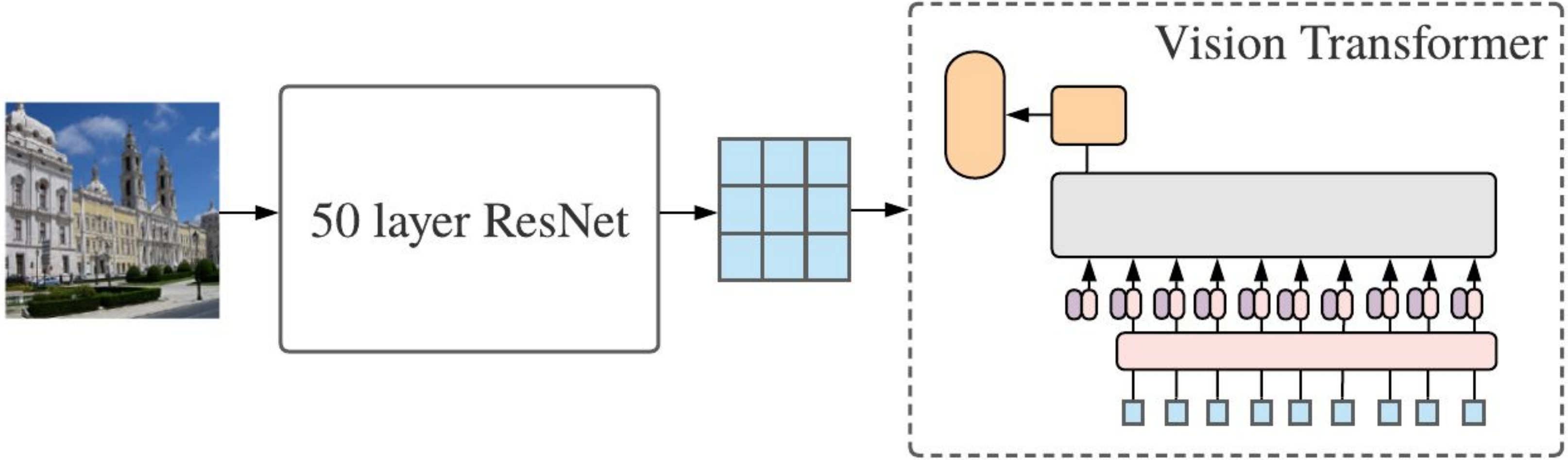
Dosovitskiy et al. 2020

How to get multi-scale features?



Lin et al. 2017

Replace strided convolution with CNN

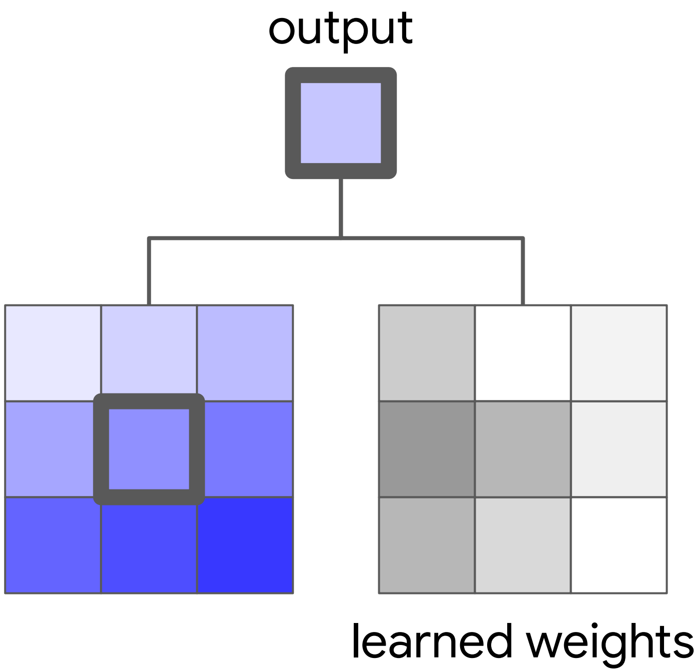


Credit: Neil Housby, Alexey Dosovitskiy

Hybrid CNN-Transformers

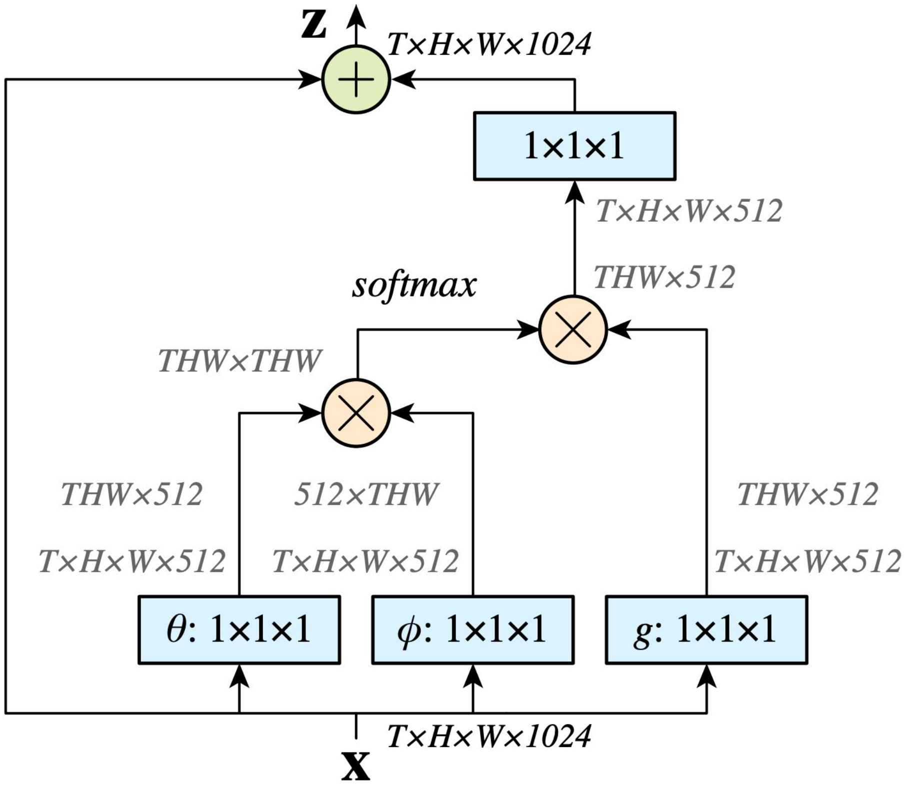
- Convolutions applied to larger resolutions
    - Linear scaling
- Attention applied to lower resolutions
    - Quadratic scaling, but okay since few pixels

Convolution: linear scaling



Ramachandran et al. 2019

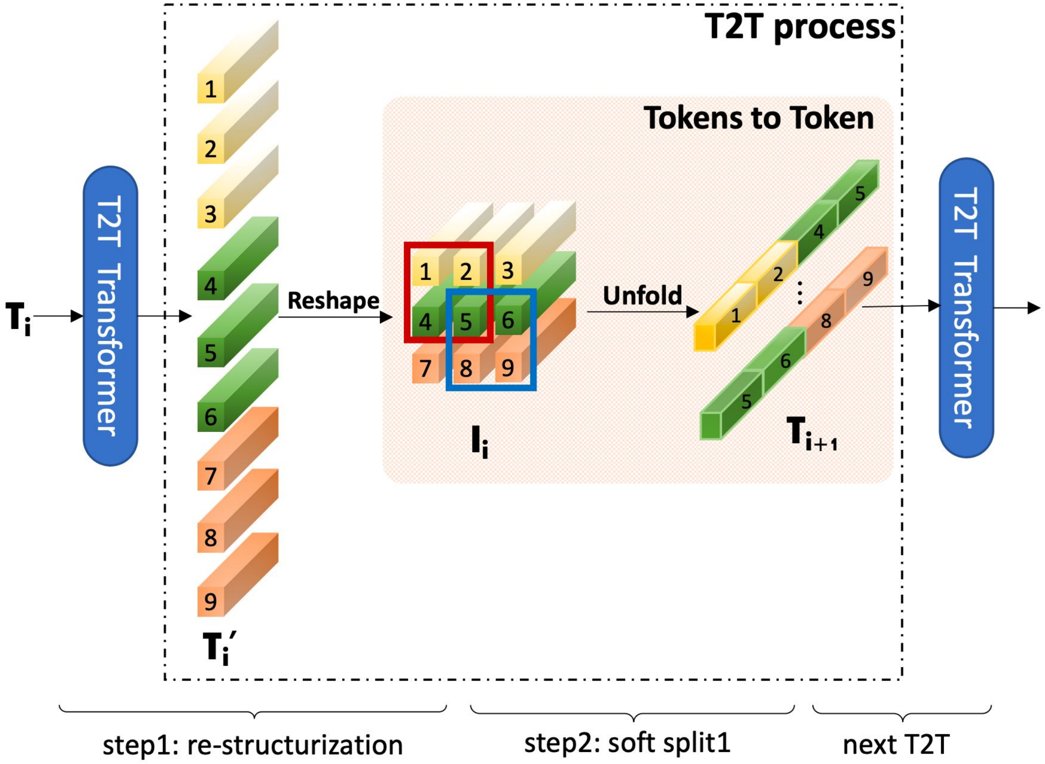
Non-local Networks



layer name	output size	50-layer
conv1	112×112	7×7, 64, stride 2
conv2_x	56×56	3×3 max pool, stride 2 $\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$
conv4_x	14×14	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$
conv5_x	7×7	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$

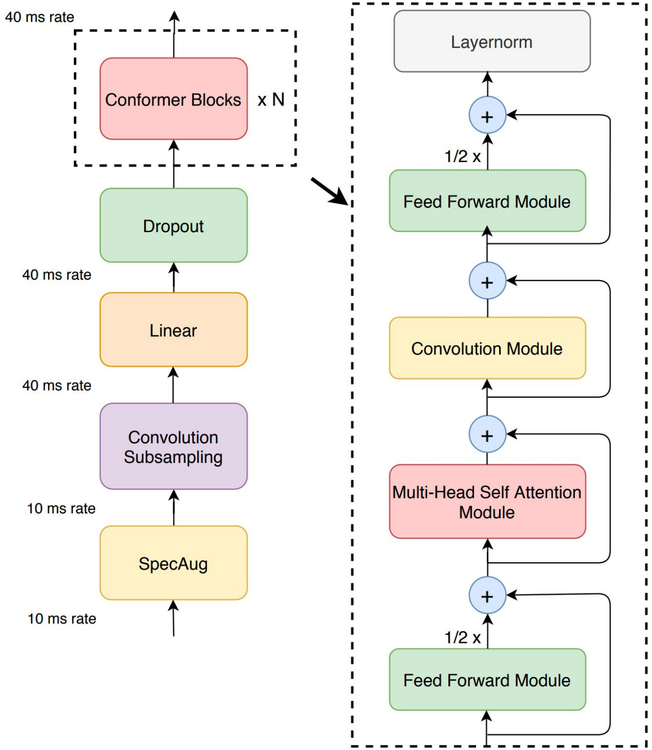
Wang et al. 2018

Different attention scales with downsampling



Yuan et al. 2021

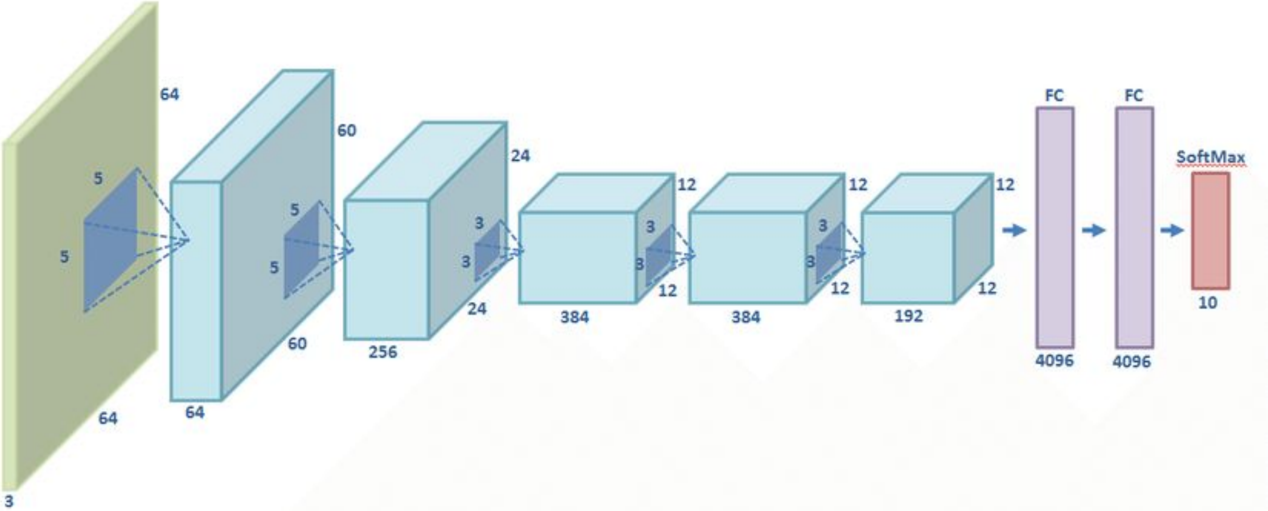
Hybrid convolution-attention in speech understanding



Gulati et al. 2020

Applying attention to larger resolutions

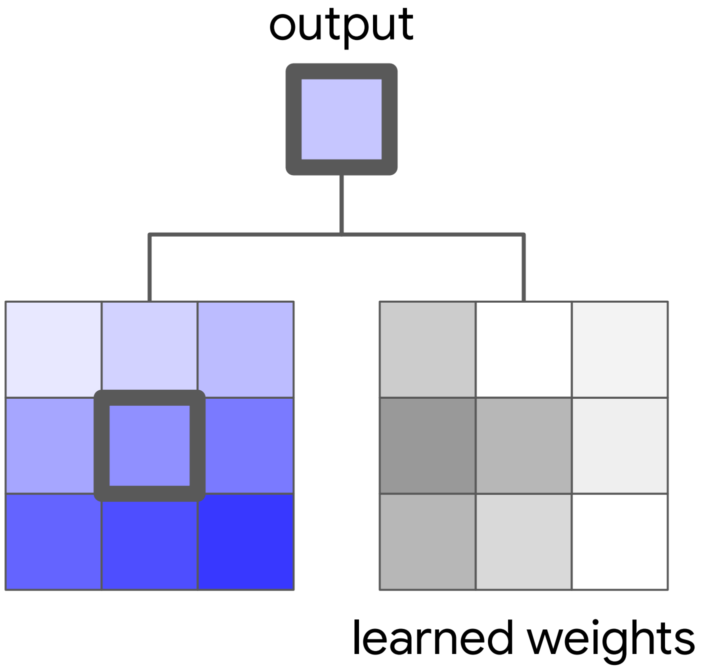
- Larger resolutions important for localization tasks
- Convolutions can be efficiently applied to larger resolutions
- How can attention be adapted for larger resolutions?



Core idea: make attention cheaper

- Larger resolutions have more pixels
- Quadratic complexity of attention is too expensive with many pixels
- Convolution linear in the number of pixels
- Try to make attention cost more linear

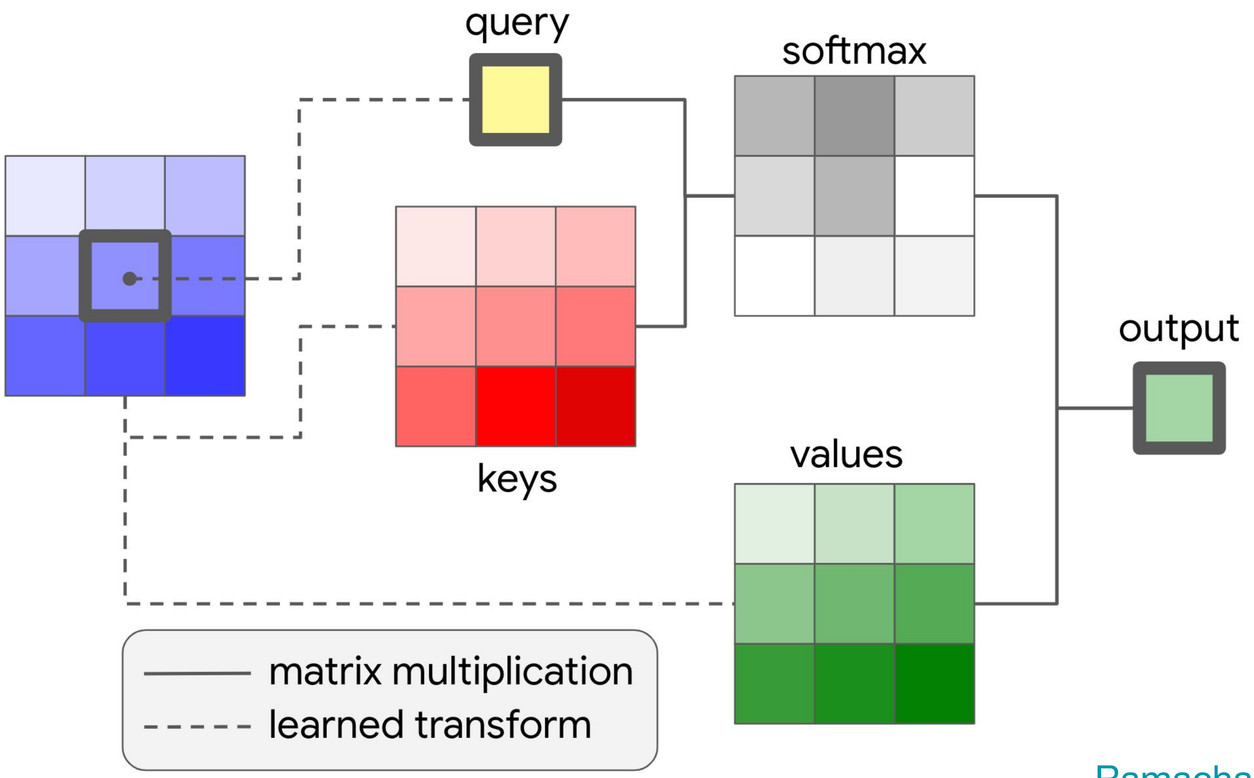
Convolution: linear transform of local window



$$y_{ij} = \sum_{a,b \in \mathcal{N}_k(i,j)} W_{i-a,j-b} \, x_{ab}$$

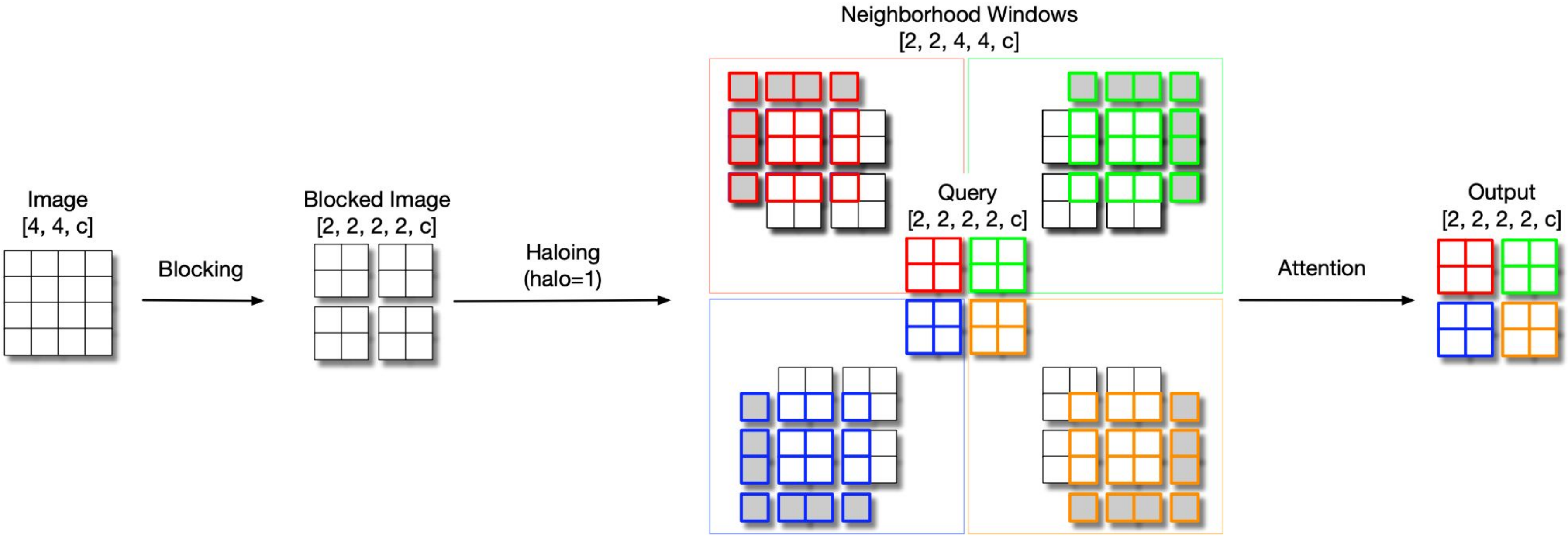
Ramachandran et al. 2019

SASA: Local attention



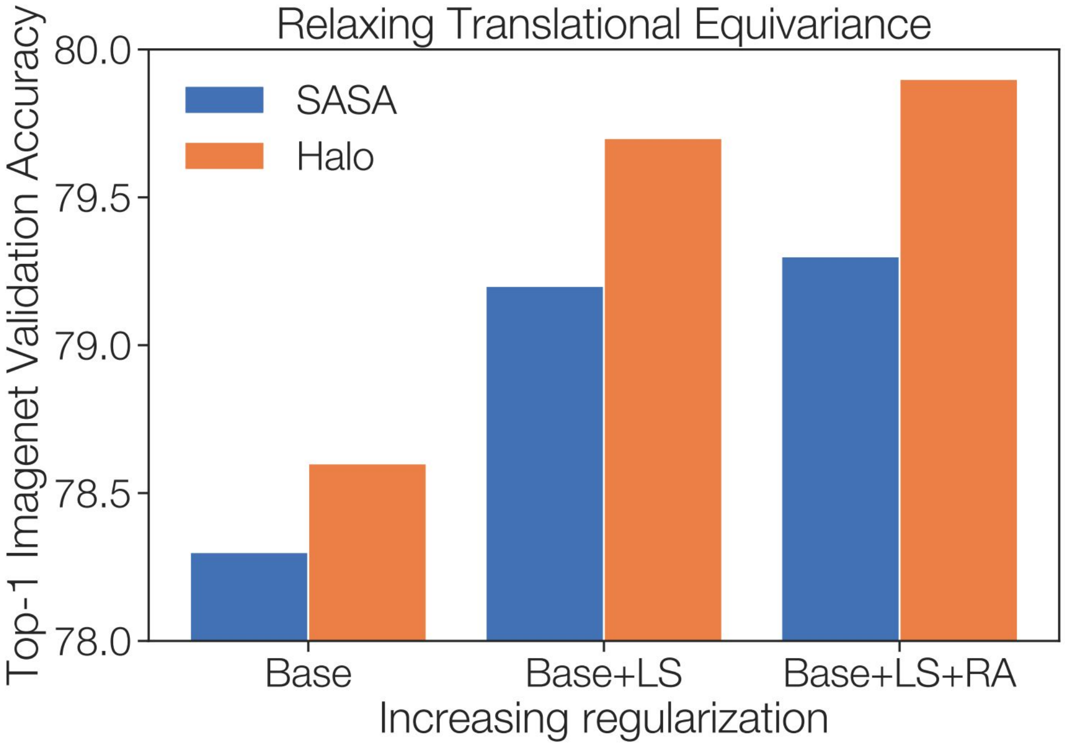
[Ramachandran et al. 2019]

HaloNet: Blocked local attention



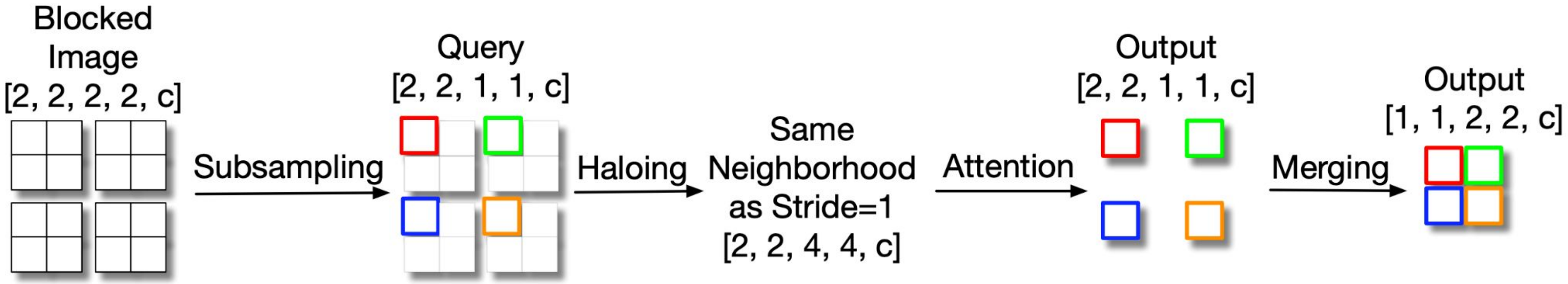
[Vaswani et al. 2021]

HaloNet: Blocking improves speed & accuracy



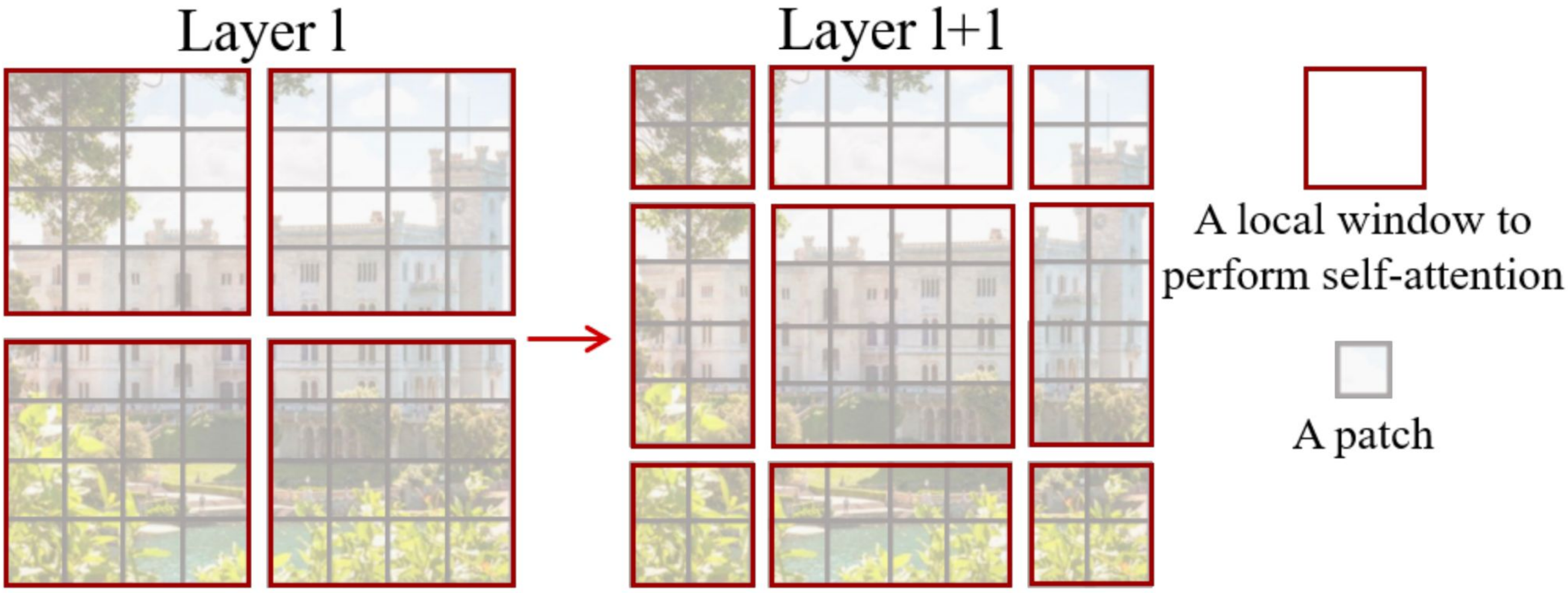
[Vaswani et al. 2021]

HaloNet: striding with local attention



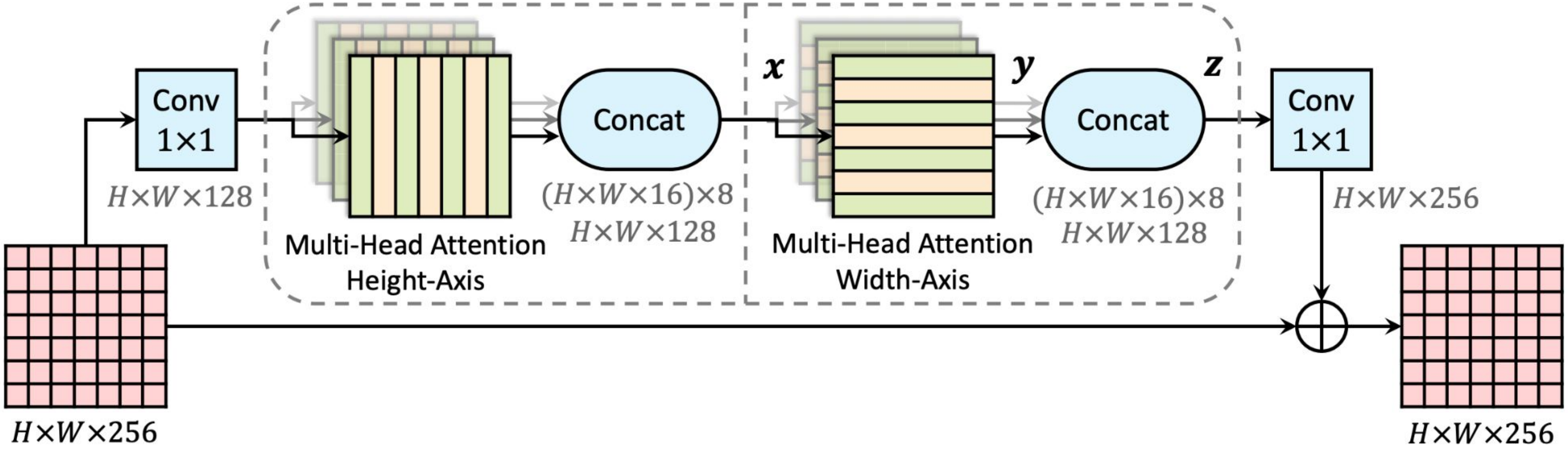
Vaswani et al. 2021

Swin Transformer: Shifted window



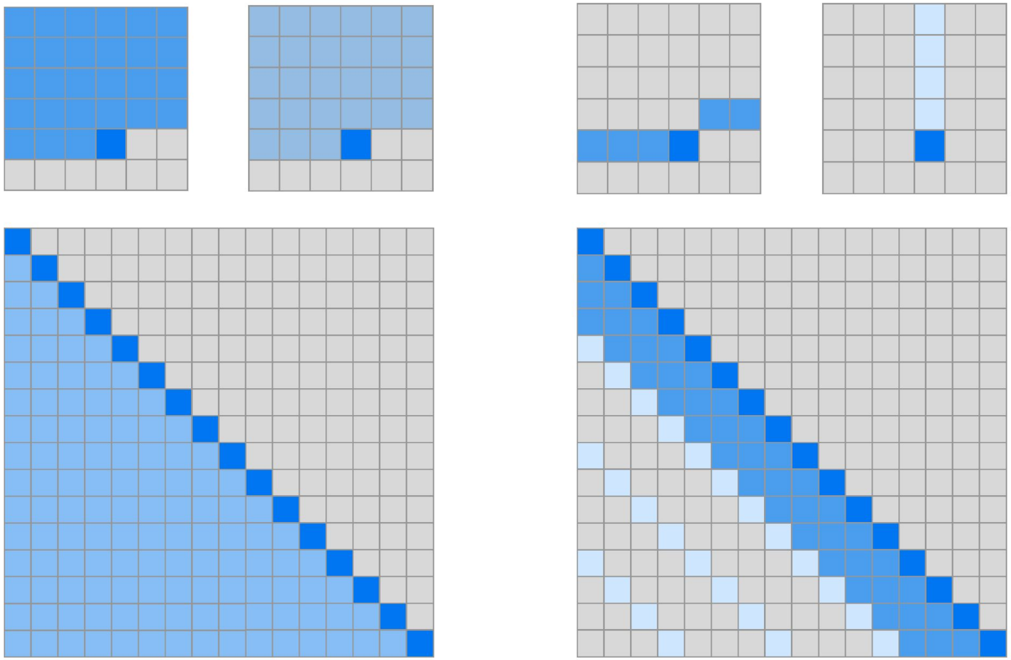
Liu et al. 2021

Axial attention



Ho et al. 2019, Wang et al. 2020

Other locality patterns



(a) Transformer (b) Sparse Transformer (strided)

Child et al. 2019

Changing attention form

$$\texttt{softmax}\left(QK^T\right)V$$

Changing attention form

$$\texttt{softmax}\left(QK^T\right)V$$

Drop the softmax

$$\left(QK^T\right)V$$

Changing attention form

$$\texttt{softmax}\left(QK^T\right)V$$

Drop the softmax

$$\left(QK^T\right)V$$

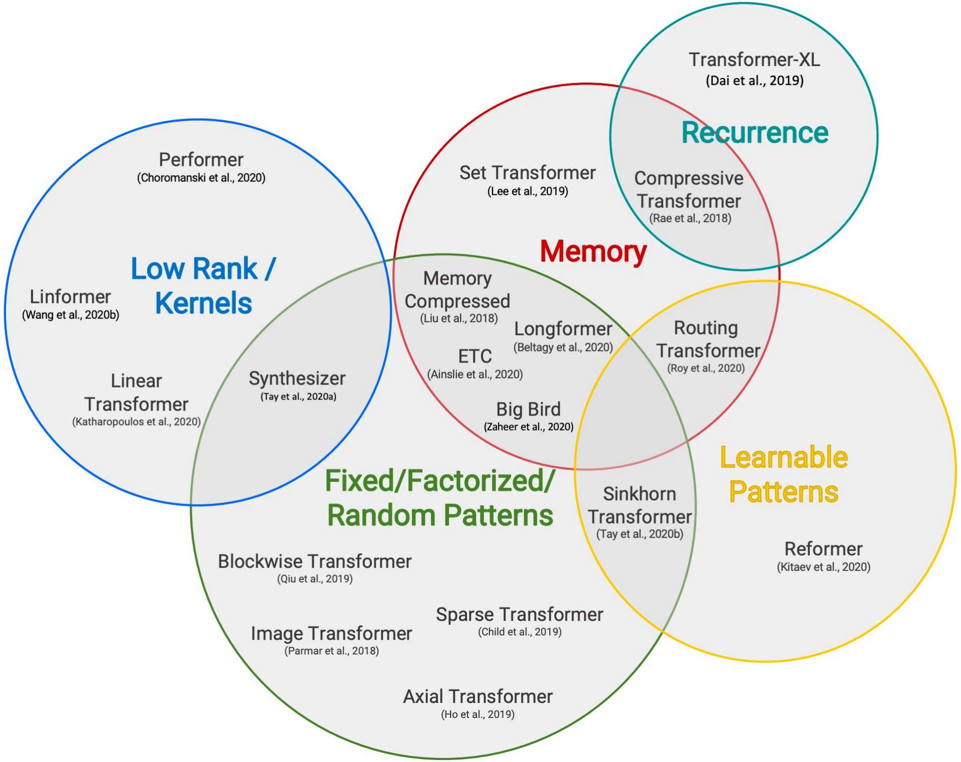
[n, c] [c, n] [n, c]

Change order of computation

$$Q\left(K^TV\right)$$

Useful when length is much larger than channels

Lot of ideas to try out!

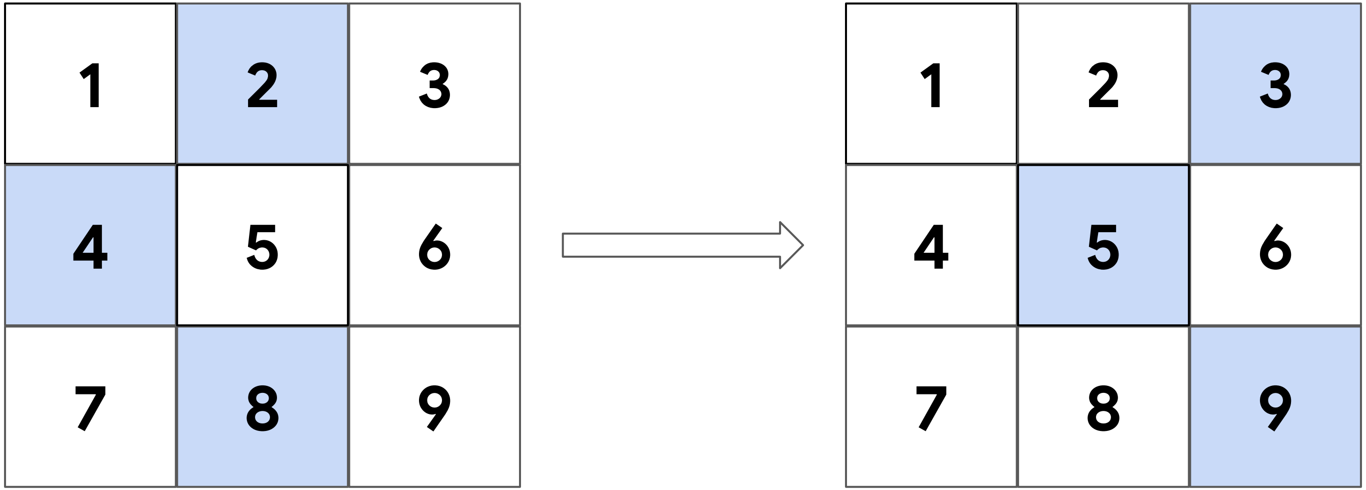


[Tay et al. 2020. Efficient Transformers: A Survey.]

Positional information in attention affects properties

- Attention needs positional information

- Absolute coordinate system does not encode translational equivariance



Relative geometry encodes translational equivariance

-1, -1	-1, 0	-1, 1
0, -1	0, 0	0, 1
1, -1	1, 0	1, 1

Bello et al. 2019
Ramachandran et al. 2019

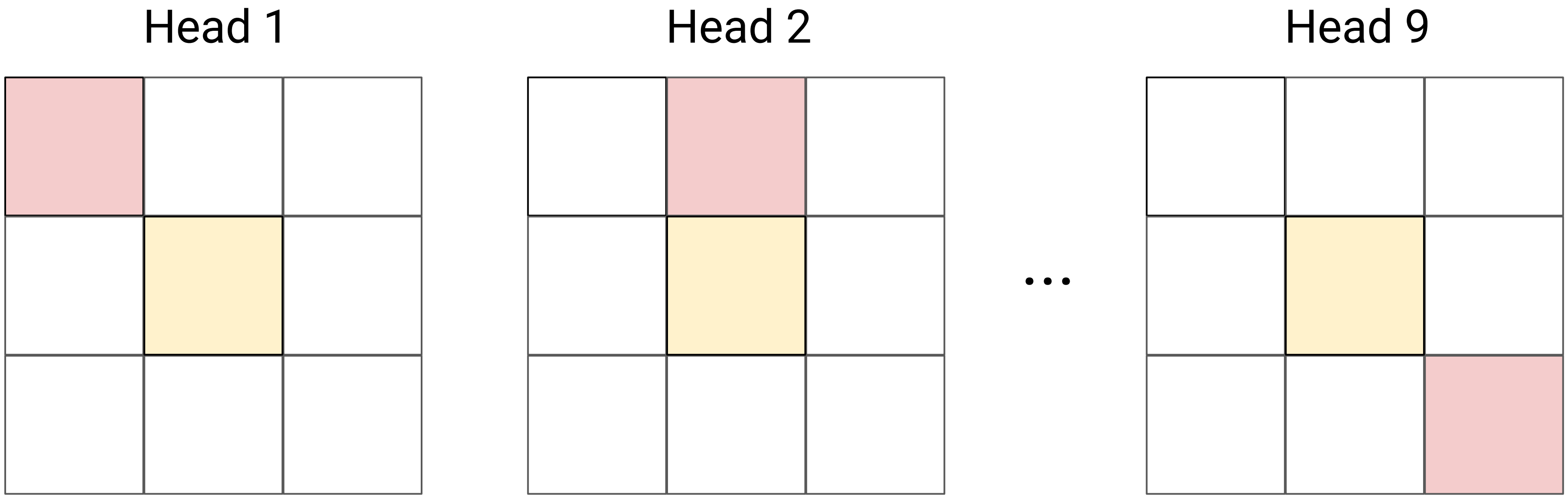
Relative geometry encodes translational equivariance

-1, -1	-1, 0	-1, 1
0, -1	0, 0	0, 1
1, -1	1, 0	1, 1

$$y_{ij} = \sum_{a,b \in \mathcal{N}(i,j)} \texttt{softmax}_{ab} \left(q_{ij}^{\top} k_{ab} + \boxed{q_{ij}^{\top} r_{a-i,b-j}} \right) v_{ab}$$

Bello et al. 2019
Ramachandran et al. 2019

Attention can act like convolutions through relative geometry



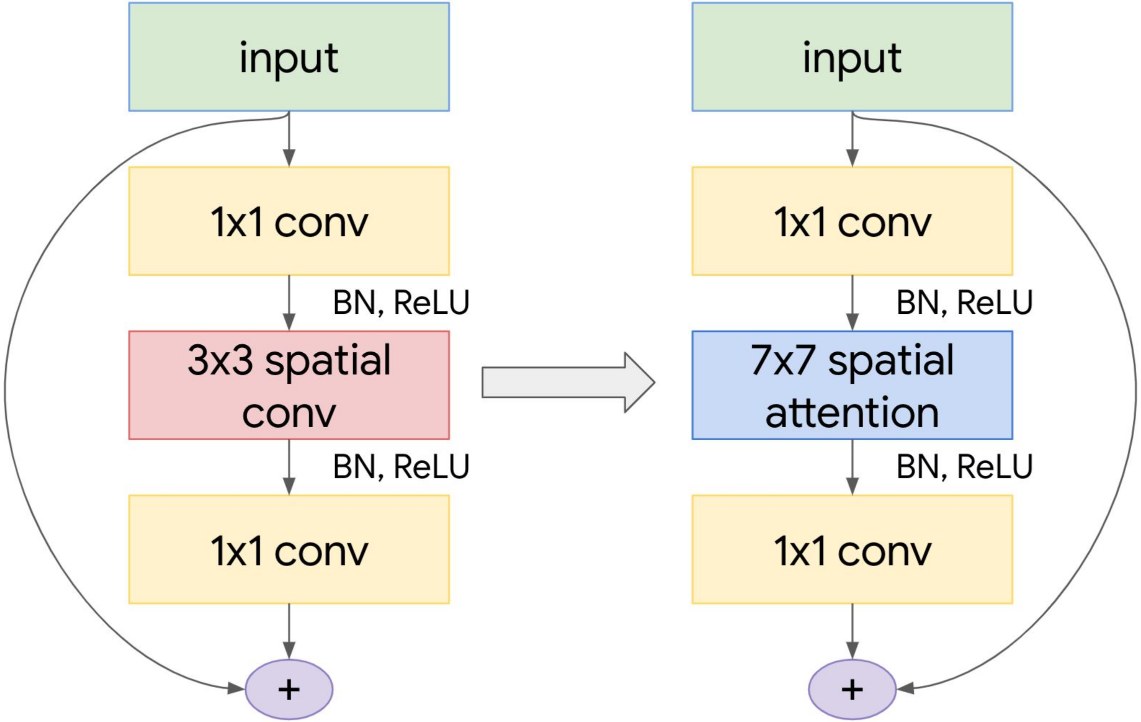
Cordonnier et al. (2020)

Relative geometry improves performance

Positional Encoding Type	FLOPS (B)	Params (M)	Top-1 Acc. (%)
none	6.9	18.0	77.6
absolute	6.9	18.0	78.2
relative	7.0	18.0	80.2

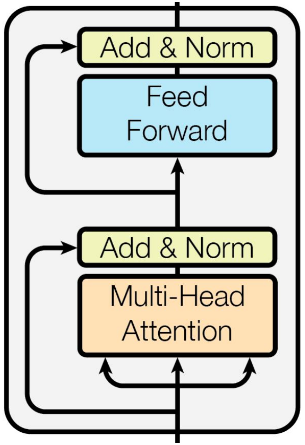
Ramachandran et al. 2019

What about starting from ResNets, not Transformers?

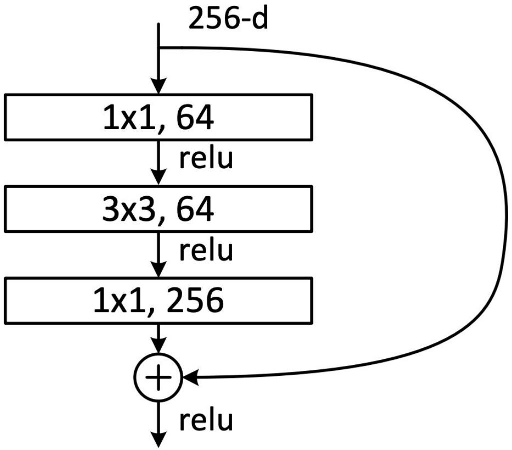


Ramachandran et al. 2019

Block type

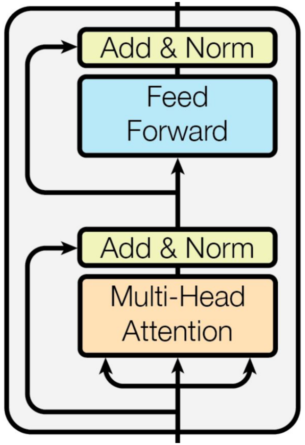


Transformer

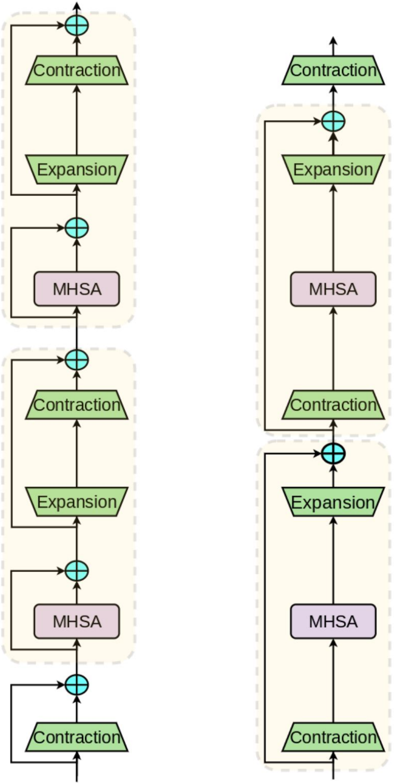


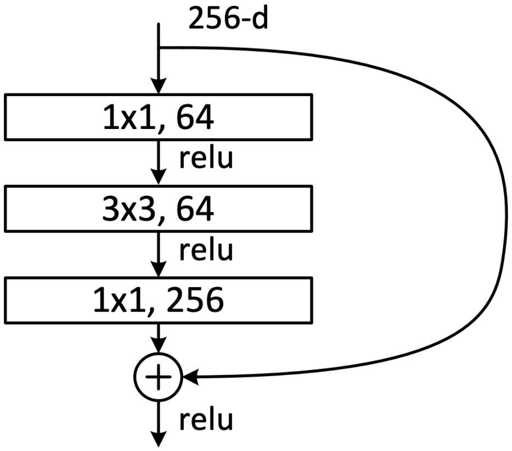
ResNet

Block type



Transformer





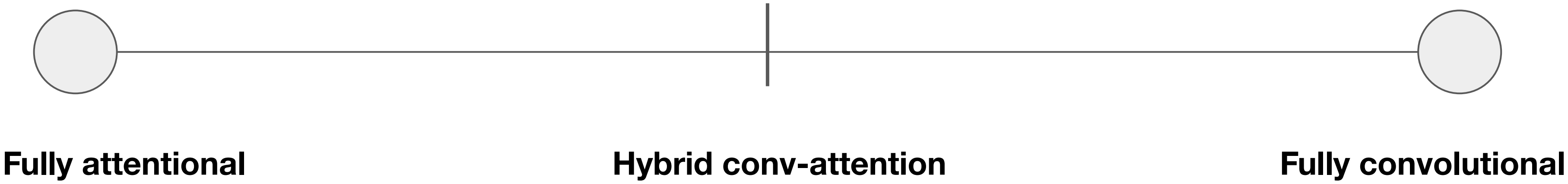
ResNet

Srinivas et al. 2020

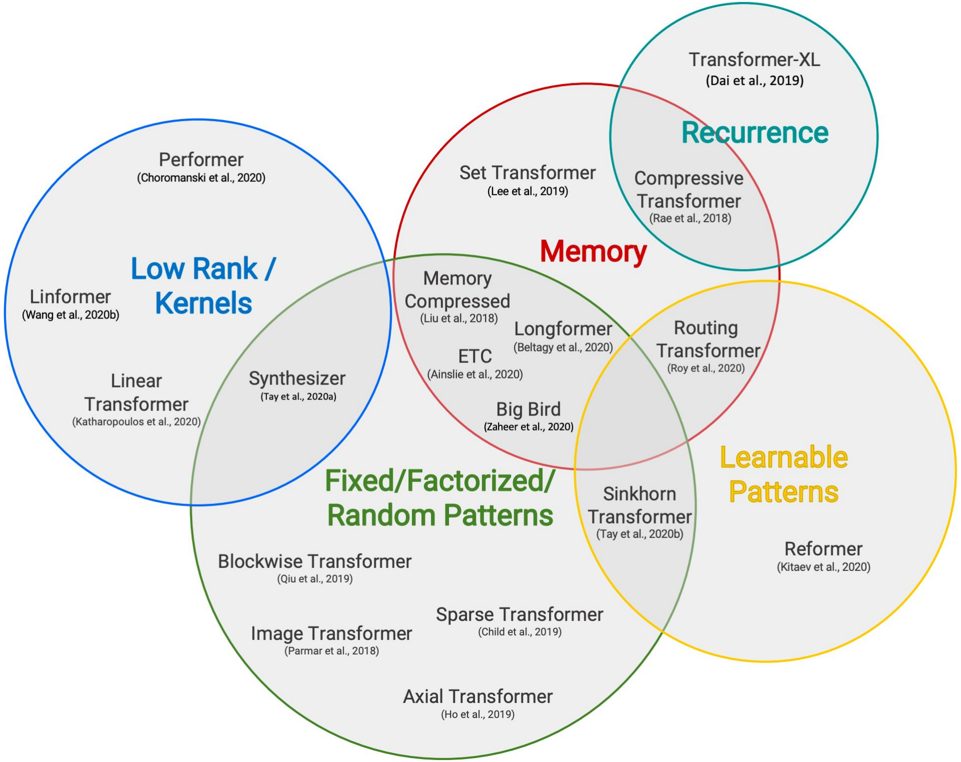
Categorizing the types of attention backbones

- Various ways to categorize a particular attention backbone
- Not comprehensive, but a good starting point

Axes: operational purity



Axes: attention form

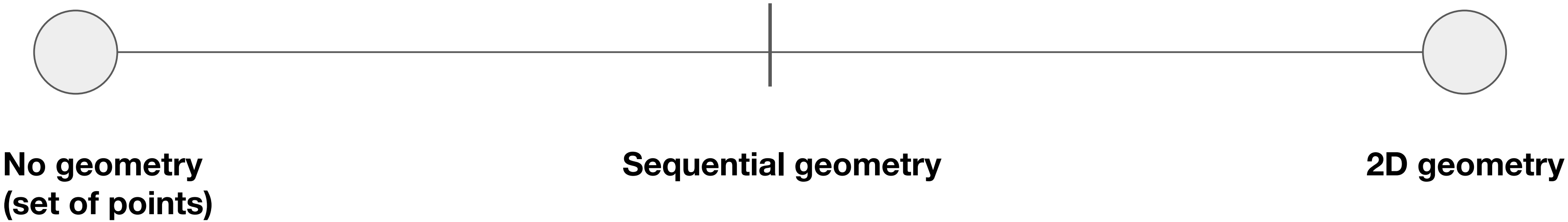


[Tay et al. 2020. Efficient Transformers: A Survey.]

Axes: number of scales

Single-scale **Multi-scale**

Axes: geometry



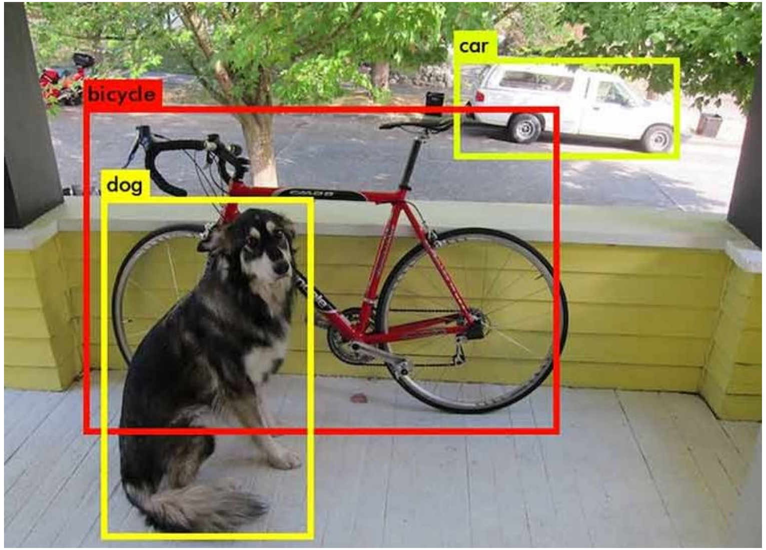
Axes: block type

Recap

- Many ways of adding attention to vision backbones
- One of the biggest challenges is the quadratic complexity of attention
- Numerous strategies developed to tackle this challenge

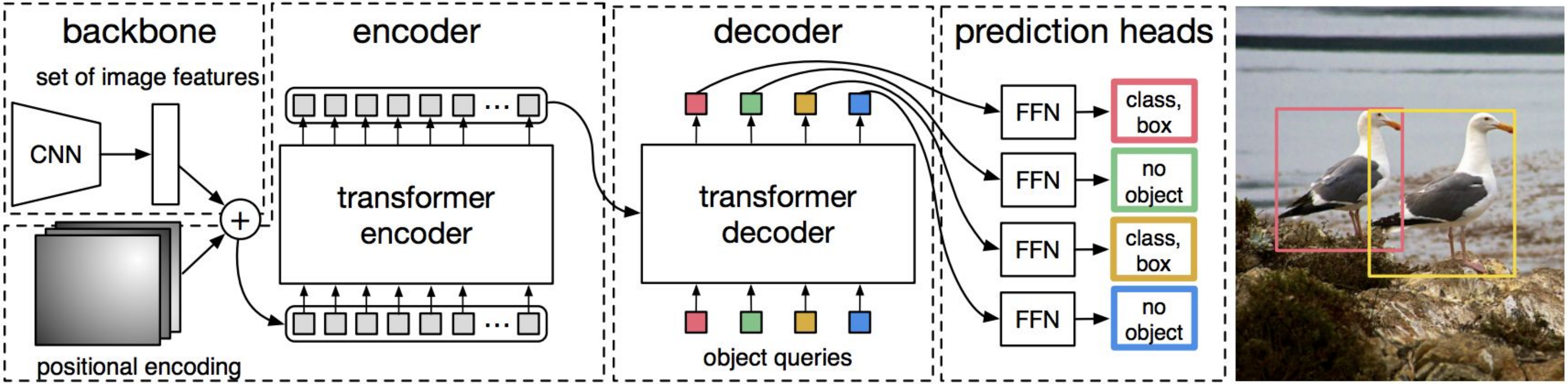
Survey of self-attention applications in Computer Vision

Transformers for Object Detection



Carion et al 2020

DETR: End-to-End Object Detection with Transformers



Carion et al 2020

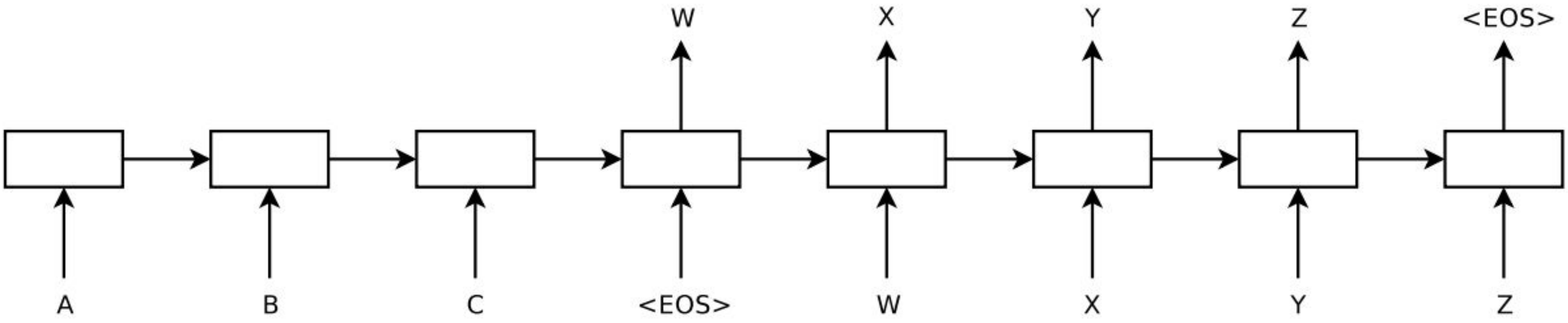
seq2seq

Sequence to Sequence Learning with Neural Networks

Ilya Sutskever
Google

Oriol Vinyals
Google

Quoc V. Le
Google



[Sutskever et al 2014]

seq2seq

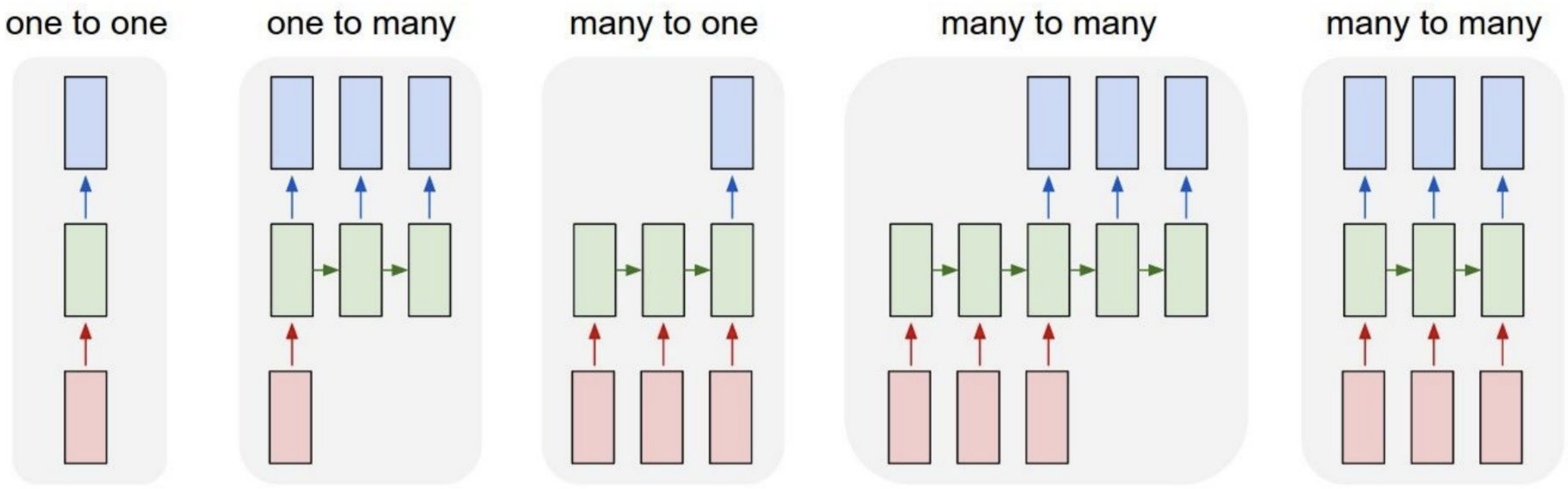
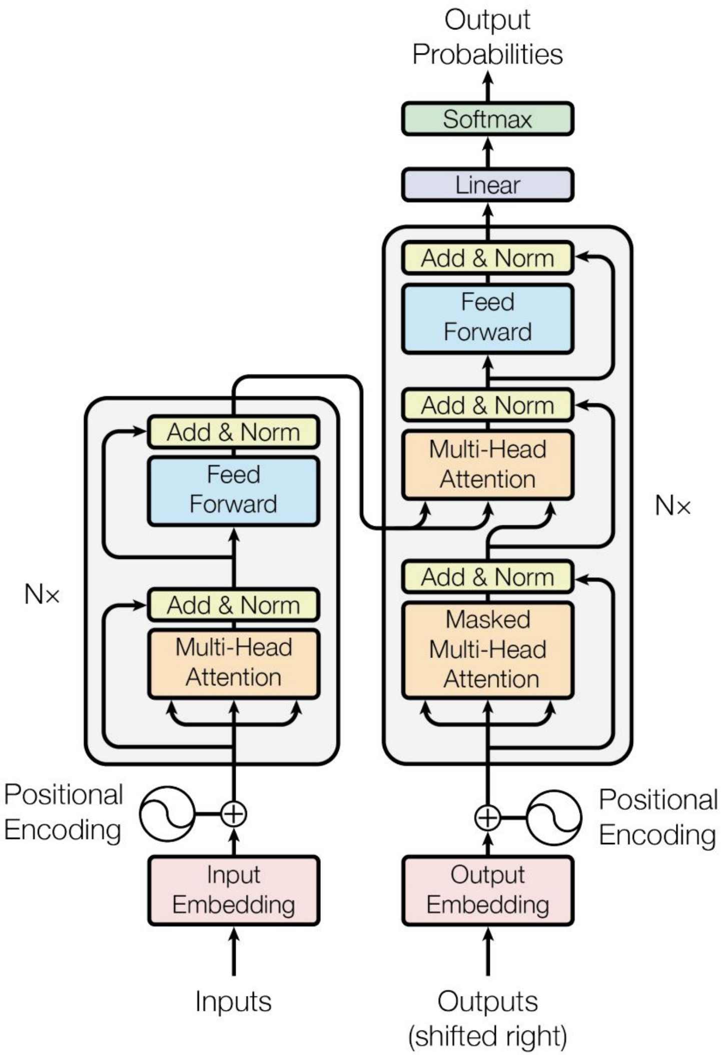


Image Captioning Sentiment Analysis Machine Translation Video Captioning

cs231n, Andrej Karpathy

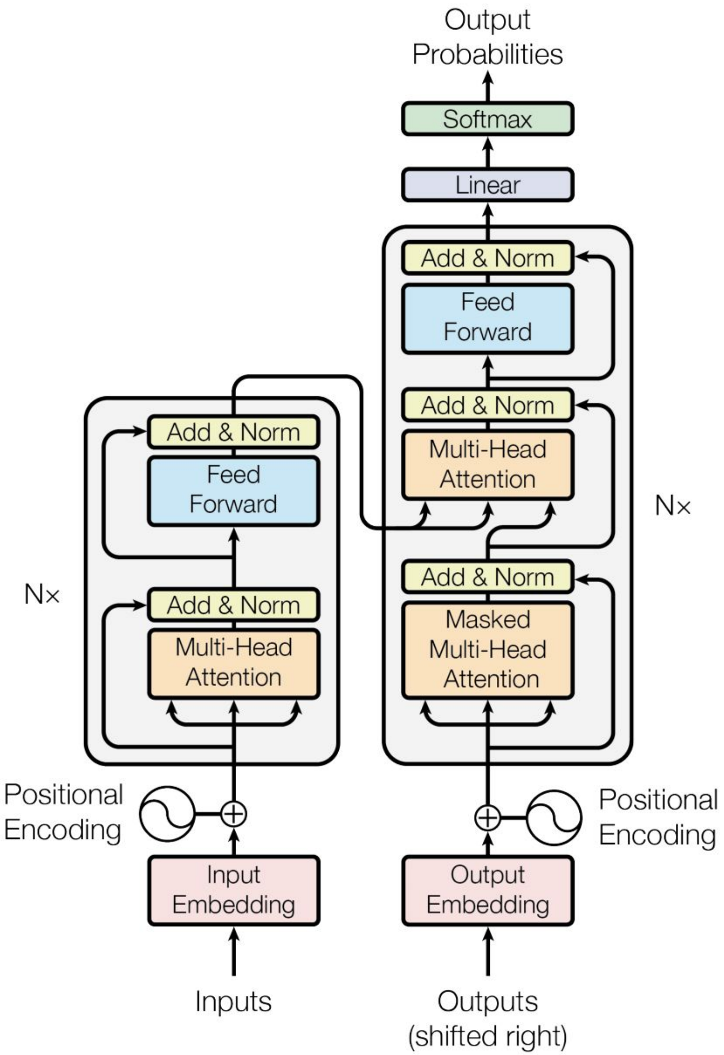
Transformer



Vaswani et al 2017

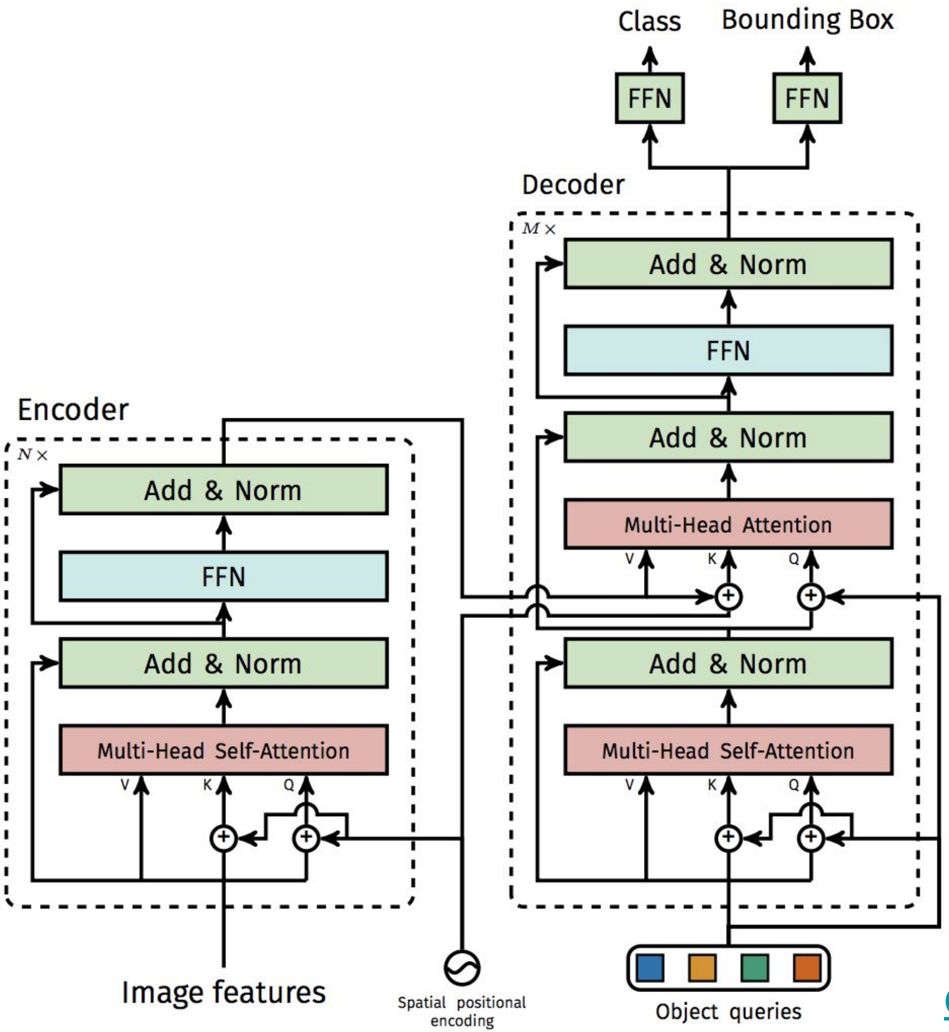
Transformer

1. Machine Translation
2. Language Modeling
3. Image Generation
4. Image Captioning
5. Multimodal
6. …..



[Vaswani et al 2017]

DETR



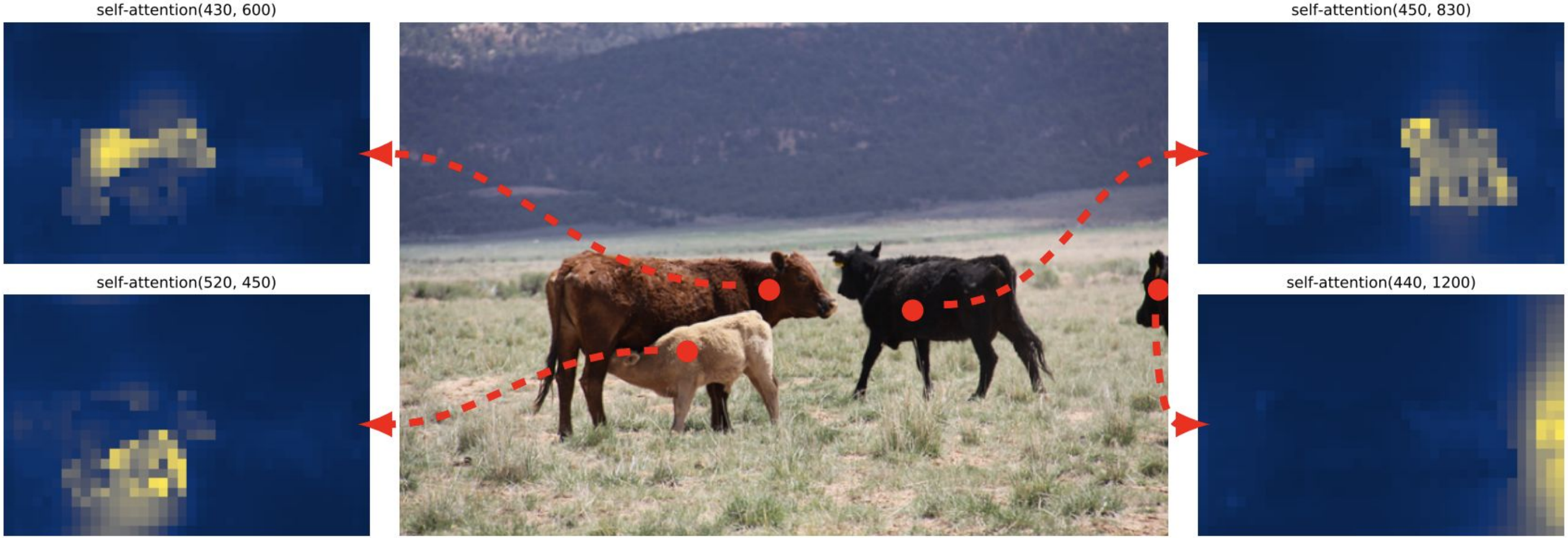
[Carion et al 2020]

DETR

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**

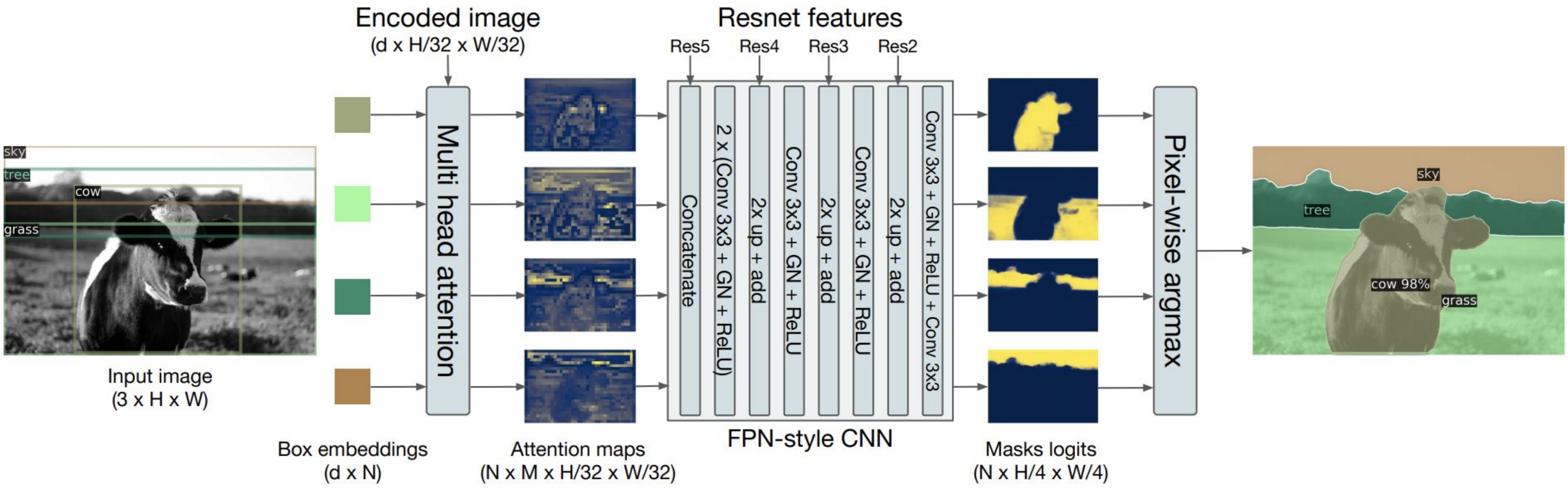
Carion et al 2020

DETR



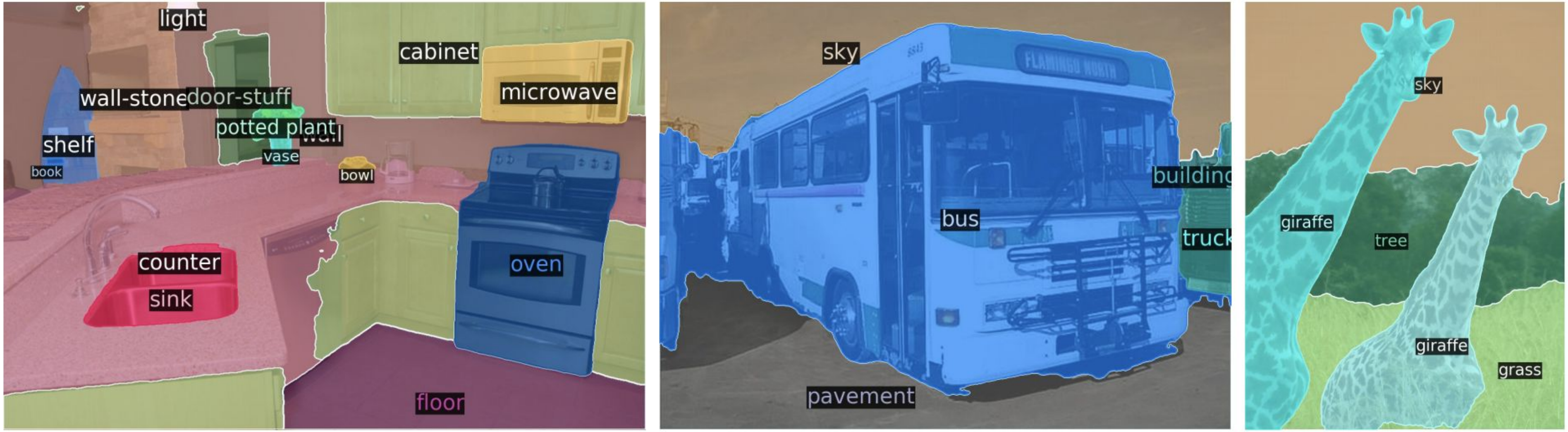
Carion et al 2020

DETR can be modified to perform panoptic segmentation



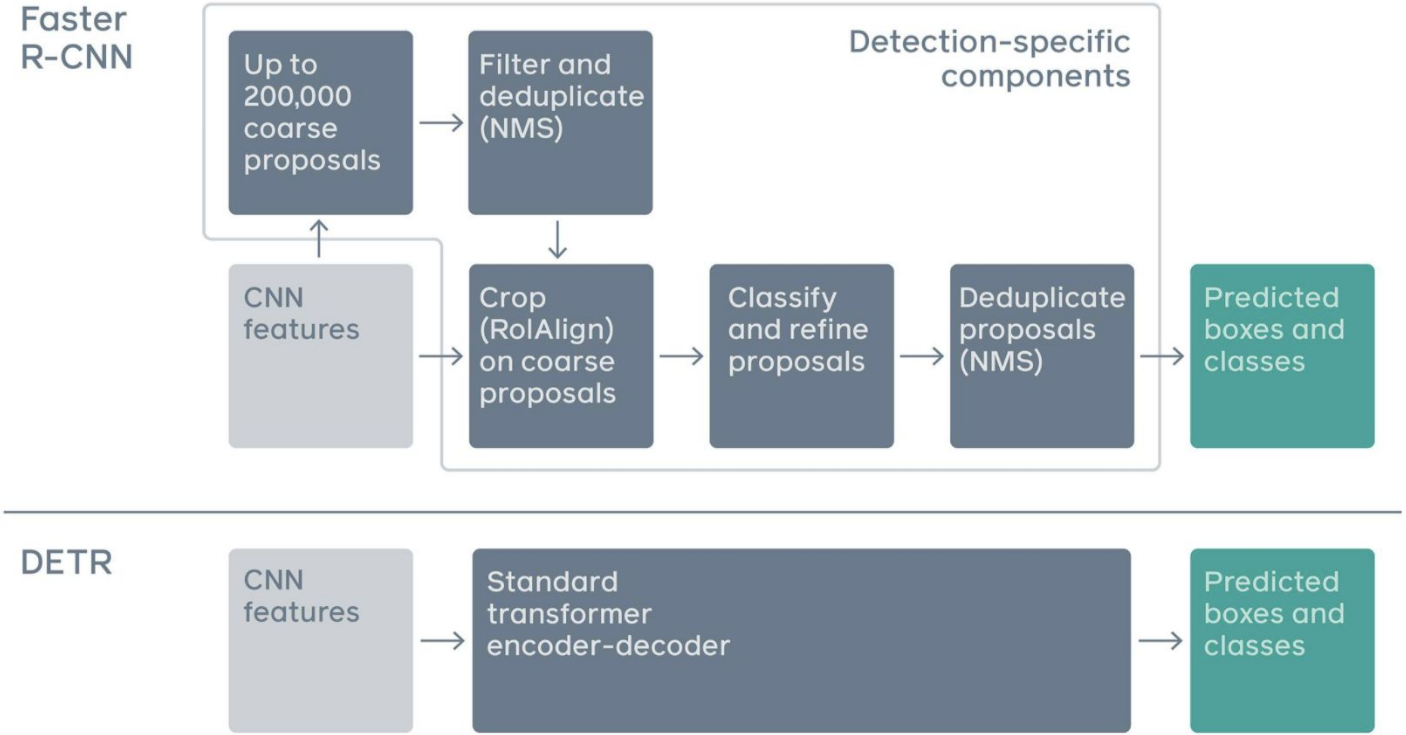
Carion et al 2020

DETR can be modified to perform panoptic segmentation



Carion et al 2020

DETR Inference Code is vastly simpler



[Carion et al 2020]

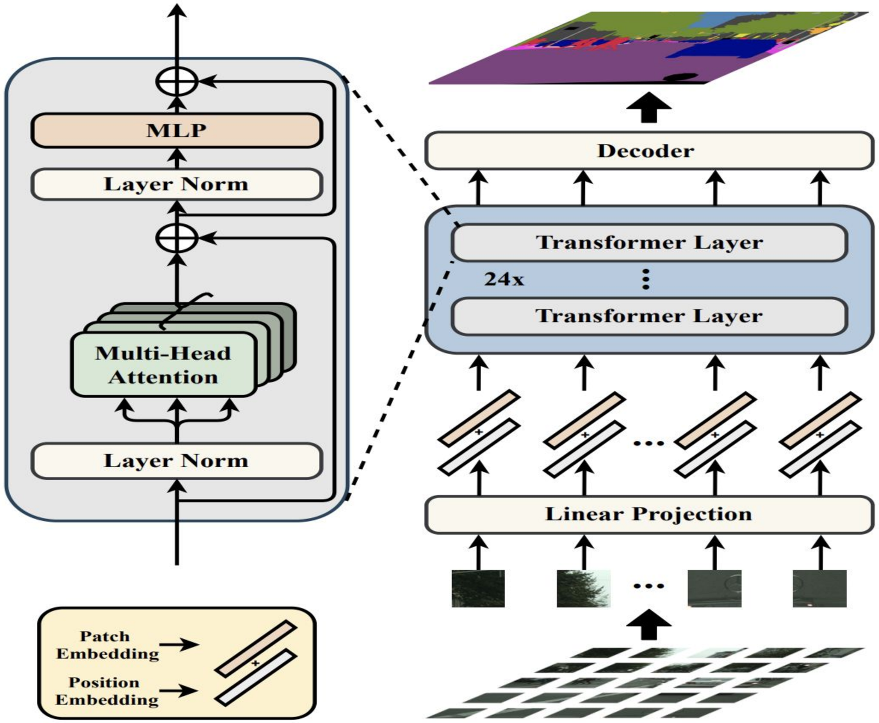
DETR Inference Code is vastly simpler

```
import torch
from torch import nn
from torchvision.models import resnet50

class DETR(nn.Module):

    def __init__(self, num_classes, hidden_dim, nheads,
                 num_encoder_layers, num_decoder_layers):
        super().__init__()
        # We take only convolutional layers from ResNet-50 model
        self.backbone = nn.Sequential(*list(resnet50(pretrained=True).children())[:-2])
        self.conv = nn.Conv2d(2048, hidden_dim, 1)
        self.transformer = nn.Transformer(hidden_dim, nheads,
                                          num_encoder_layers, num_decoder_layers)
        self.linear_class = nn.Linear(hidden_dim, num_classes + 1)
        self.linear_bbox = nn.Linear(hidden_dim, 4)
        self.query_pos = nn.Parameter(torch.rand(100, hidden_dim))
        self.row_embed = nn.Parameter(torch.rand(50, hidden_dim // 2))
        self.col_embed = nn.Parameter(torch.rand(50, hidden_dim // 2))

    def forward(self, inputs):
        x = self.backbone(inputs)
        h = self.conv(x)
        H, W = h.shape[-2:]
        pos = torch.cat([
            self.col_embed[:W].unsqueeze(0).repeat(H, 1, 1),
            self.row_embed[:H].unsqueeze(1).repeat(1, W, 1),
        ], dim=-1).flatten(0, 1).unsqueeze(1)
        h = self.transformer(pos + h.flatten(2).permute(2, 0, 1),
                             self.query_pos.unsqueeze(1))
        return self.linear_class(h), self.linear_bbox(h).sigmoid()

detr = DETR(num_classes=91, hidden_dim=256, nheads=8, num_encoder_layers=6, num_decoder_layers=6)
detr.eval()
inputs = torch.randn(1, 3, 800, 1200)
logits, bboxes = detr(inputs)
```

Carion et al 2020

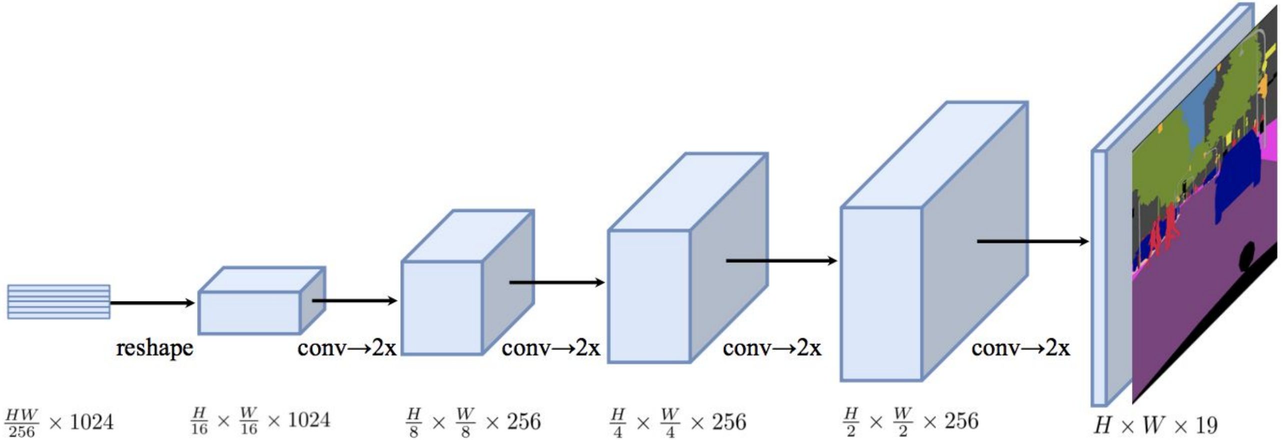
Transformers for Semantic Segmentation

Segmentation Transformer (SETR)

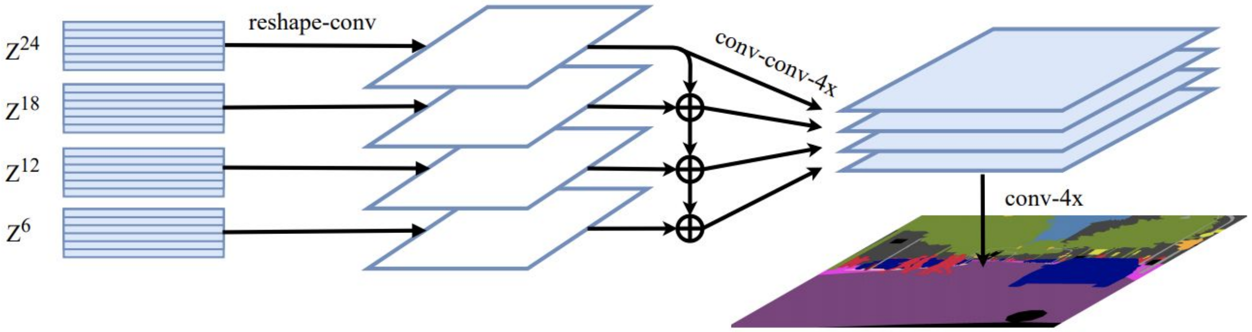


Zheng et al 2020

SETR



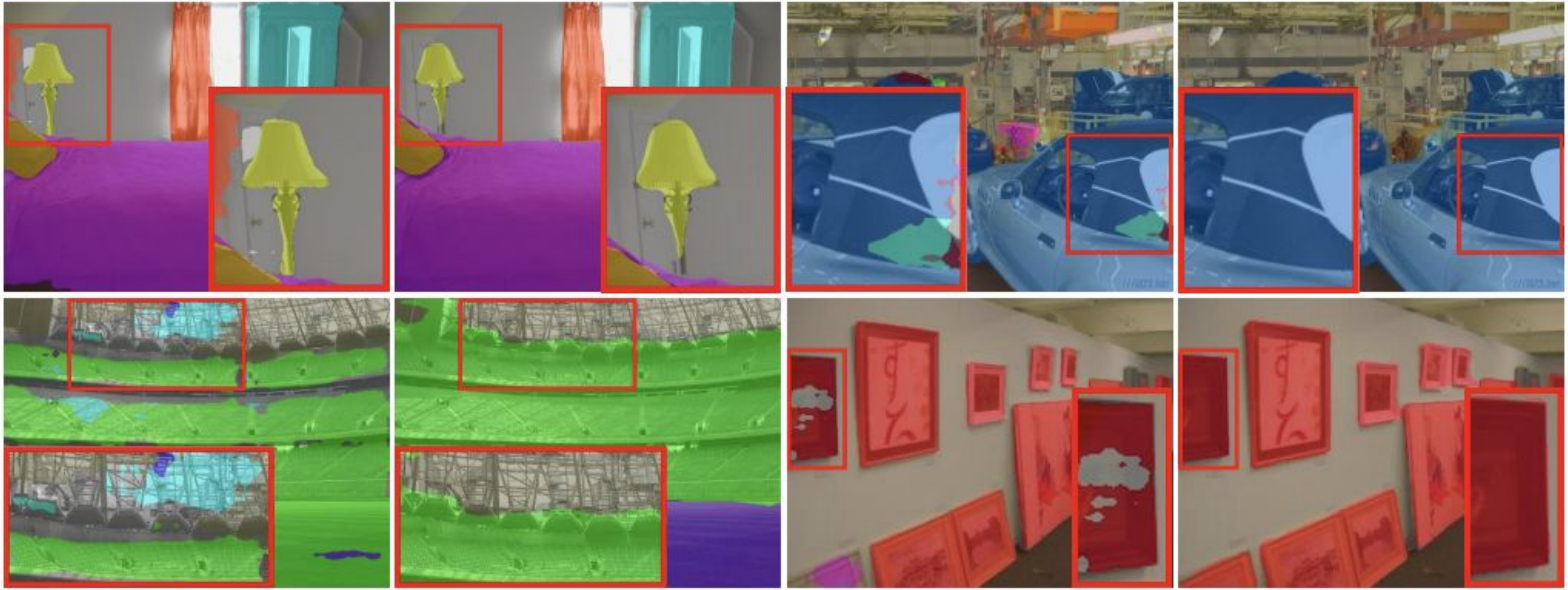
(b)



(c)

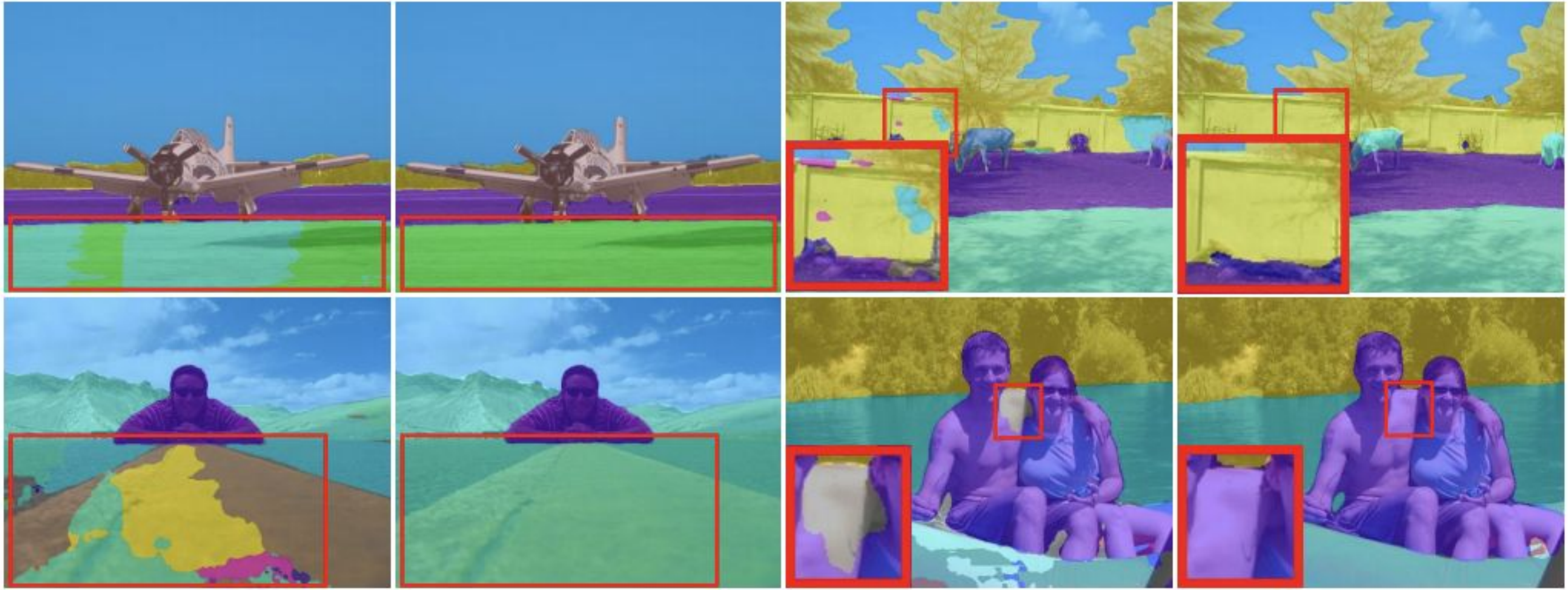
[Zheng et al 2020]

SETR



Zheng et al 2020

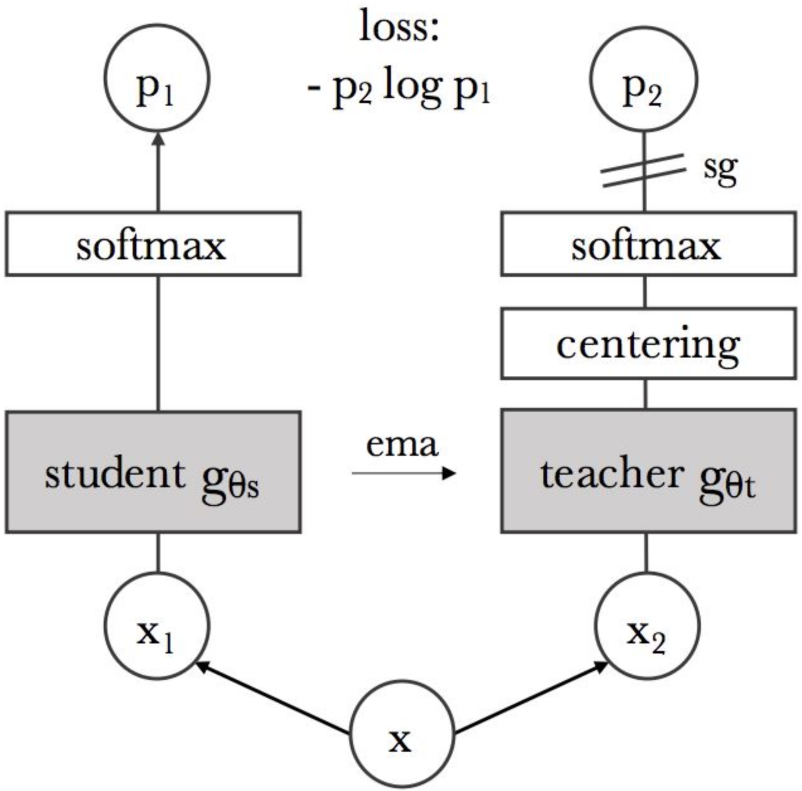
SETR



Zheng et al 2020

Transformers for Self-Supervised Learning

DINO

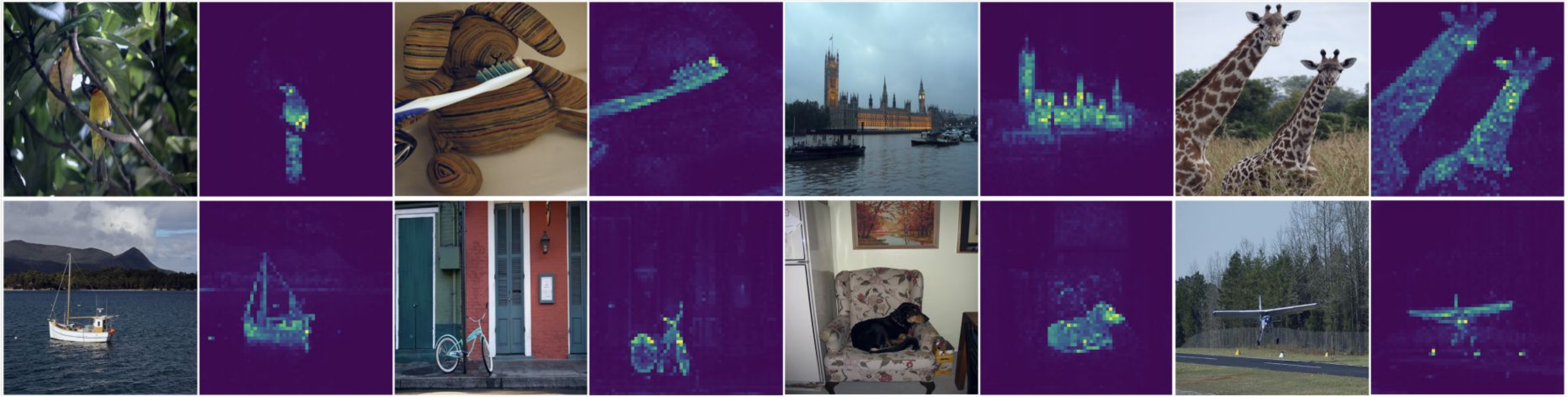


Caron et al 2020

DINO

Caron et al 2020

DINO



Caron et al 2020

DINO

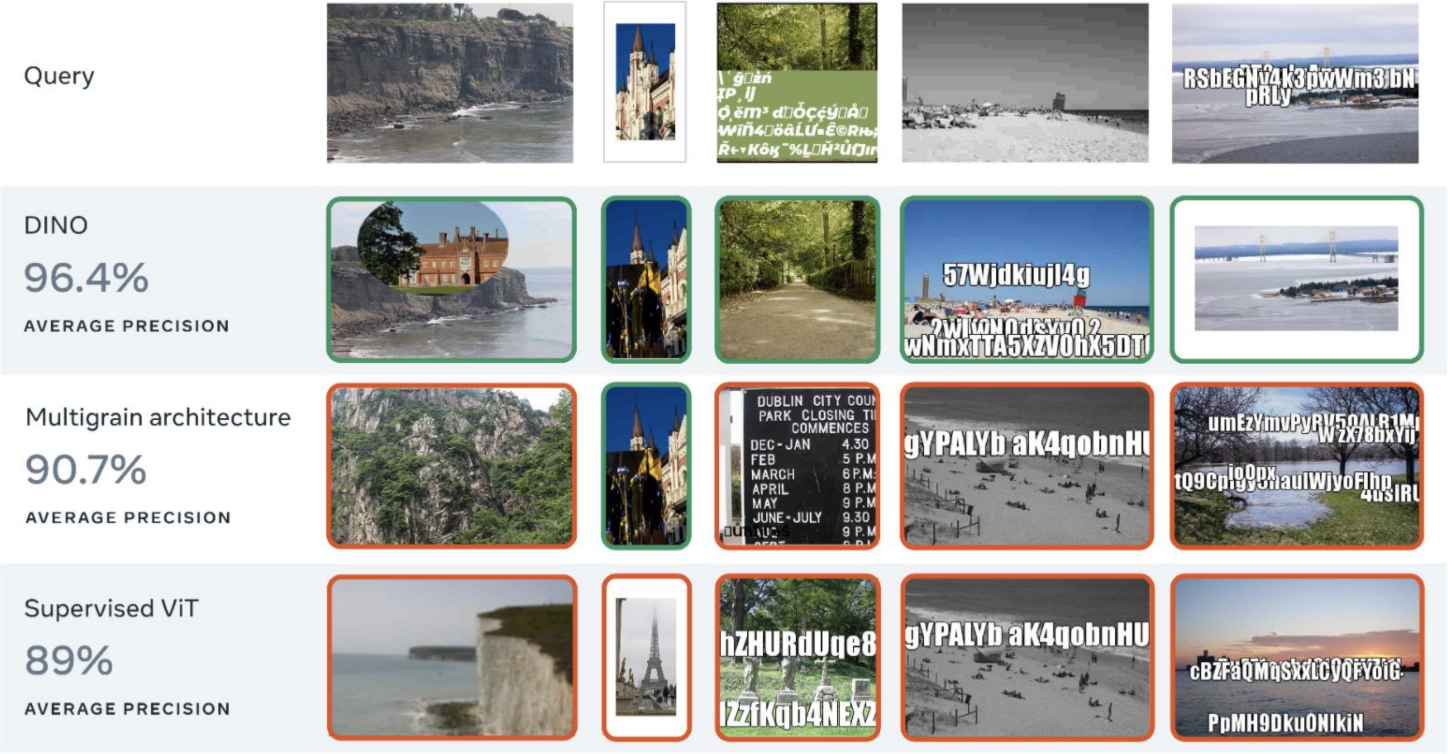
Caron et al 2020

DINO

Method	Arch.	Param.	im/s	Linear	k-NN
Supervised	RN50	23	1237	79.3	79.3
SCLR [12]	RN50	23	1237	69.1	60.7
MoCov2 [15]	RN50	23	1237	71.1	61.9
InfoMin [67]	RN50	23	1237	73.0	65.3
BarlowT [81]	RN50	23	1237	73.2	66.0
OBoW [27]	RN50	23	1237	73.8	61.9
BYOL [30]	RN50	23	1237	74.4	64.8
DCv2 [10]	RN50	23	1237	75.2	67.1
SwAV [10]	RN50	23	1237	**75.3**	65.7
DINO	RN50	23	1237	**75.3**	**67.5**
Supervised	ViT-S	21	1007	79.8	79.8
BYOL* [30]	ViT-S	21	1007	71.4	66.6
MoCov2* [15]	ViT-S	21	1007	72.7	64.4
SwAV* [10]	ViT-S	21	1007	73.5	66.3
DINO	ViT-S	21	1007	**77.0**	**74.5**
Comparison across architectures					
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 2020

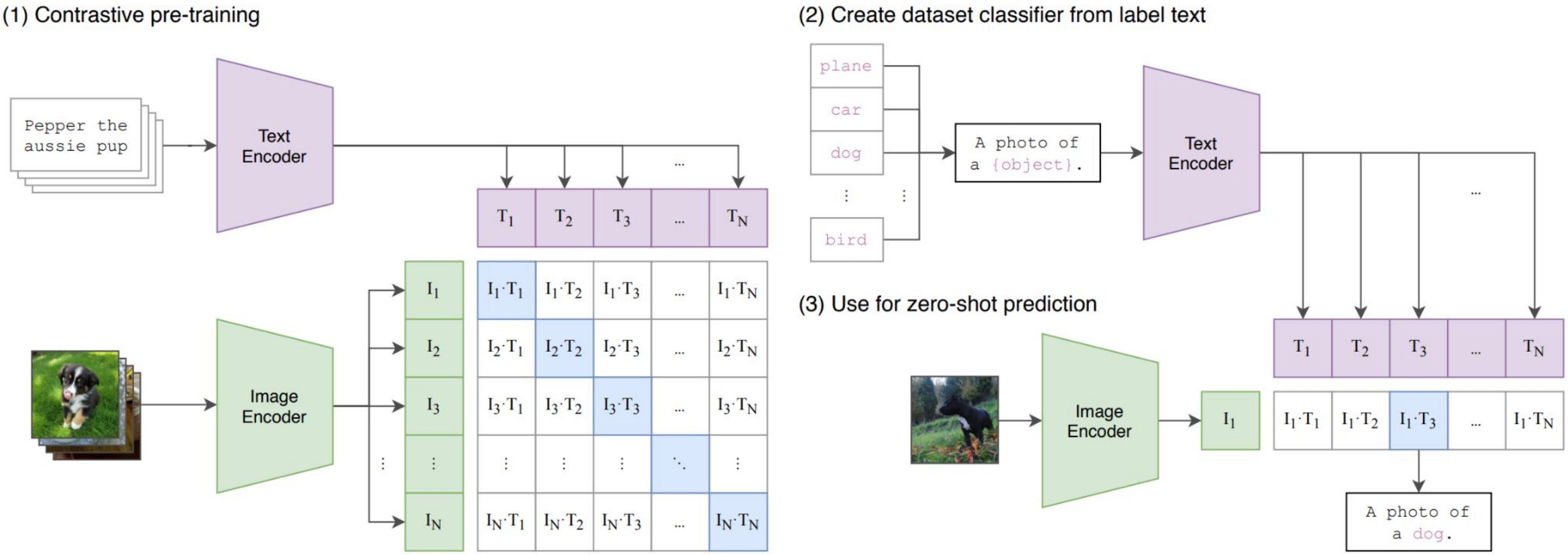
DINO



[Caron et al 2020]

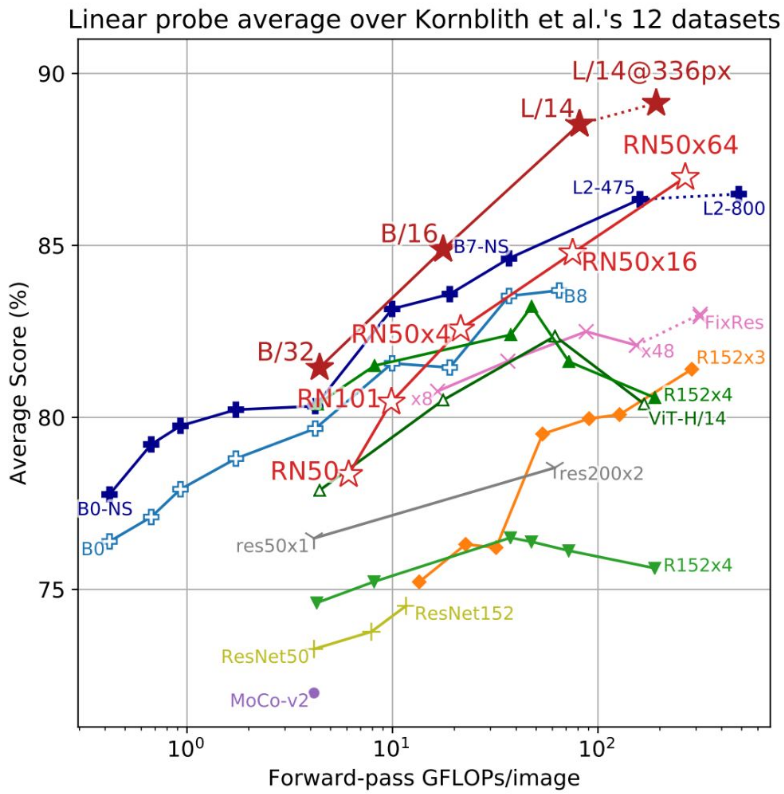
Transformers for Multi-Modal Learning

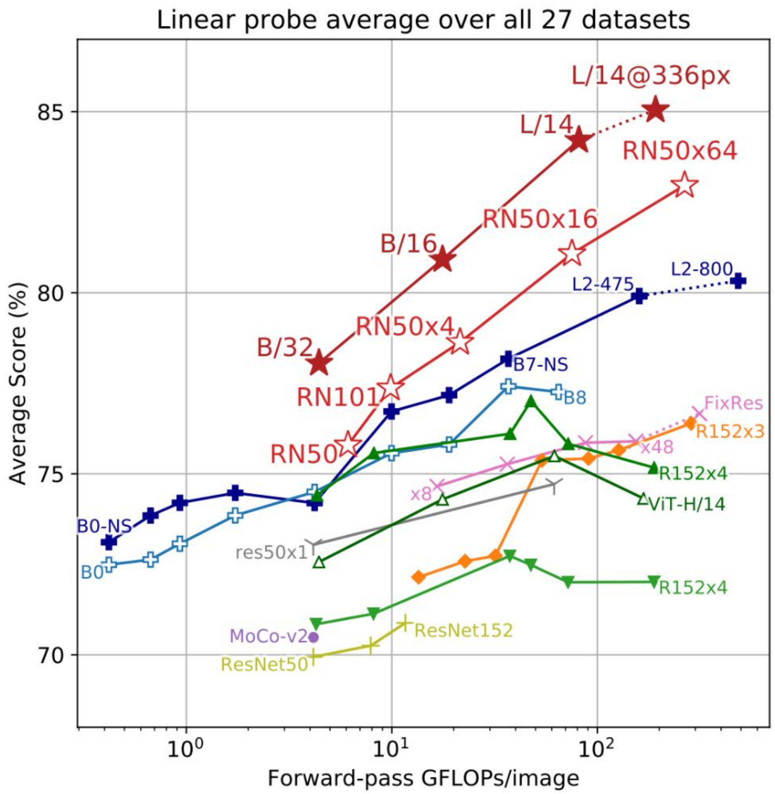
CLIP



Radford et al 2021

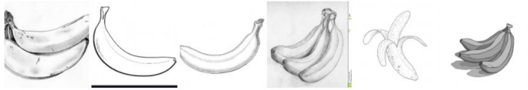
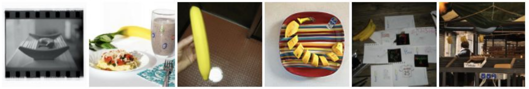
CLIP





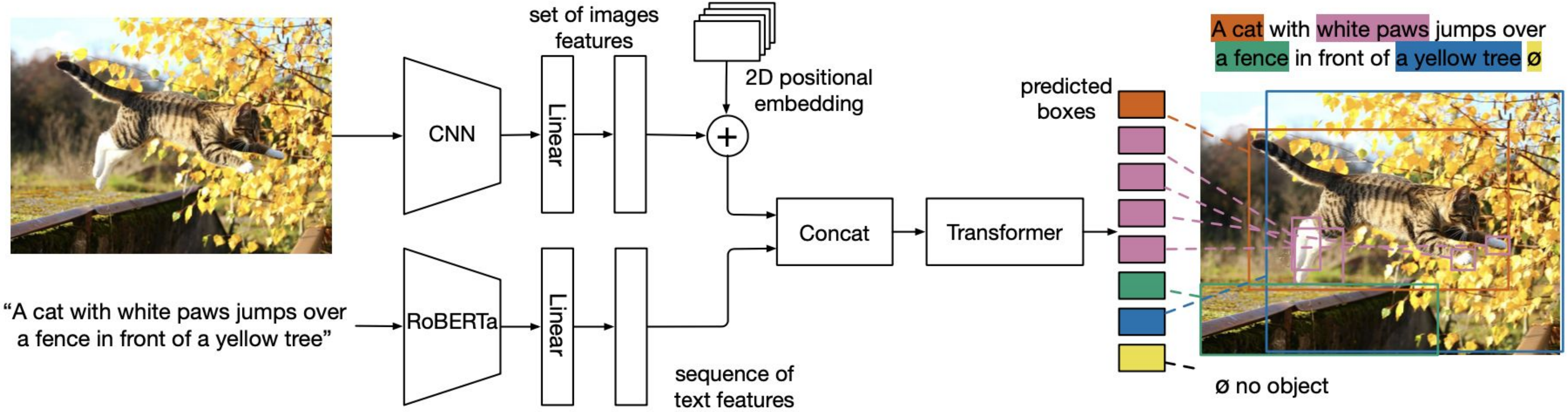
Radford et al 2021

CLIP

DATASET	IMAGENET RESNET101	CLIP VIT-L
 ImageNet	76.2%	76.2%
 ImageNet V2	64.3%	70.1%
 ImageNet Rendition	37.7%	88.9%
 ObjectNet	32.6%	72.3%
 ImageNet Sketch	25.2%	60.2%
 ImageNet Adversarial	2.7%	77.1%

[Radford et al 2021]

Multimodal DETR (M-DETR)

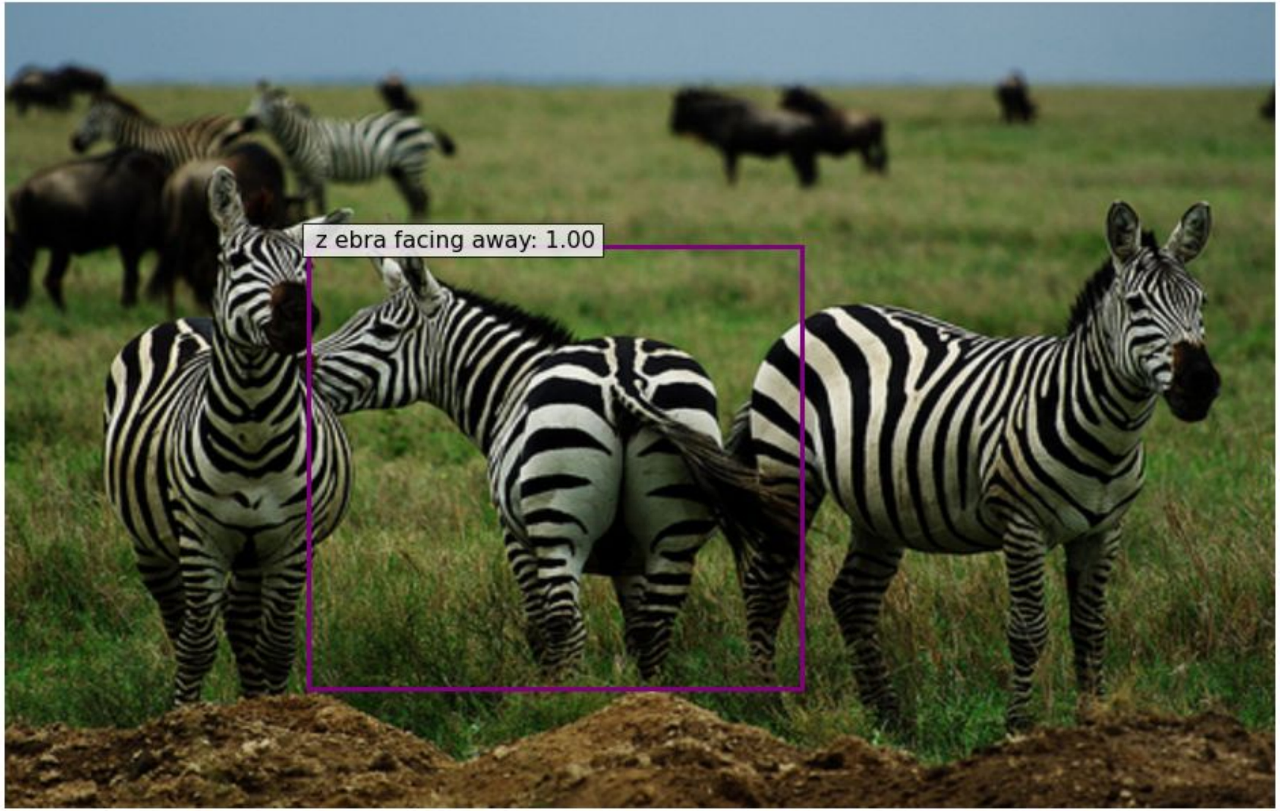


[Kamath et al 2021]

M-DETR

Kamath et al 2021

M-DETR

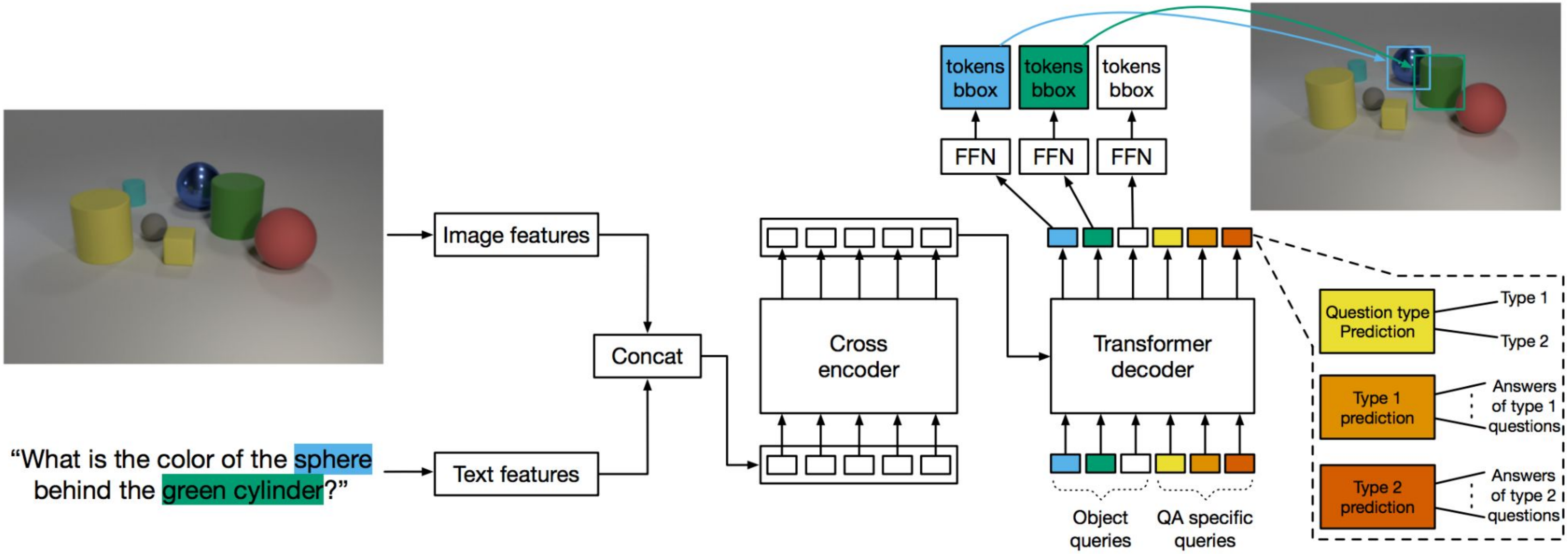


Kamath et al 2021

M-DETR

Kamath et al 2021

M-DETR



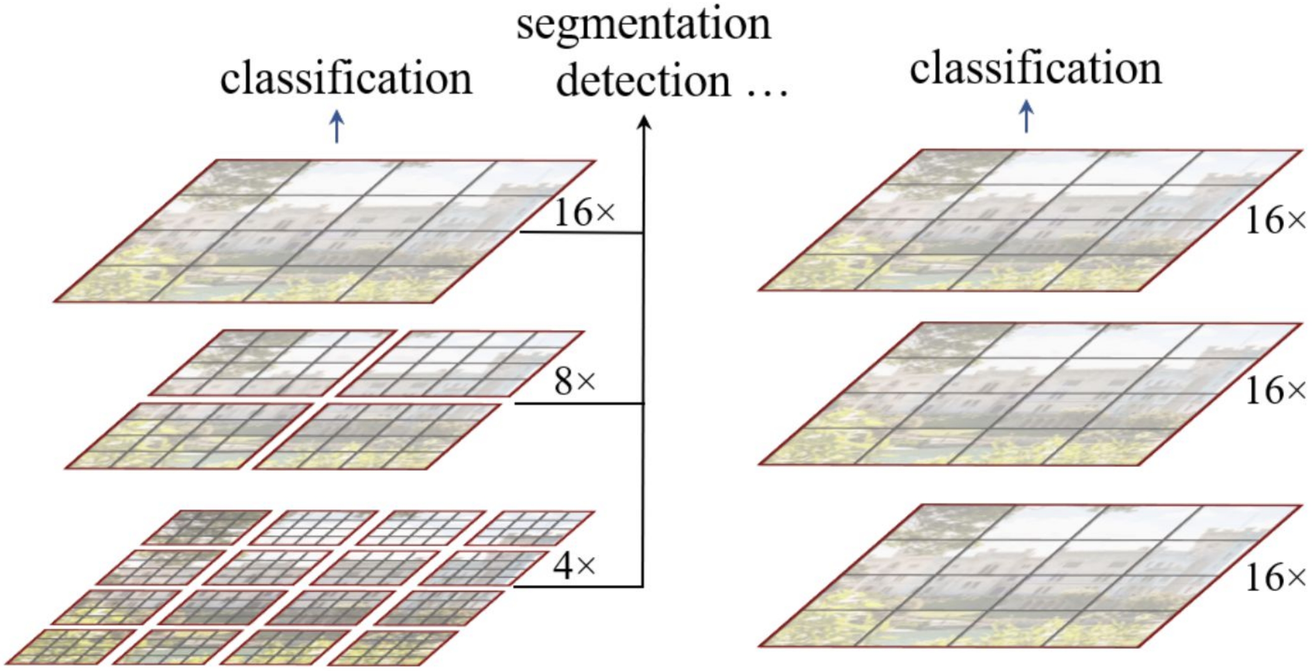
Kamath et al 2021

M-DETR

Kamath et al 2021

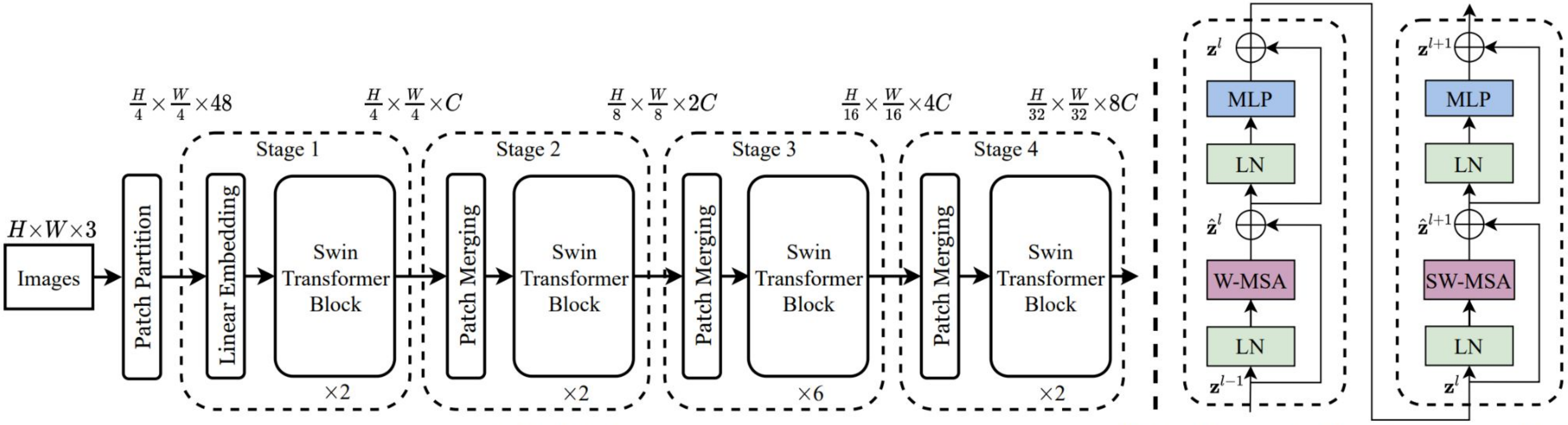
Multi-Scale Features in Transformers

Swin Transformer



[Liu et al 2021](#)

Swin Transformer



(a) Architecture
(b) Two Successive Swin Transformer Blocks

Liu et al 2021

Swin Transformer

method	image size	#param.	FLOPs	throughput (image / s)	ImageNet top-1 acc.
(a) Regular ImageNet-1K trained models					
RegNetY-4G [47]	224^2	21M	4.0G	1156.7	80.0
RegNetY-8G [47]	224^2	39M	8.0G	591.6	81.7
RegNetY-16G [47]	224^2	84M	16.0G	334.7	82.9
EffNet-B3 [57]	300^2	12M	1.8G	732.1	81.6
EffNet-B4 [57]	380^2	19M	4.2G	349.4	82.9
EffNet-B5 [57]	456^2	30M	9.9G	169.1	83.6
EffNet-B6 [57]	528^2	43M	19.0G	96.9	84.0
EffNet-B7 [57]	600^2	66M	37.0G	55.1	84.3
ViT-B/16 [19]	384^2	86M	55.4G	85.9	77.9
ViT-L/16 [19]	384^2	307M	190.7G	27.3	76.5
DeiT-S [60]	224^2	22M	4.6G	940.4	79.8
DeiT-B [60]	224^2	86M	17.5G	292.3	81.8
DeiT-B [60]	384^2	86M	55.4G	85.9	83.1
Swin-T	224^2	29M	4.5G	755.2	81.3
Swin-S	224^2	50M	8.7G	436.9	83.0
Swin-B	224^2	88M	15.4G	278.1	83.3
Swin-B	384^2	88M	47.0G	84.7	84.2
(b) ImageNet-22K pre-trained models					
method	image size	#param.	FLOPs	throughput (image / s)	ImageNet top-1 acc.
R-101x3 [37]	384^2	388M	204.6G	-	84.4
R-152x4 [37]	480^2	937M	840.5G	-	85.4
ViT-B/16 [19]	384^2	86M	55.4G	85.9	84.0
ViT-L/16 [19]	384^2	307M	190.7G	27.3	85.2
Swin-B	224^2	88M	15.4G	278.1	85.2
Swin-B	384^2	88M	47.0G	84.7	86.0
Swin-L	384^2	197M	103.9G	42.1	86.4

Liu et al 2021

Swin Transformer

(a) Various frameworks

Method	Backbone	AP^{box}	AP^{box}_{50}	AP^{box}_{75}	#param.	FLOPs	FPS
Cascade	R-50	46.3	64.3	50.5	82M	739G	18.0
Mask R-CNN	Swin-T	**50.5**	**69.3**	**54.9**	86M	745G	15.3
ATSS	R-50	43.5	61.9	47.0	32M	205G	28.3
	Swin-T	**47.2**	**66.5**	**51.3**	36M	215G	22.3
RepPointsV2	R-50	46.5	64.6	50.3	42M	274G	13.6
	Swin-T	**50.0**	**68.5**	**54.2**	45M	283G	12.0
Sparse	R-50	44.5	63.4	48.2	106M	166G	21.0
R-CNN	Swin-T	**47.9**	**67.3**	**52.3**	110M	172G	18.4

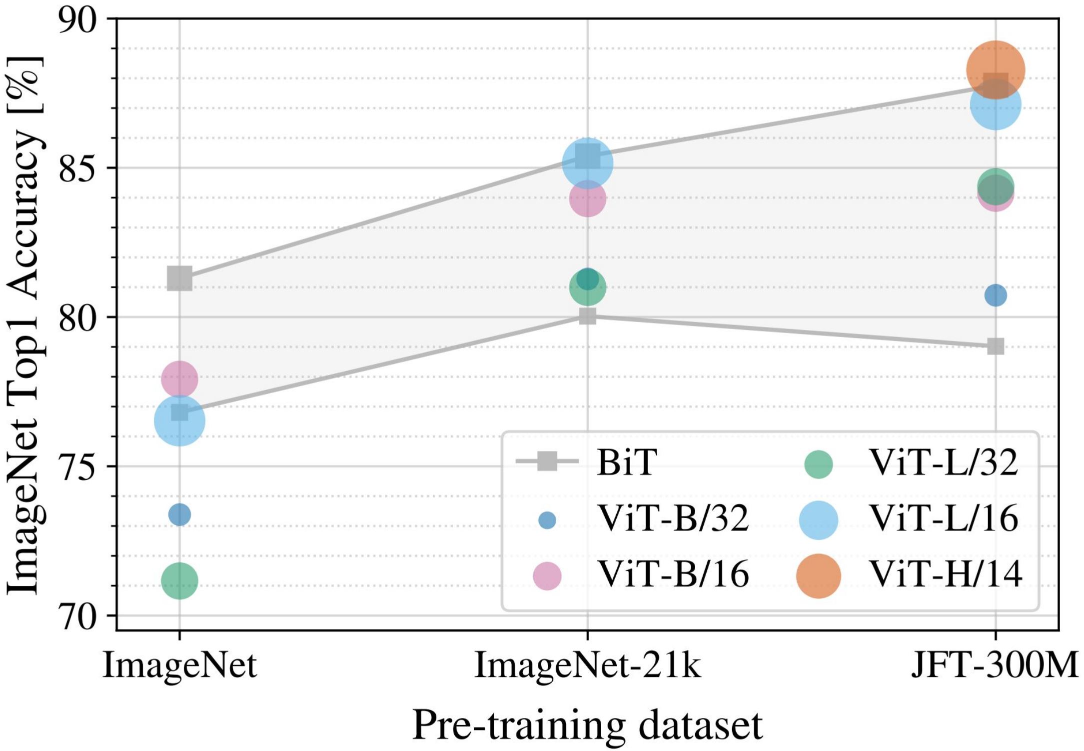
(b) Various backbones w. Cascade Mask R-CNN

	AP^{box}	AP^{box}_{50}	AP^{box}_{75}	AP^{mask}	AP^{mask}_{50}	AP^{mask}_{75}	param	FLOPs	FPS
DeiT-S†	48.0	67.2	51.7	41.4	64.2	44.3	80M	889G	10.4
R50	46.3	64.3	50.5	40.1	61.7	43.4	82M	739G	18.0
Swin-T	**50.5**	**69.3**	**54.9**	**43.7**	**66.6**	**47.1**	86M	745G	15.3
X101-32	48.1	66.5	52.4	41.6	63.9	45.2	101M	819G	12.8
Swin-S	**51.8**	**70.4**	**56.3**	**44.7**	**67.9**	**48.5**	107M	838G	12.0
X101-64	48.3	66.4	52.3	41.7	64.0	45.1	140M	972G	10.4
Swin-B	**51.9**	**70.9**	**56.5**	**45.0**	**68.4**	**48.7**	145M	982G	11.6

Liu et al 2021

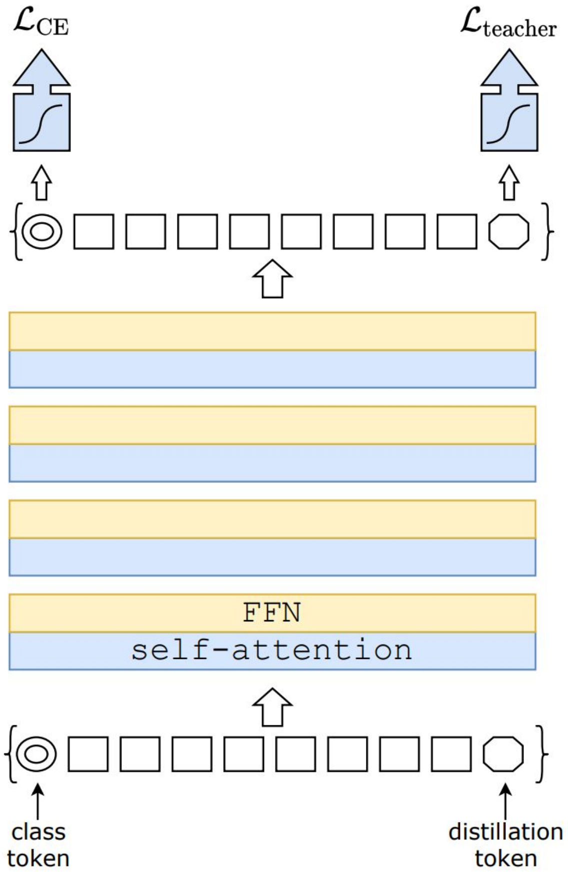
Transformers: Data and Model Regularization

Transformers scale well with data



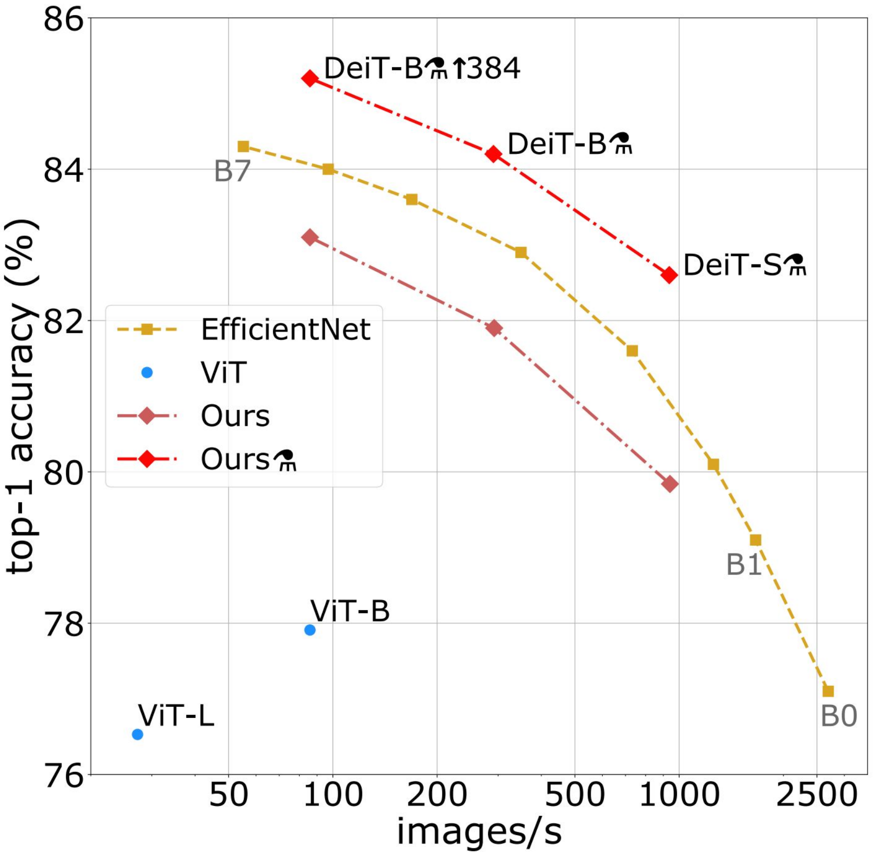
Dosovitskiy et al. 2020

DeiT (Data-Efficient Image Transformer)



Touvron et al 2020

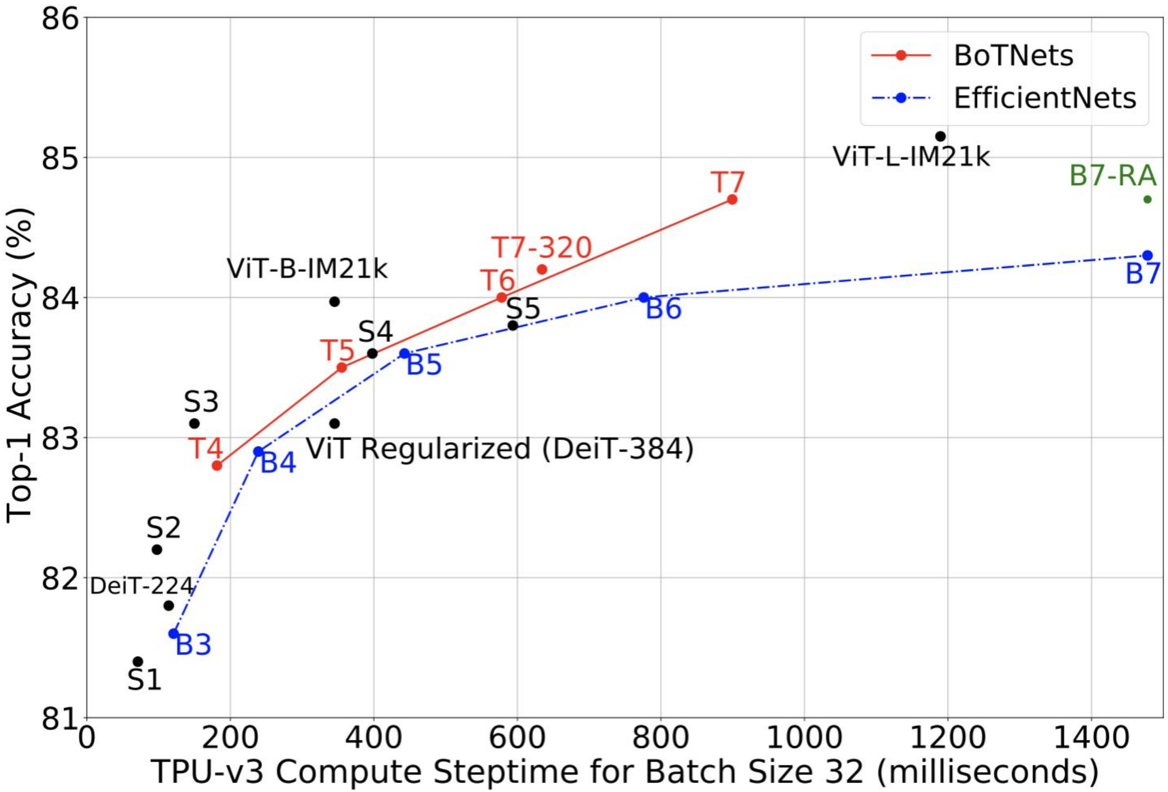
Data-Augmentation and Distillation are powerful for limited data settings



Touvron et al 2020

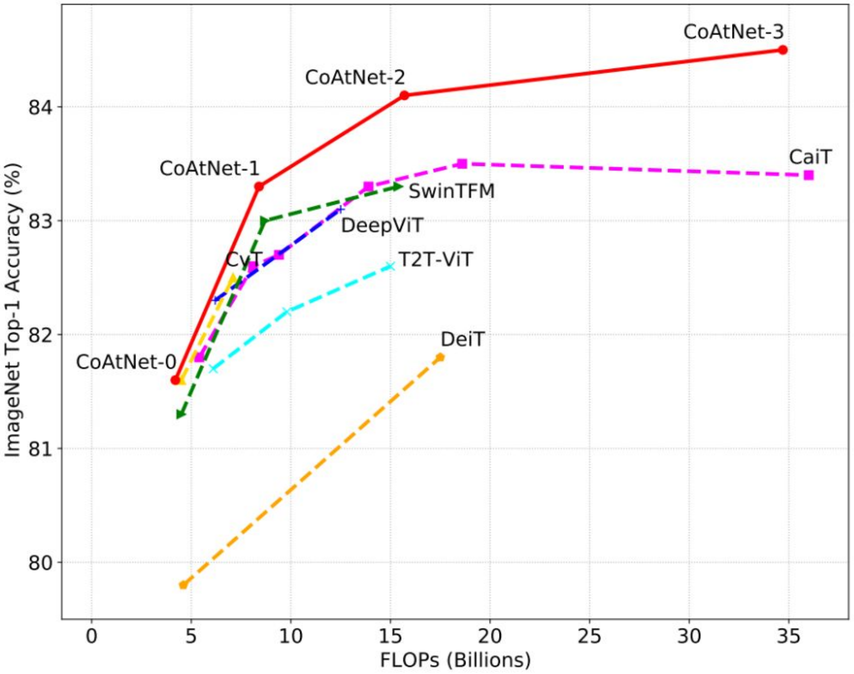
Hybrid Models: Combining Convolutions and Transformers (Self-Attention)

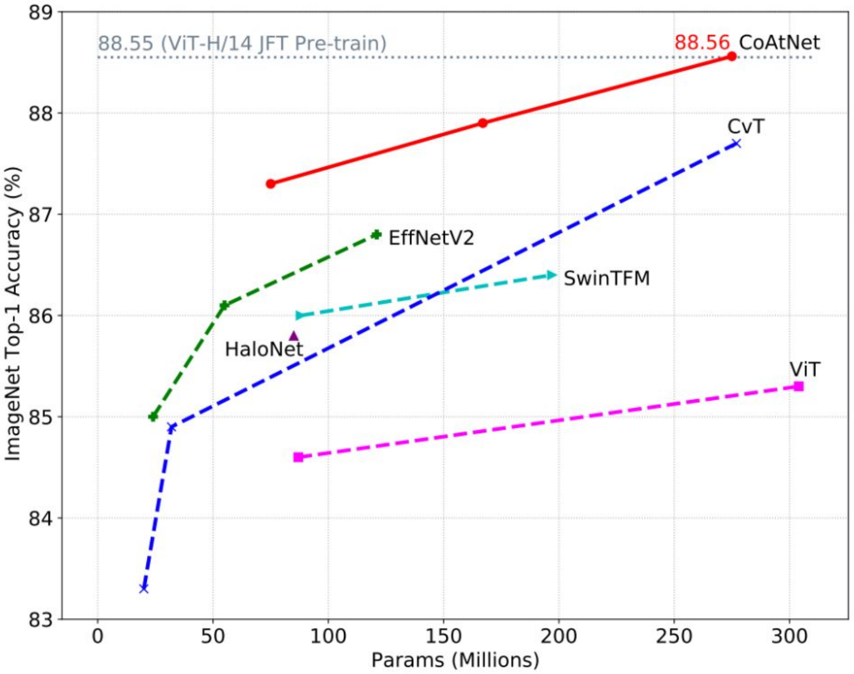
BoTNet: Bottleneck Transformers for Visual Recognition



Srinivas et al 2021

CoatNet: Marrying Convolutions and Attention for all Data Sizes





Dai et al 2021

HaloNet: Scaling Local Self-Attention Models for Vision

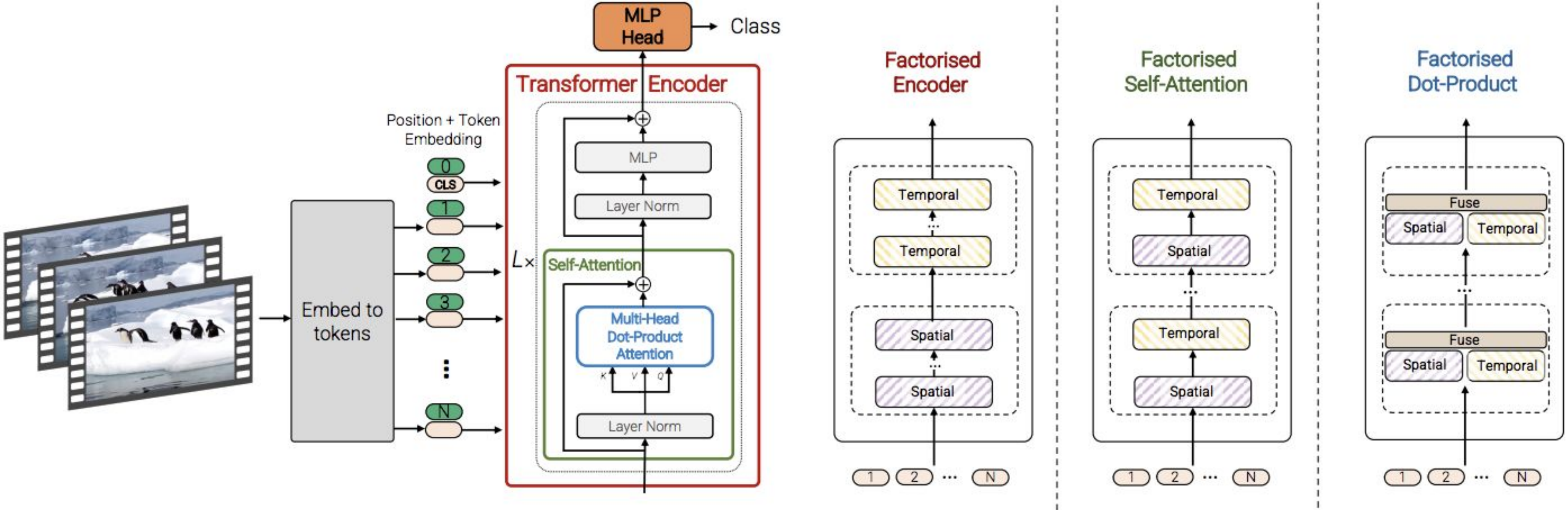
Model	Parameters (Millions)	Pretraining Image Size (Pixels)	Pretraining Step Time (32 per core)	Finetuning Image Size	Finetuning Top-1 Accuracy (%)	Inference Speed img/sec/core
H4 (base 128)	85	256	377 ms	384/512	85.6/85.8	121.3/48.6
H4 (base 128, 4×4 patch)	85	256	366 ms	384/512	85.4/85.4	125.7/56.5
H4 (base 128, Conv-12)	87	256	213 ms	384/512	85.5/85.8	257.6/120.2
ViT-L/16	300	224	445 ms	384/512	85.2/85.3	74.6/27.4
BiT-M	928	224	1021 ms	384	85.4	54.2

Vaswani et al 2021

Plenty of work in the field on hybrid models (ConViT, LeViT, CMT,……)

Transformers for Video Recognition

Video ViT



[Arnab et al 2021](#)

Video ViT

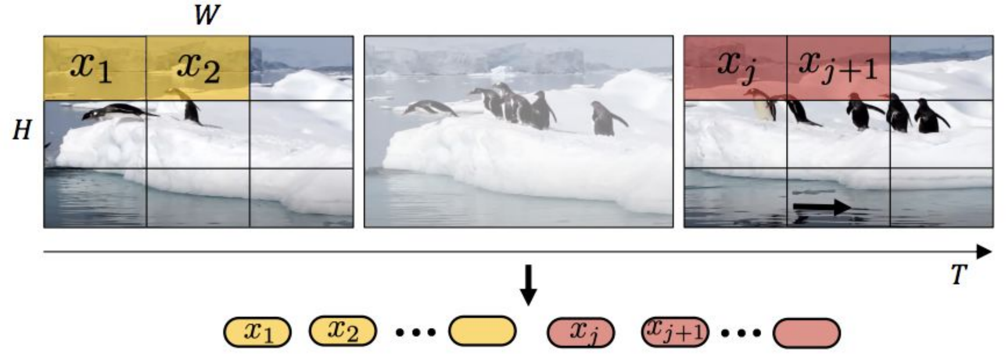
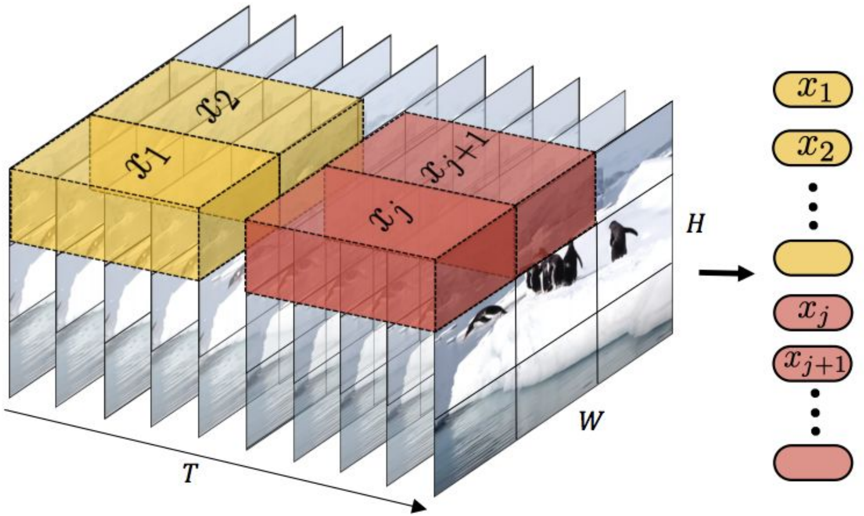
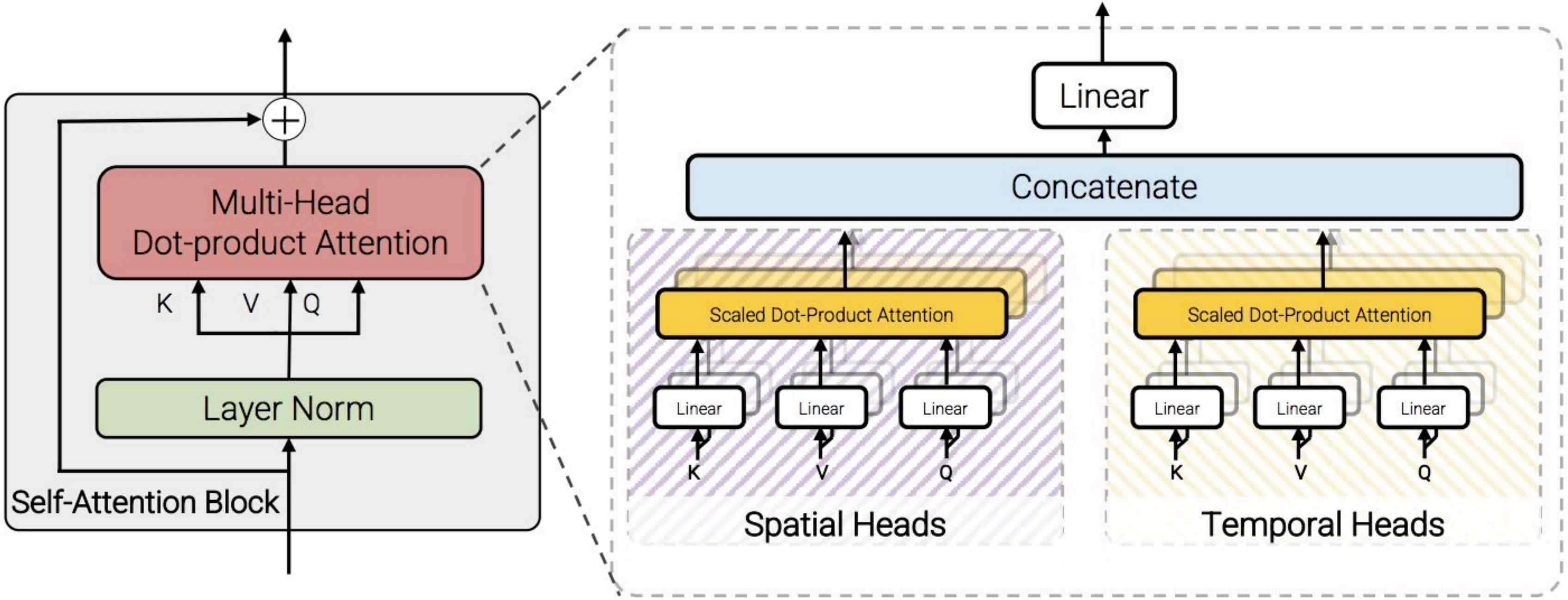


Figure 2: Uniform frame sampling: We simply sample n_t frames, and embed each 2D frame independently following ViT [15].



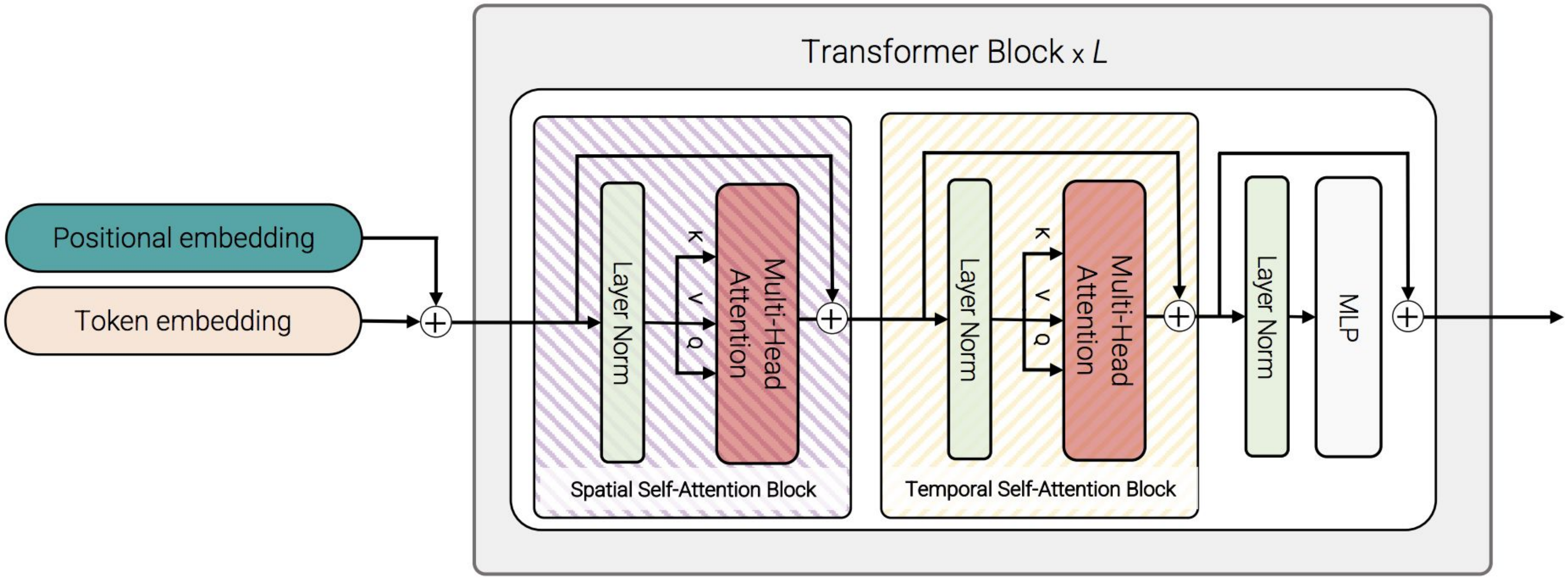
[Arnab et al 2021]

Video ViT



[Arnab et al 2021]

Video ViT



[Arnab et al 2021]

Video ViT

(a) Kinetics 400

Method	Top 1	Top 5	Views
bIVNet [16]	73.5	91.2	–
STM [30]	73.7	91.6	–
TEA [39]	76.1	92.5	10 × 3
TSM-ResNeXt-101 [40]	76.3	–	–
I3D NL [72]	77.7	93.3	10 × 3
CorrNet-101 [67]	79.2	–	10 × 3
ip-CSN-152 [63]	79.2	93.8	10 × 3
LGD-3D R101 [48]	79.4	94.4	–
SlowFast R101-NL [18]	79.8	93.9	10 × 3
X3D-XXL [17]	80.4	94.6	10 × 3
TimeSformer-L [2]	80.7	94.7	1 × 3
ViViT-L/16x2	80.6	94.7	4 × 3
ViViT-L/16x2 320	**81.3**	**94.7**	4 × 3
Methods with large-scale pretraining			
ip-CSN-152 [63] (IG [41])	82.5	95.3	10 × 3
ViViT-L/16x2 (JFT)	82.8	95.5	4 × 3
ViViT-L/16x2 320 (JFT)	83.5	95.5	4 × 3
ViViT-H/16x2 (JFT)	**84.8**	**95.8**	4 × 3

(b) Kinetics 600

Method	Top 1	Top 5	Views
AttentionNAS [73]	79.8	94.4	–
LGD-3D R101 [48]	81.5	95.6	–
SlowFast R101-NL [18]	81.8	95.1	10 × 3
X3D-XL [17]	81.9	95.5	10 × 3
TimeSformer-HR [2]	82.4	**96.0**	–
ViViT-L/16x2	82.5	95.6	4 × 3
ViViT-L/16x2 320	**83.0**	95.7	4 × 3
ViViT-L/16x2 (JFT)	84.3	96.2	4 × 3
ViViT-H/16x2 (JFT)	**85.8**	**96.5**	4 × 3

(c) Moments in Time

	Top 1	Top 5
TSN [69]	25.3	50.1
TRN [83]	28.3	53.4
I3D [6]	29.5	56.1
bIVNet [16]	31.4	59.3
AssembleNet-101 [51]	34.3	62.7
ViViT-L/16x2	**38.0**	**64.9**

(d) Epic Kitchens 100 Top 1 accuracy

Method	Action	Verb	Noun
TSN [69]	33.2	60.2	46.0
TRN [83]	35.3	65.9	45.4
TBN [33]	36.7	66.0	47.2
TSM [40]	38.3	**67.9**	49.0
SlowFast [18]	38.5	65.6	50.0
ViViT-L/16x2 Fact. encoder	**44.0**	66.4	**56.8**

(e) Something-Something v2

Method	Top 1	Top 5
TRN [83]	48.8	77.6
SlowFast [17, 77]	61.7	–
TimeSformer-HR [2]	62.5	–
TSM [40]	63.4	88.5
STM [30]	64.2	89.8
TEA [39]	65.1	–
bIVNet [16]	65.2	**90.3**
ViViT-L/16x2 Fact. encoder	**65.4**	89.8

Arnab et al 2021

Takeaways for practitioners

- Pure attention models require a lot of data OR data-augmentations and regularization for ~SoTA performance

- Hybrid and (or) multi-scale models perform best (efficient for the same high accuracy) across all data regimes

- Huge promise for multimodal (combining with language)

- Good Resource: https://github.com/rwightman/pytorch-image-models