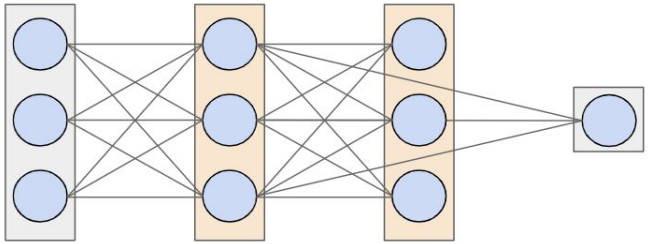
Lecture 2:
Neural Networks for Image Data:
From CNNs to Transformers

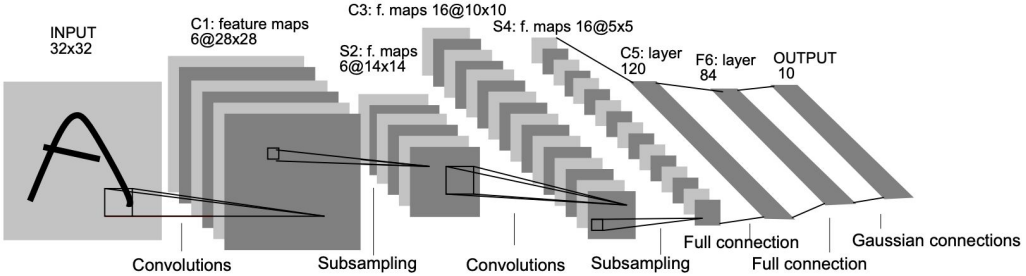
Today’s agenda

- Review of convolutional neural networks
- Vision Transformers (ViTs)

Traditional classes of neural networks

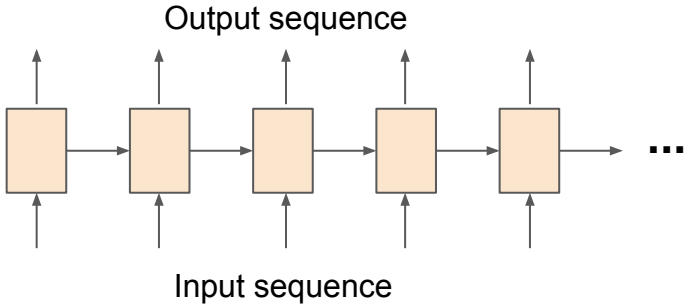


Fully connected neural networks
(linear layers, good for “feature vector” inputs)

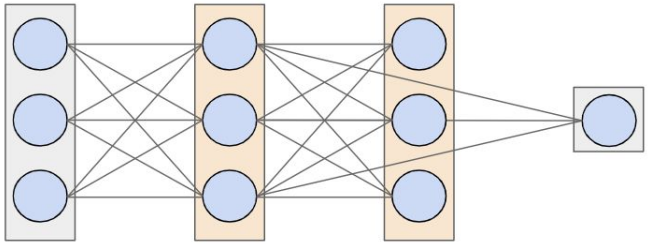


Convolutional neural networks
(convolutional layers, good for image inputs)

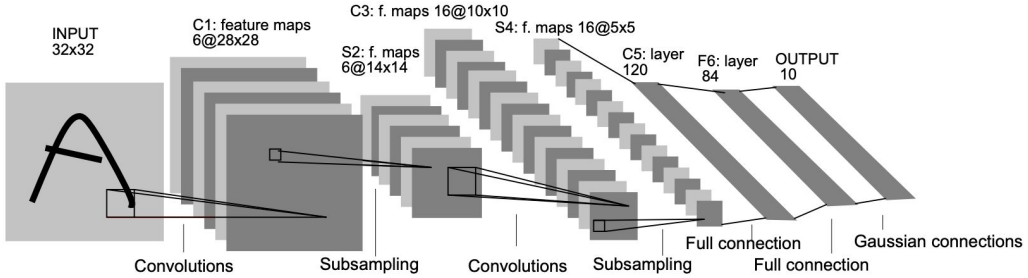
Recurrent neural networks
(linear layers modeling recurrence relation across sequence, good for sequence inputs)



Traditional classes of neural networks



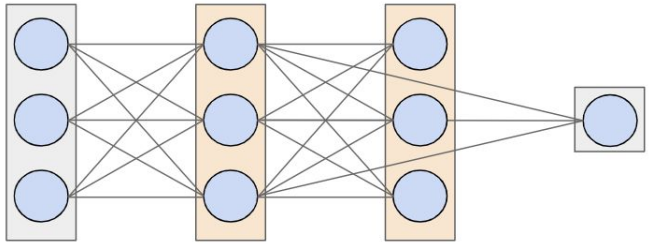
Fully connected neural networks
(linear layers, good for “feature vector” inputs)



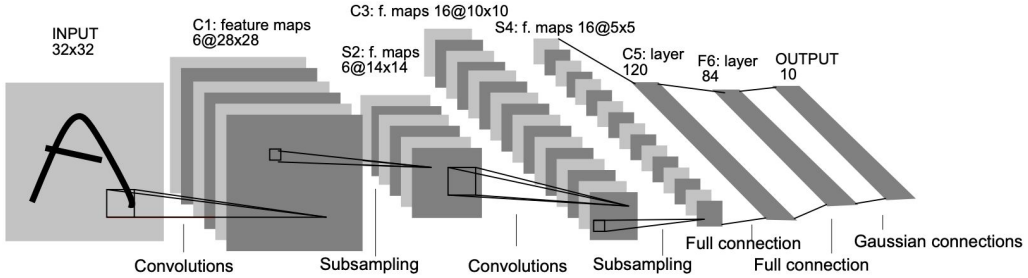
Convolutional neural networks
(convolutional layers, good for image inputs)

In the past few years, **Transformer neural networks** have become the dominant architecture for modeling sequence data – will discuss more later today

Traditional classes of neural networks



Fully connected neural networks
(linear layers, good for “feature vector” inputs)

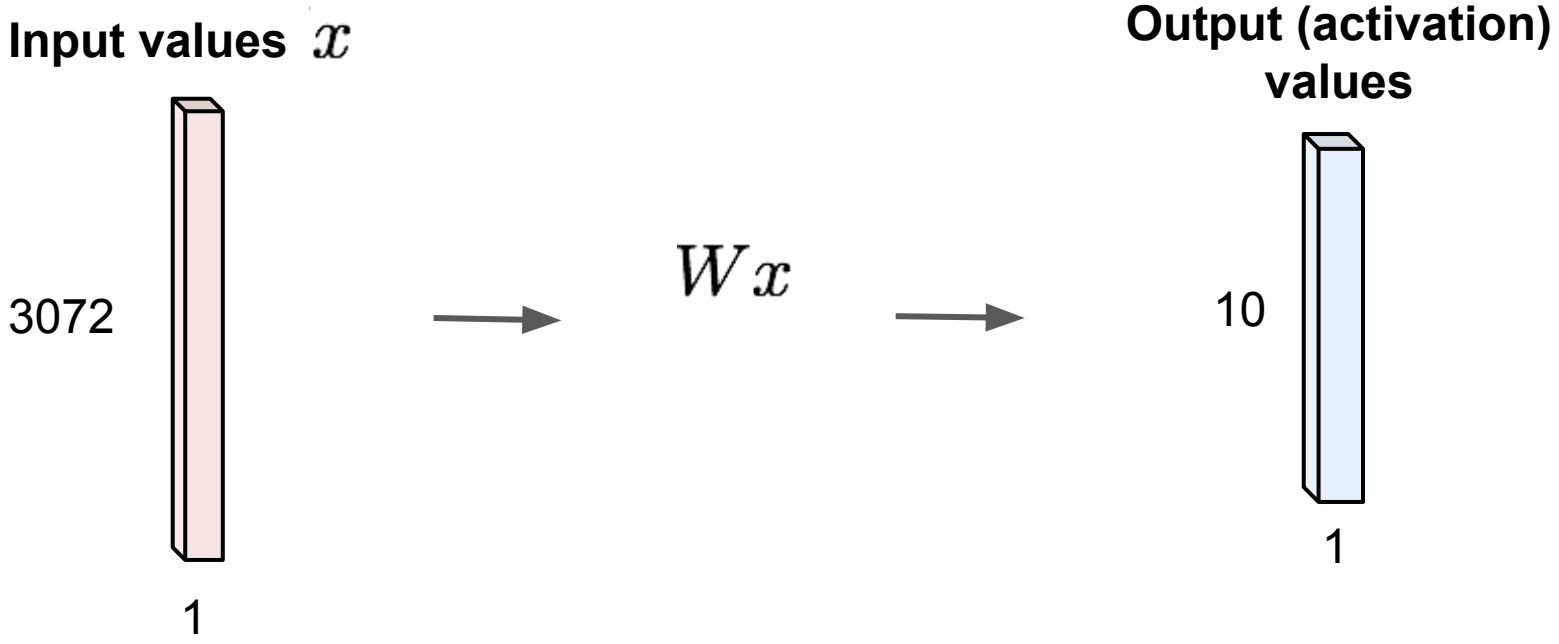


Convolutional neural networks
(convolutional layers, good for image inputs)

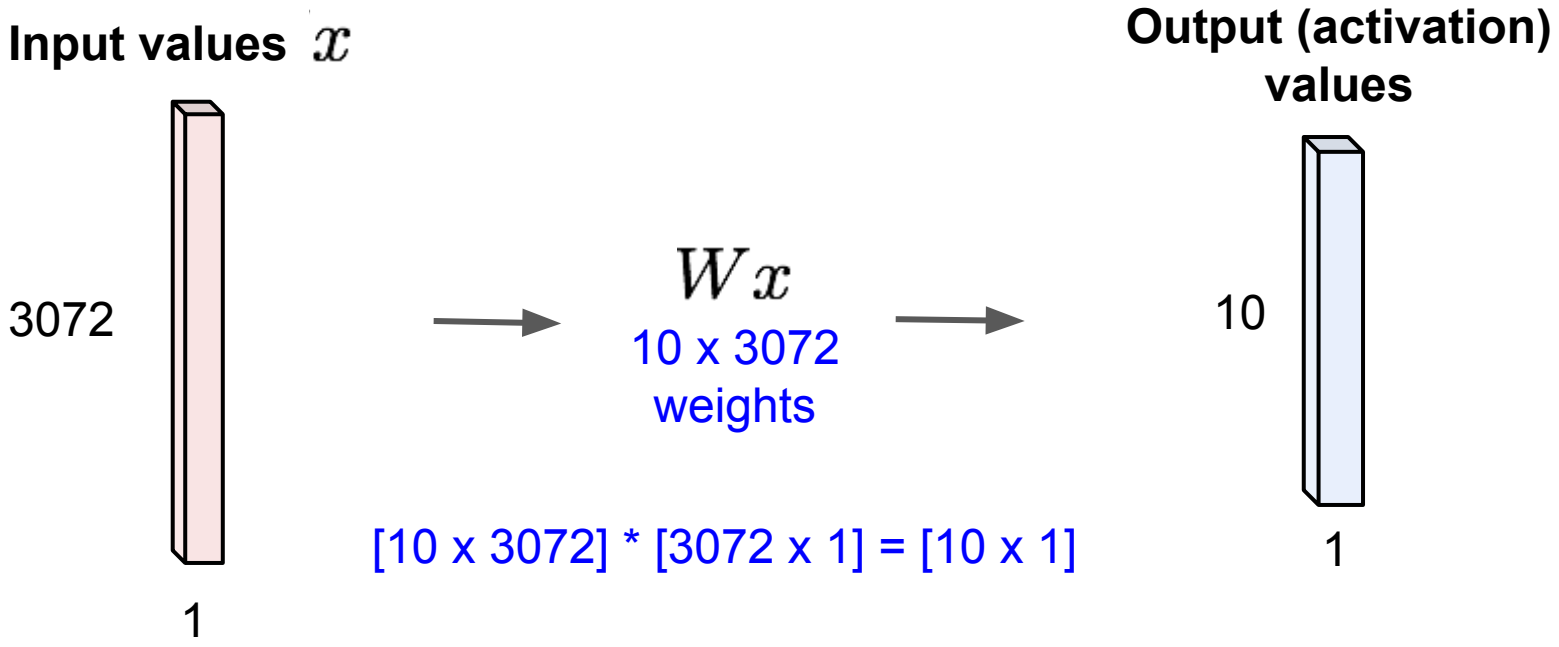
In the past few years, **Transformer neural networks** have become the dominant architecture for modeling sequence data – will discuss more later today

Remember: fully connected neural network layers

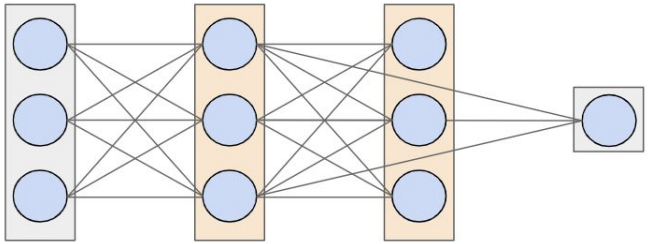
Remember: fully connected neural network layers



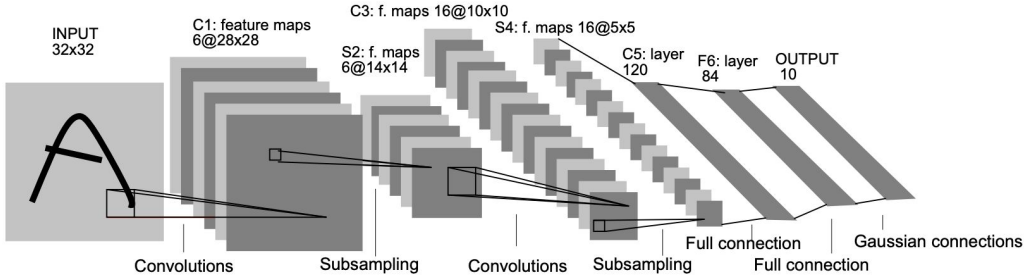
Remember: fully connected neural network layers



Traditional classes of neural networks



Fully connected neural networks
(linear layers, good for “feature vector” inputs)

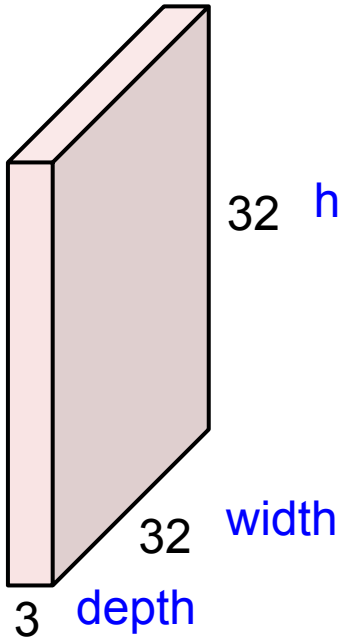


Convolutional neural networks
(convolutional layers, good for image inputs)

In the past few years, **Transformer neural networks** have become the dominant architecture for modeling sequence data – will discuss more later today

Now: Convolutional layer

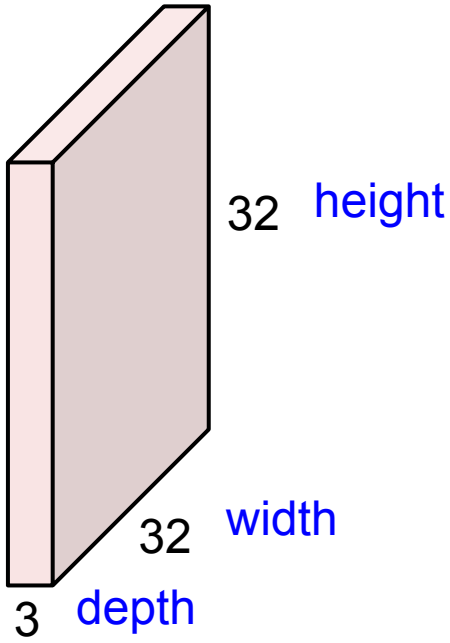
32x32x3 image -> preserve spatial structure



Slide credit: CS231n

Convolutional layer

32x32x3 image -> preserve spatial structure



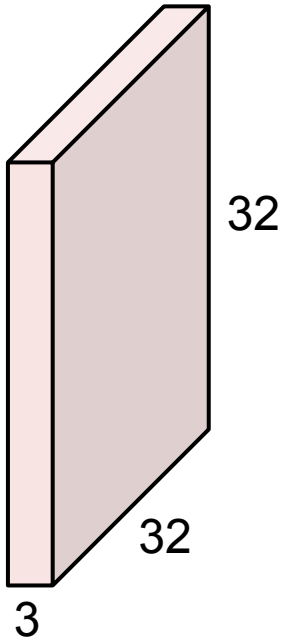
Input now has spatial height and width dimensions!

In contrast to fully-connected layers, want to preserve spatial structure when processing with a convolutional layer

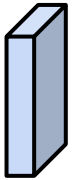
Slide credit: CS231n

Convolutional layer

32x32x3 image



5x5x3 filter (weights)



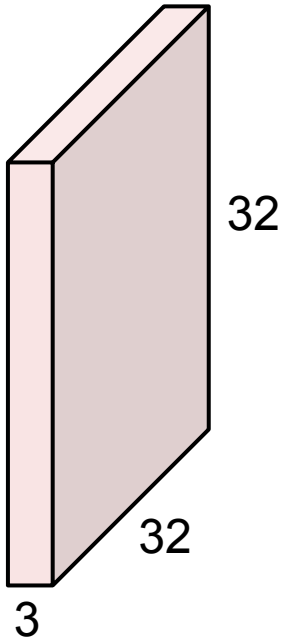
Convolve the filter with the image i.e. “slide over the image spatially, computing dot products”

Slide credit: CS231n

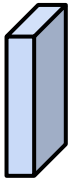
Convolutional layer

Filters always extend the full depth of the input volume

32x32x3 image



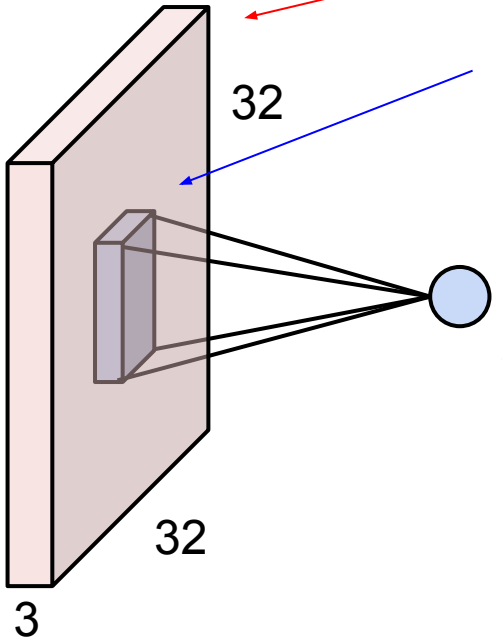
5x5x3 filter (weights)



Convolve the filter with the image i.e. “slide over the image spatially, computing dot products”

Slide credit: CS231n

Convolutional layer



32x32x3 image

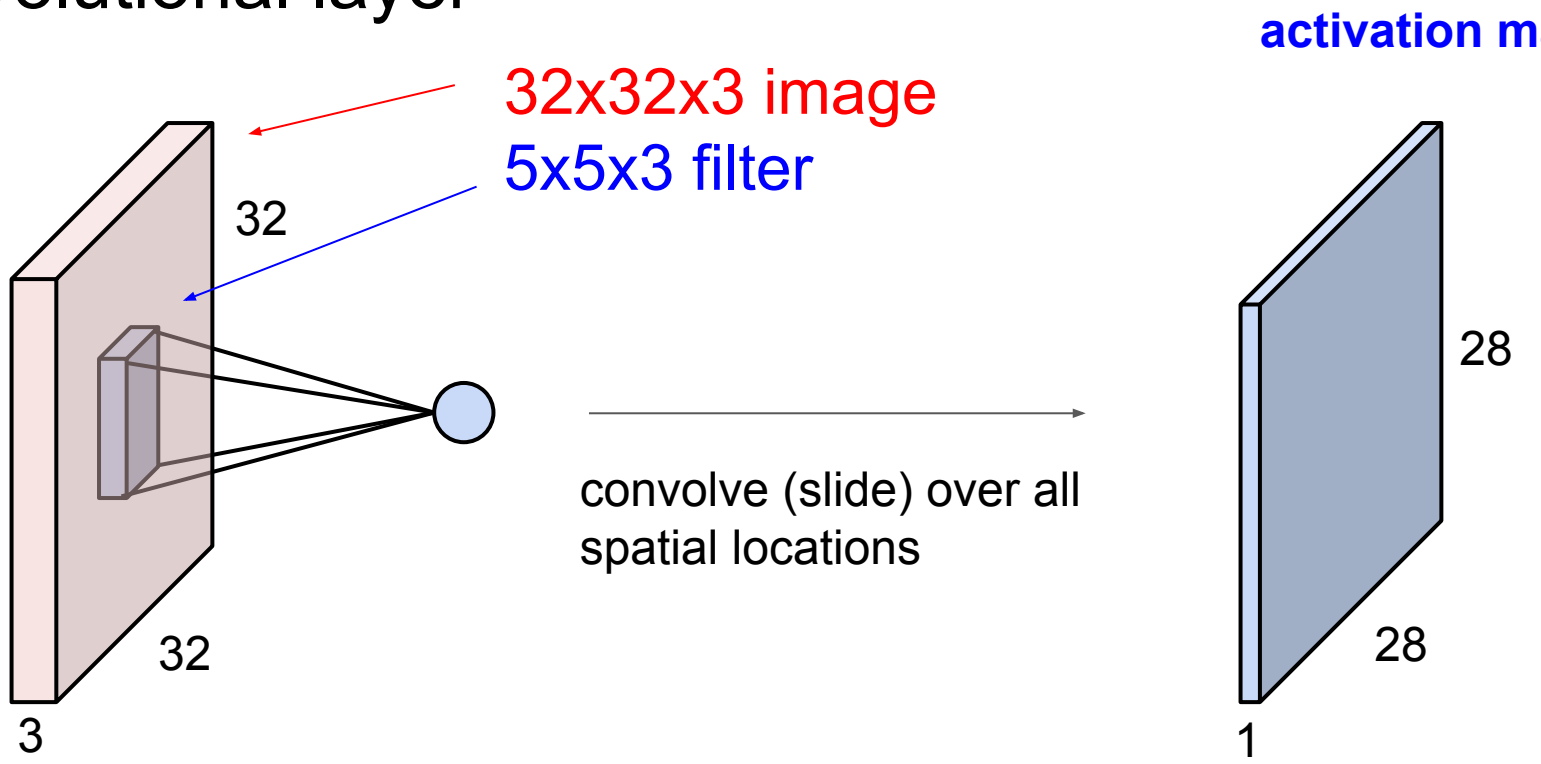
5x5x3 filter w

1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image (i.e. 5*5*3 = 75-dimensional dot product + bias)

$$w^T x + b$$

Slide credit: CS231n

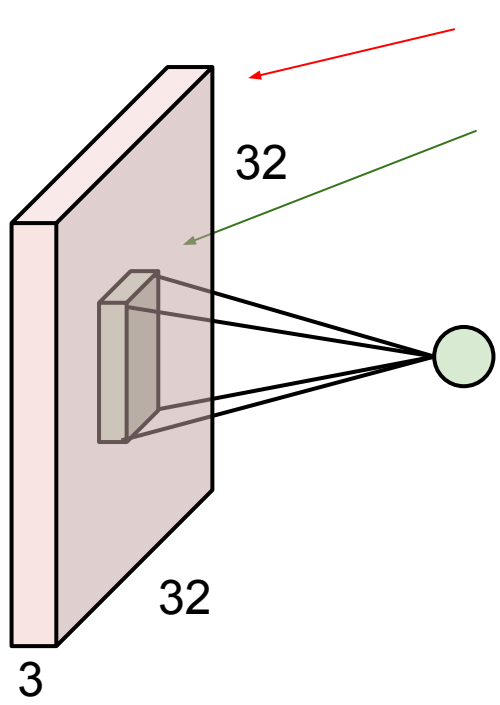
Convolutional layer



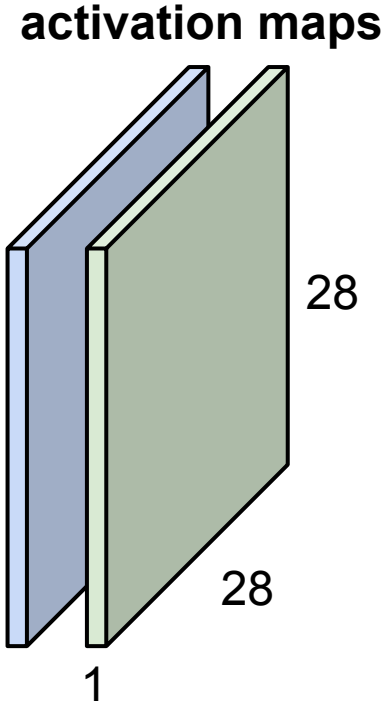
Slide credit: CS231n

consider a second, green filter

Convolutional layer

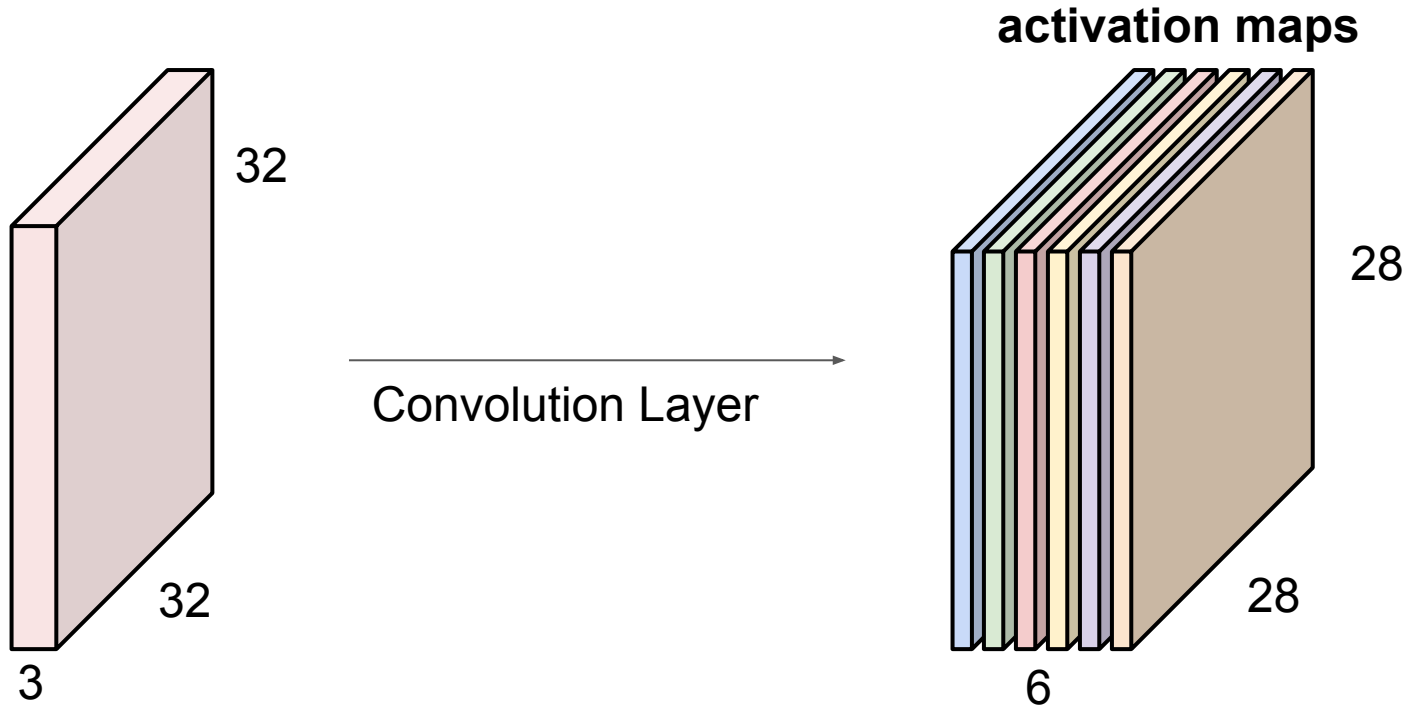


convolve (slide) over all spatial locations



Slide credit: CS231n

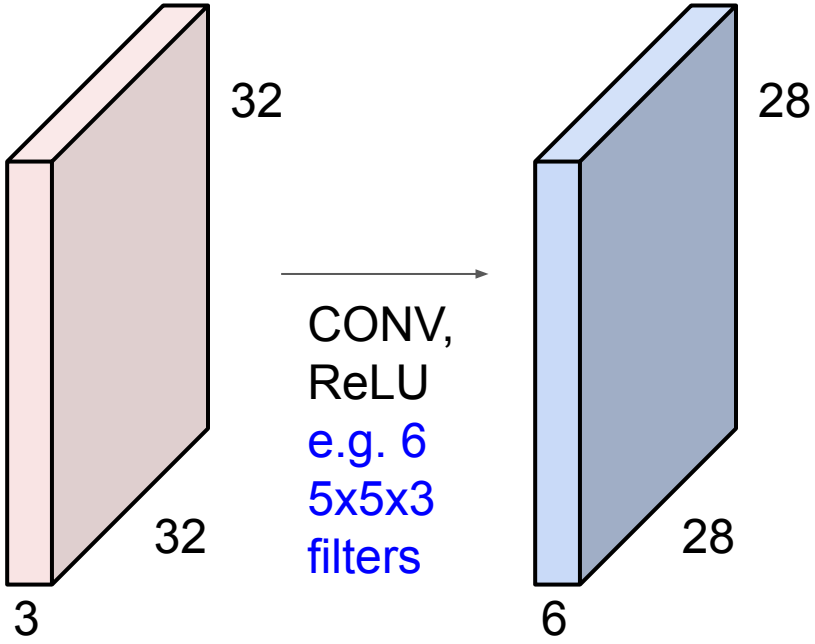
For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:



We stack these up to get a "new image" of size 28x28x6!

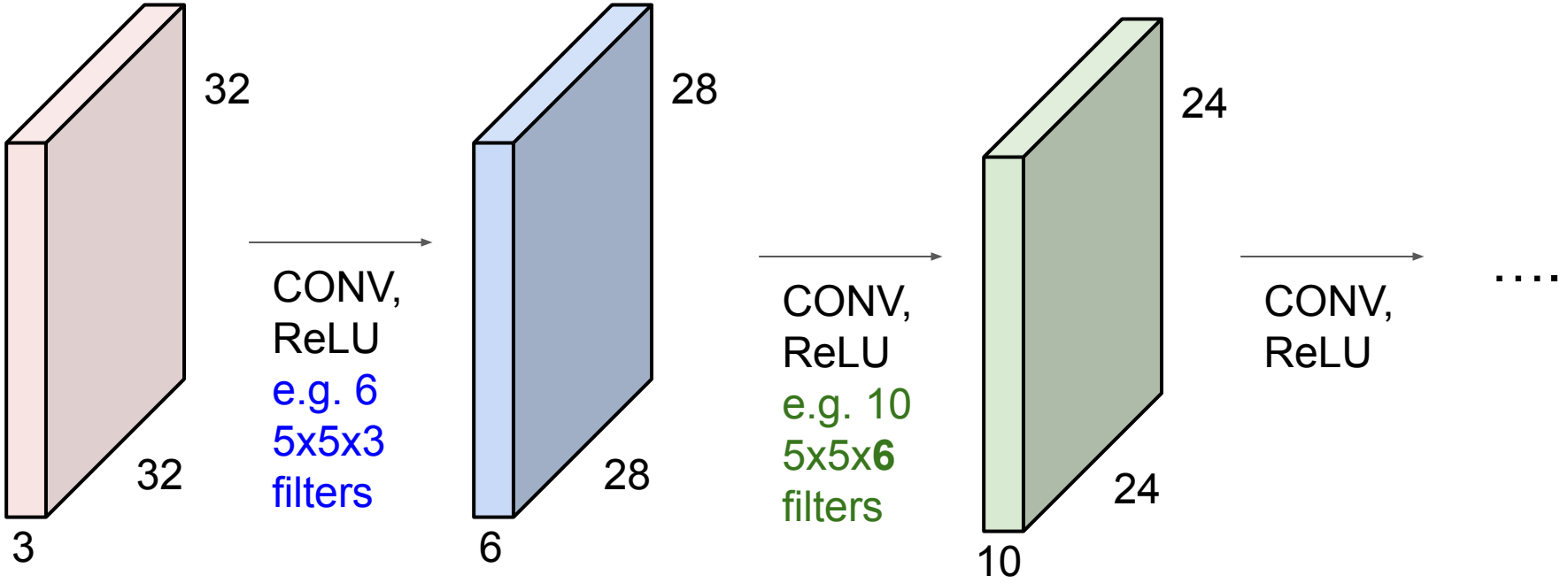
Slide credit: CS231n

A ConvNet (or CNN) is a sequence of Convolution Layers, interspersed with activation functions



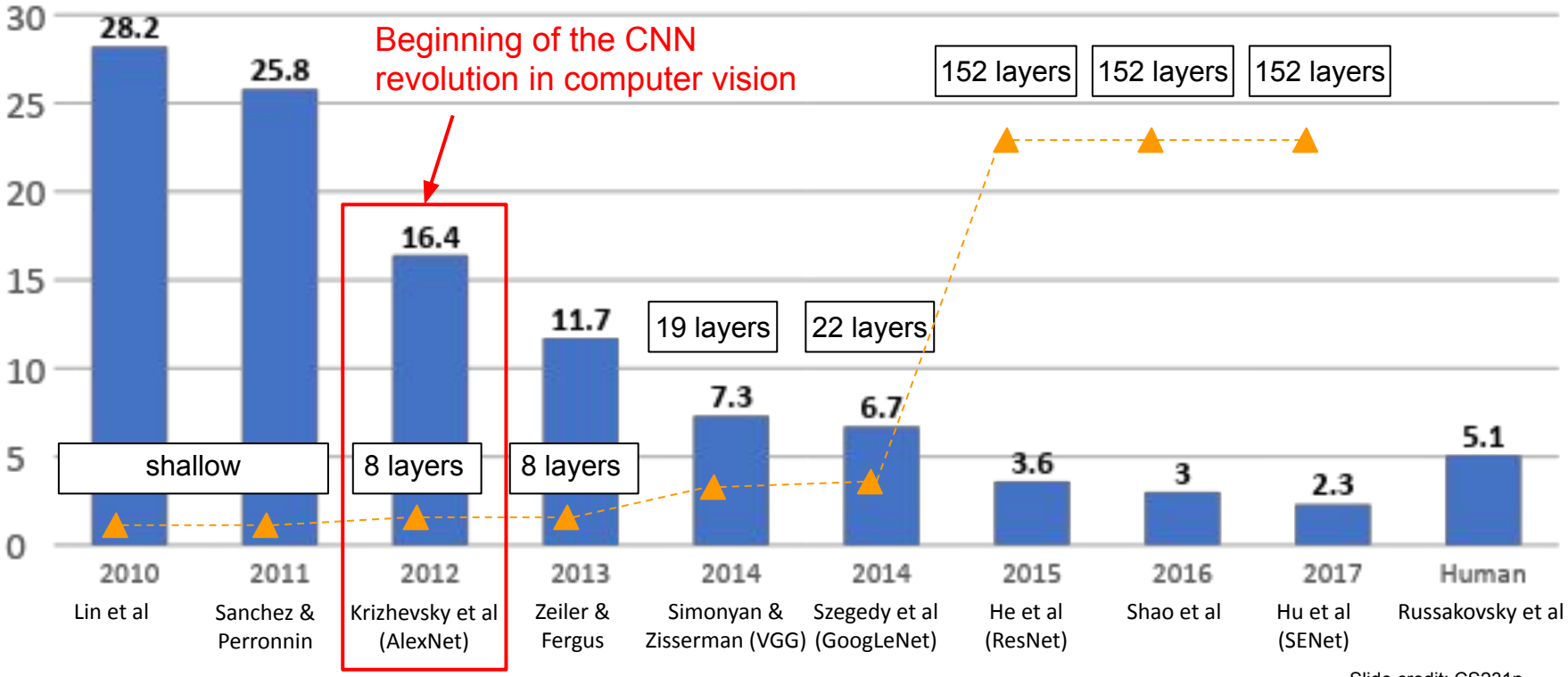
Slide credit: CS231n

A ConvNet (or CNN) is a sequence of Convolution Layers, interspersed with activation functions



Slide credit: CS231n

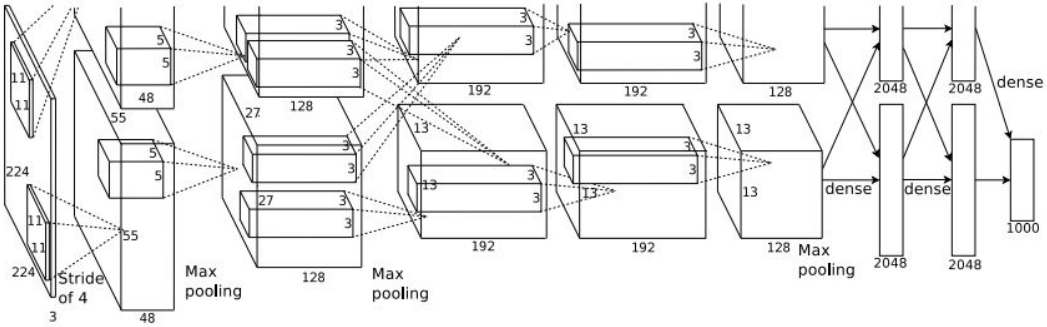
ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



Slide credit: CS231n

AlexNet

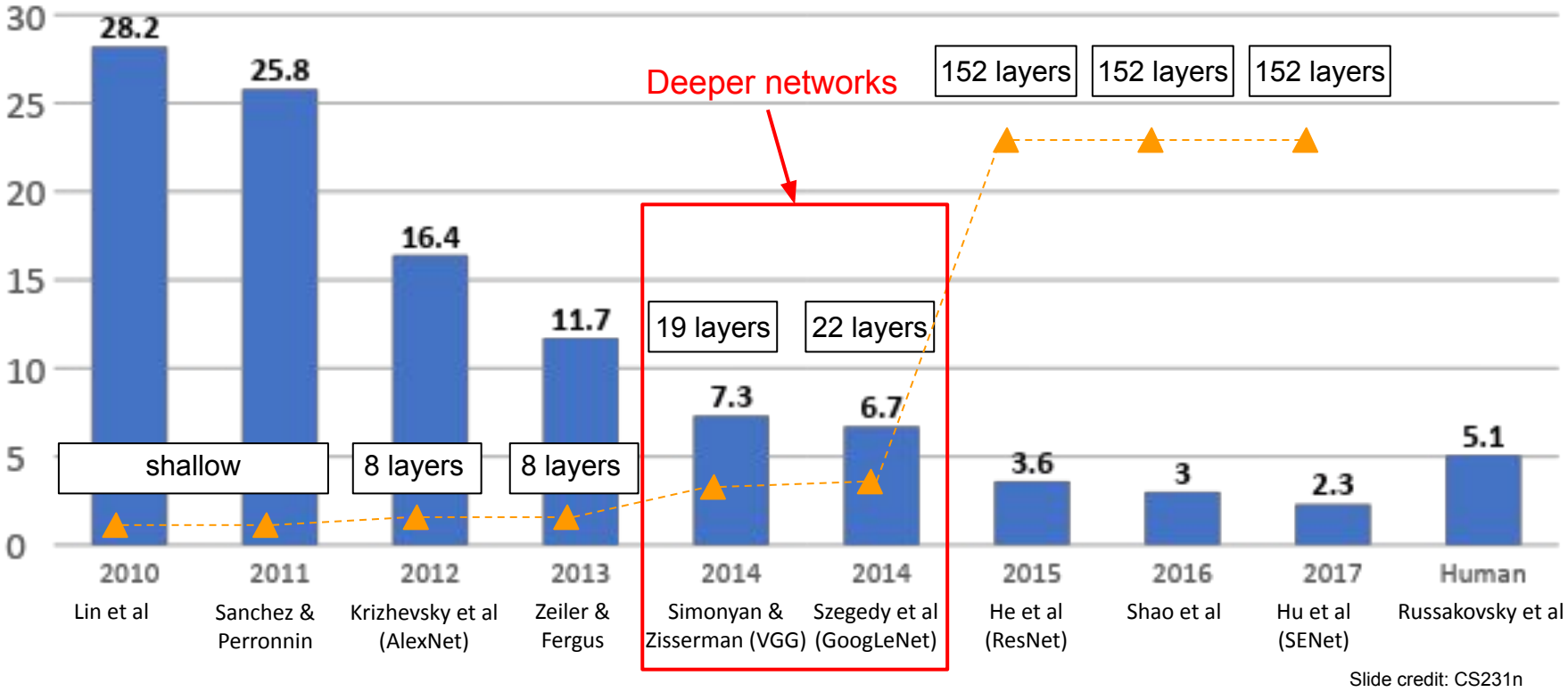
[Krizhevsky et al. 2012]



Full (simplified) AlexNet architecture:
[227x227x3] INPUT
[55x55x96] CONV1: 96 11x11 filters at stride 4, pad 0
[27x27x96] MAX POOL1: 3x3 filters at stride 2
[27x27x96] NORM1: Normalization layer
[27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2
[13x13x256] MAX POOL2: 3x3 filters at stride 2
[13x13x256] NORM2: Normalization layer
[13x13x384] CONV3: 384 3x3 filters at stride 1, pad 1
[13x13x384] CONV4: 384 3x3 filters at stride 1, pad 1
[13x13x256] CONV5: 256 3x3 filters at stride 1, pad 1
[6x6x256] MAX POOL3: 3x3 filters at stride 2
[4096] FC6: 4096 neurons
[4096] FC7: 4096 neurons
[1000] FC8: 1000 neurons (class scores)

Slide credit: CS231n

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



Slide credit: CS231n

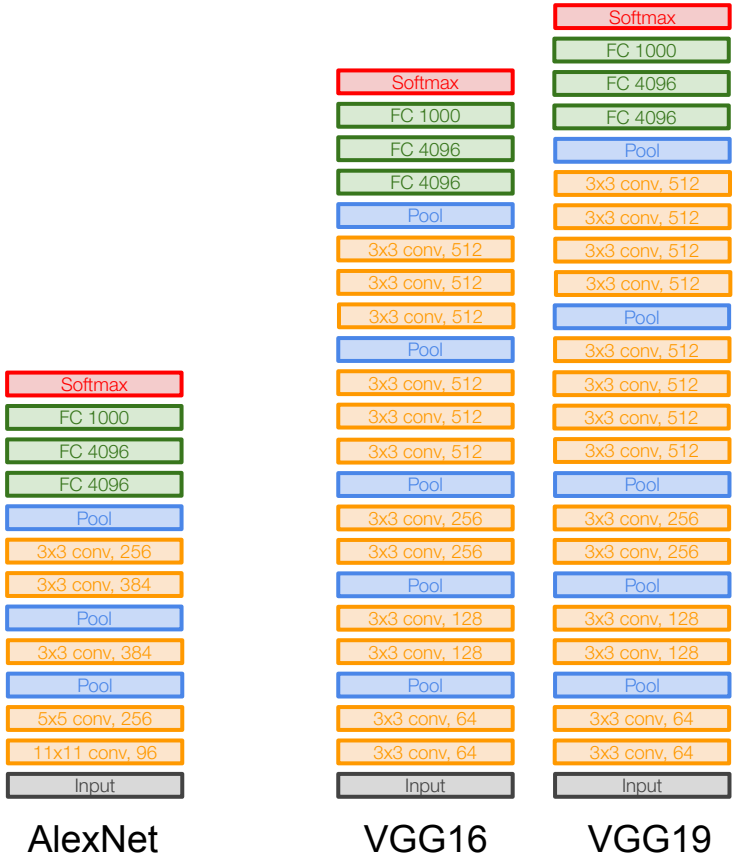
VGGNet

[Simonyan and Zisserman, 2014]

Small filters, Deeper networks

8 layers (AlexNet)
-> 16 - 19 layers (VGG16Net)

Only 3x3 CONV stride 1, pad 1
and 2x2 MAX POOL stride 2

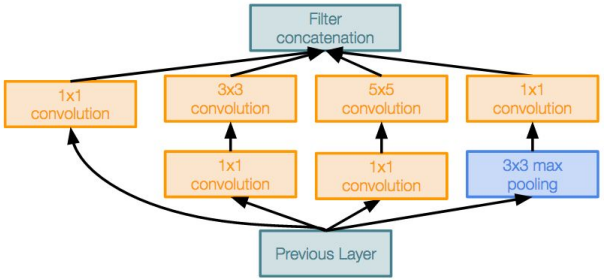


Slide credit: CS231n

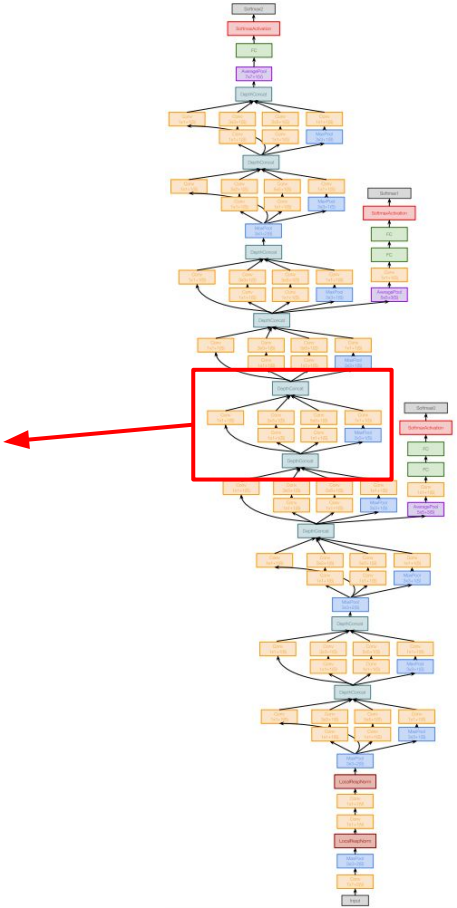
GoogLeNet

[Szegedy et al., 2014]

"Inception module": design a good local network topology (network within a network) and then stack these modules on top of each other

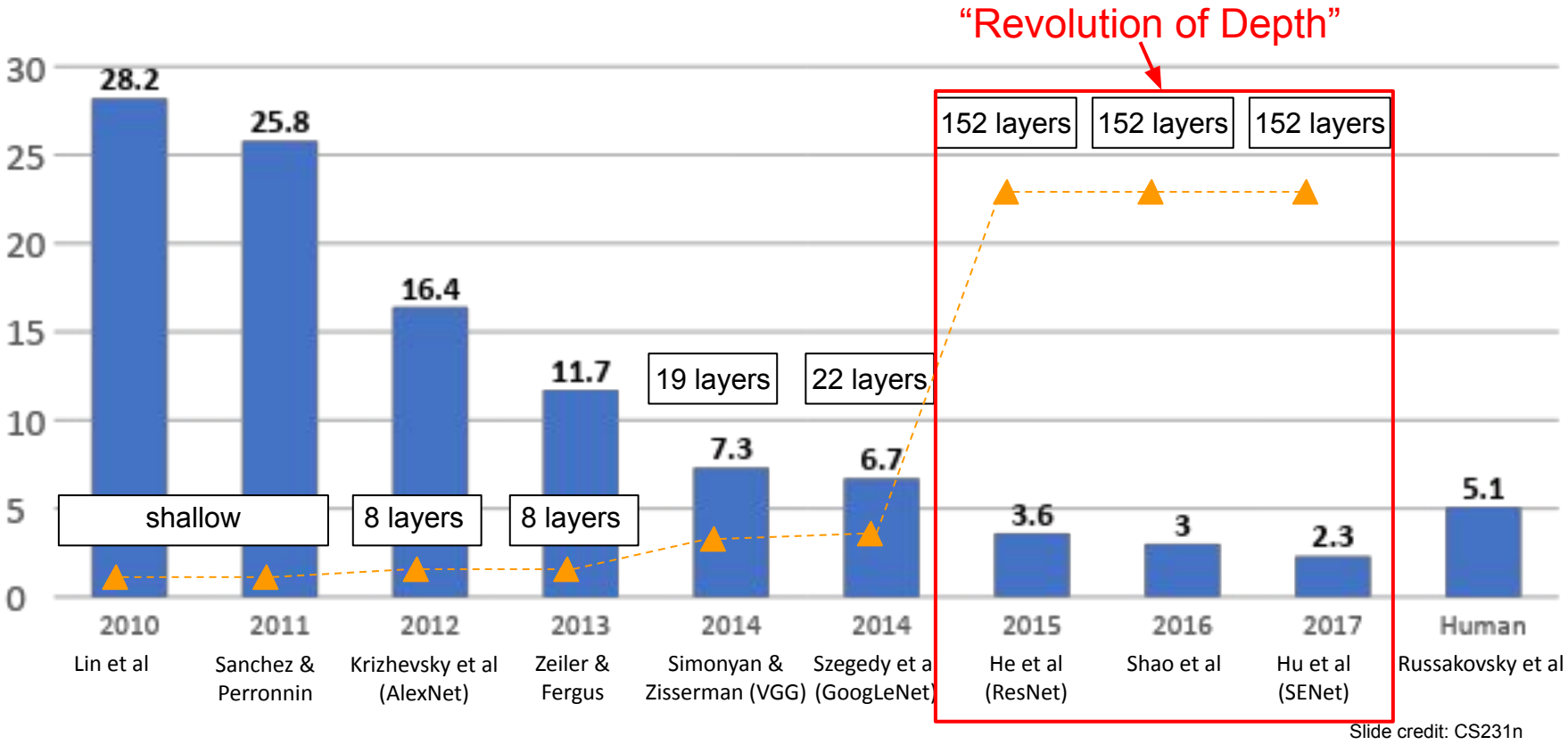


Inception module



Slide credit: CS231n

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



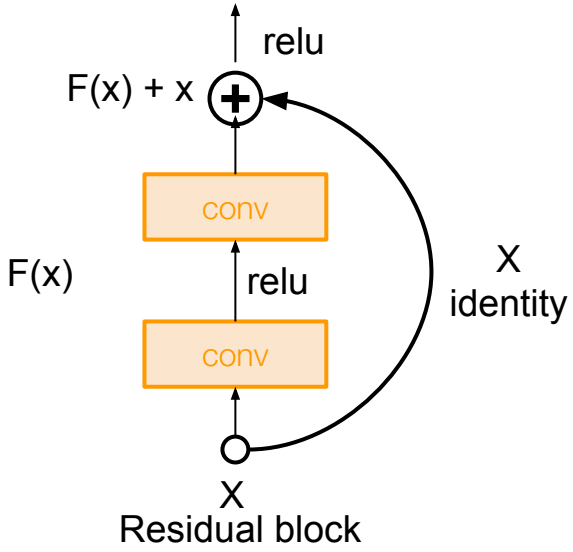
Slide credit: CS231n

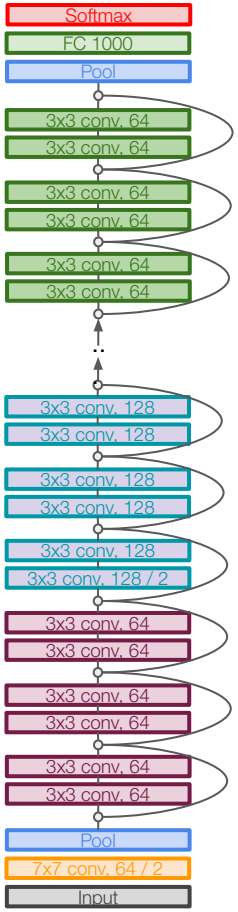
ResNet

[He et al., 2015]

Very deep networks using residual connections

- 152-layer model for ImageNet
- Won all major classification and detection benchmark challenges in 2015

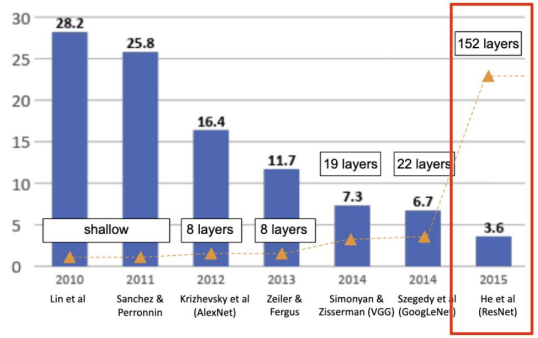




Slide credit: CS231n

Remember from Lecture 1…

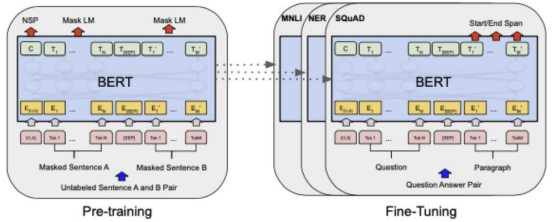
2015: Very deep convnets and challenging vision tasks



He et al. 2015. ResNet.

2018: Breakthroughs in deep learning for natural language processing (sequences)

Transformer neural network architectures (containing self-attention mechanism) and pre-training -> fine-tuning. State-of-the-art on 11 NLP benchmarks.



Vaswani et al. 2017. "Attention is all you need."
Devlin et al. 2018. BERT.

Remember from Lecture 1…

2020: Very large-scale text generation models

SYSTEM PROMPT (HUMAN-WRITTEN)

In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.

MODEL COMPLETION (MACHINE-WRITTEN, 10 TRIES)

The scientist named the population, after their distinctive horn, Ovid's Unicorn. These four-horned, silver-white unicorns were previously unknown to science.

Now, after almost two centuries, the mystery of what sparked this odd phenomenon is finally solved.

Dr. Jorge Pérez, an evolutionary biologist from the University of La Paz, and several companions, were exploring the Andes Mountains when they found a small valley, with no other animals or humans. Pérez noticed that the valley had what appeared to be a natural fountain, surrounded by two peaks of rock and silver snow.

Transformer architectures become ubiquitous

Training these models on very large amounts of data -> rise of "foundation models"

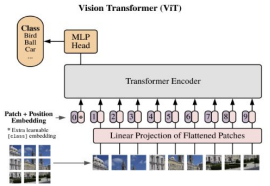
OpenAI GPT-3 (figure from GPT-2). Brown et al. 2020.

2021: Transformers & foundation models reach computer vision

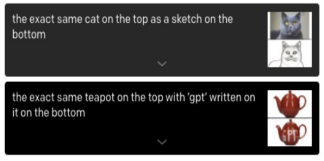
Transformer architecture can be applied to images as well!

Key idea: Convert image into sequence of patches (i.e. image tokens).

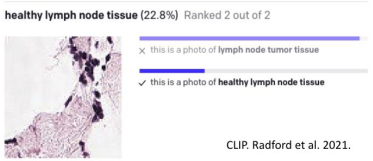
Can then benefit from Transformer architecture and its self-attention mechanism. Allows process of token encoding to incorporate contextual understanding (attention) from other tokens of the same (self) sequence.



ViT. Dosovitsky et al. 2021



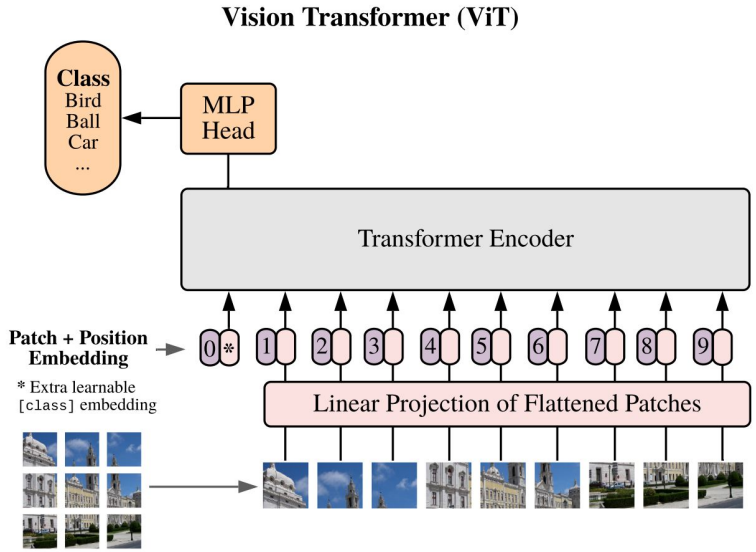
DALL-E. Ramesh et al. 2021.



CLIP. Radford et al. 2021.

A more detailed look at Vision Transformers (ViT)

Key idea: Convert image into sequence of patches. Can then benefit from Transformer architecture and self-attention, which jointly attends over all patches



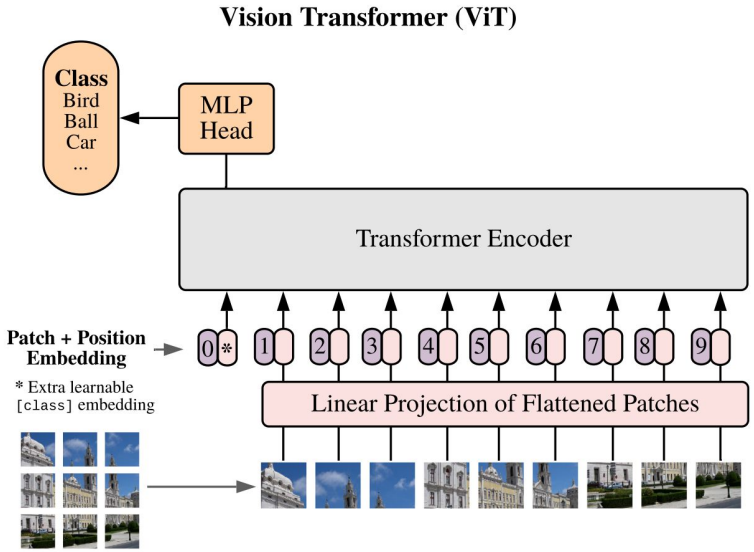
Dosovitsky et al. 2021

A more detailed look at Vision Transformers (ViT)

Key idea: Convert image into sequence of patches. Can then benefit from Transformer architecture and self-attention, which jointly attends over all patches

Reshape HxWxC image into size PxP patches (e.g. 16x16, 32x32).

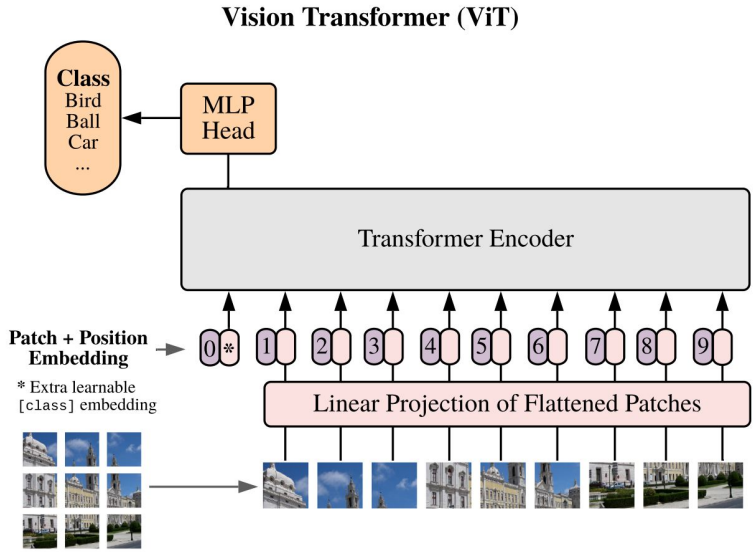
With 16x16, a 224x224 image would produce 196 patches, and a 384x384 image would produce 576 patches



Dosovitsky et al. 2021

A more detailed look at Vision Transformers (ViT)

Key idea: Convert image into sequence of patches. Can then benefit from Transformer architecture and self-attention, which jointly attends over all patches

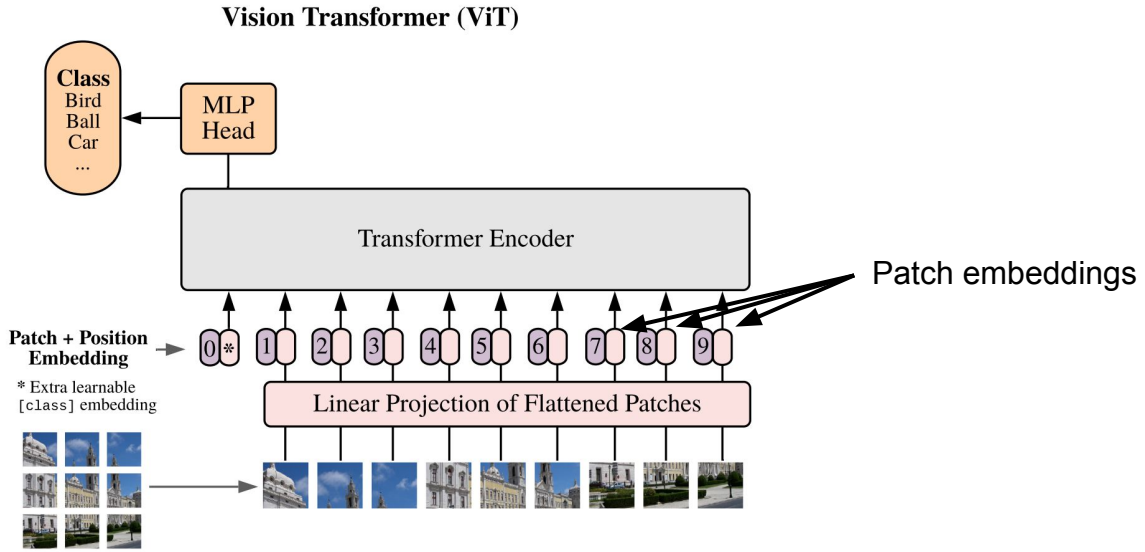


A trainable linear projection layer (fully connected layer) maps each PxP patch to a D-dimensional patch embedding (e.g., D = 768 or 1024)

Dosovitsky et al. 2021

A more detailed look at Vision Transformers (ViT)

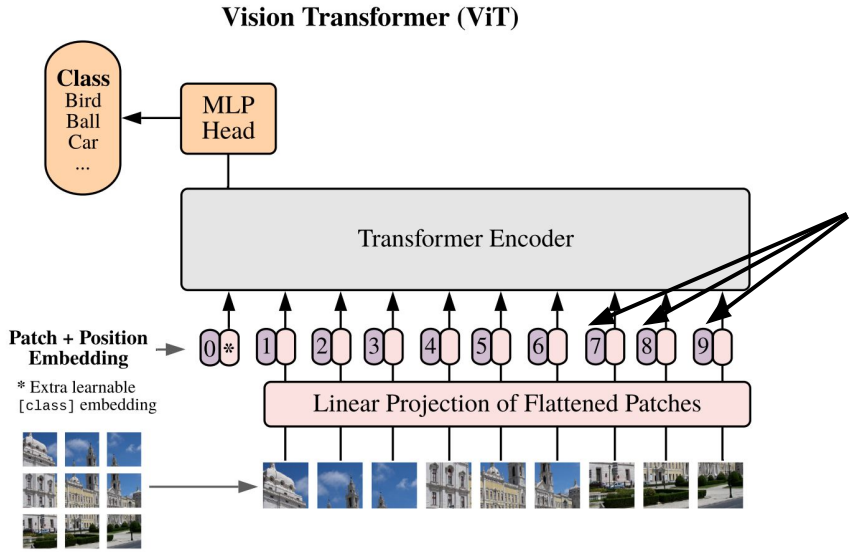
Key idea: Convert image into sequence of patches. Can then benefit from Transformer architecture and self-attention, which jointly attends over all patches



Dosovitsky et al. 2021

A more detailed look at Vision Transformers (ViT)

Key idea: Convert image into sequence of patches. Can then benefit from Transformer architecture and self-attention, which jointly attends over all patches



Dosovitsky et al. 2021

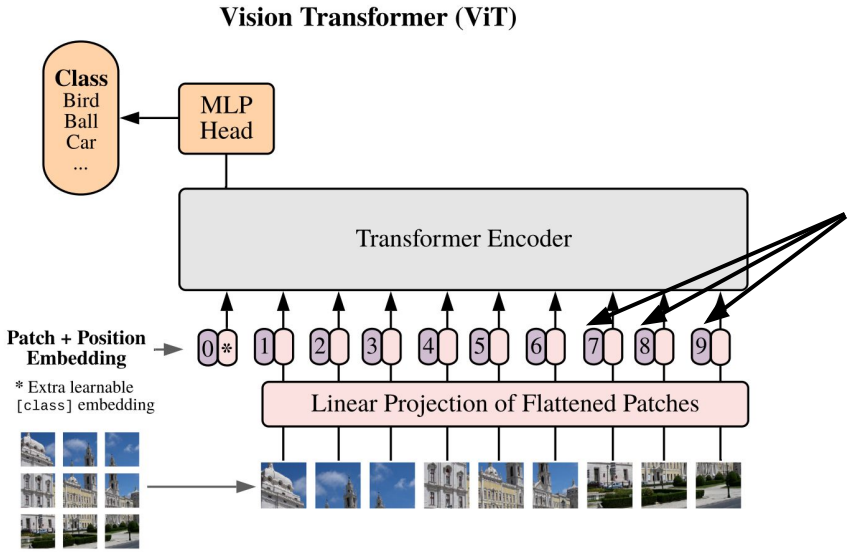
A position embedding is added to each patch embedding to provide information about patch order. Otherwise, given the Transformer structure, this order information is lost.

The position embedding has the same D-dimensional size as the patch embeddings. For ViT, these are learned embeddings (a learnable vector at each of L possible positions, for max sequence length L).

A more detailed look at Vision Transformers (ViT)

Key idea: Convert image into sequence of patches. Can then benefit from Transformer architecture and self-attention, which jointly attends over all patches

An active area of research is how to optimally encode positional information that can scale and generalize well to long sequences. Examples include sinusoidal embeddings, learned embeddings, relative embeddings, rotary embeddings, etc.



Dosovitsky et al. 2021

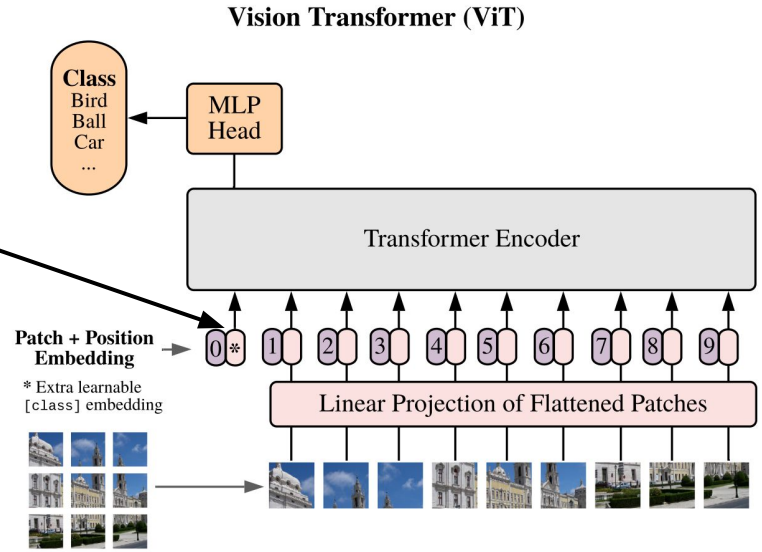
A position embedding is added to each patch embedding to provide information about patch order. Otherwise, given the Transformer structure, this order information is lost.

The position embedding has the same D-dimensional size as the patch embeddings. For ViT, these are learned embeddings (a learnable vector at each of L possible positions, for max sequence length L).

A more detailed look at Vision Transformers (ViT)

Key idea: Convert image into sequence of patches. Can then benefit from Transformer architecture and self-attention, which jointly attends over all patches

Include an extra “class token” (a learnable vector) whose role will be to summarize information from the entire input.

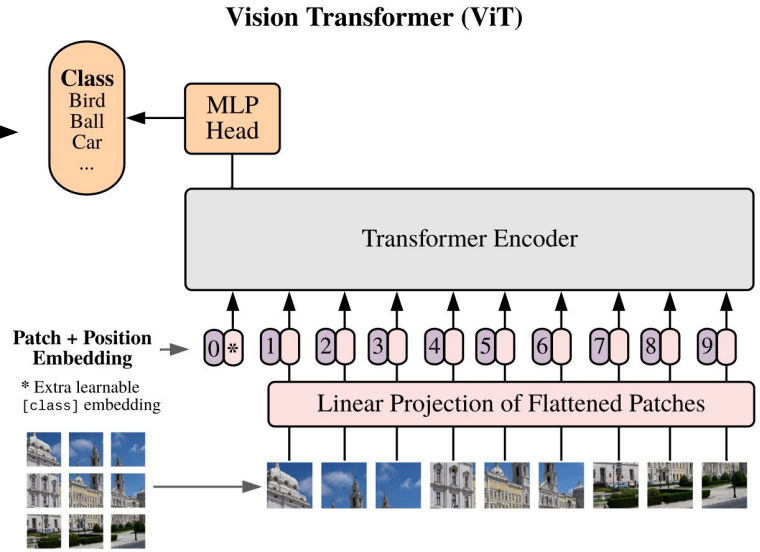


Dosovitsky et al. 2021

A more detailed look at Vision Transformers (ViT)

Key idea: Convert image into sequence of patches. Can then benefit from Transformer architecture and self-attention, which jointly attends over all patches

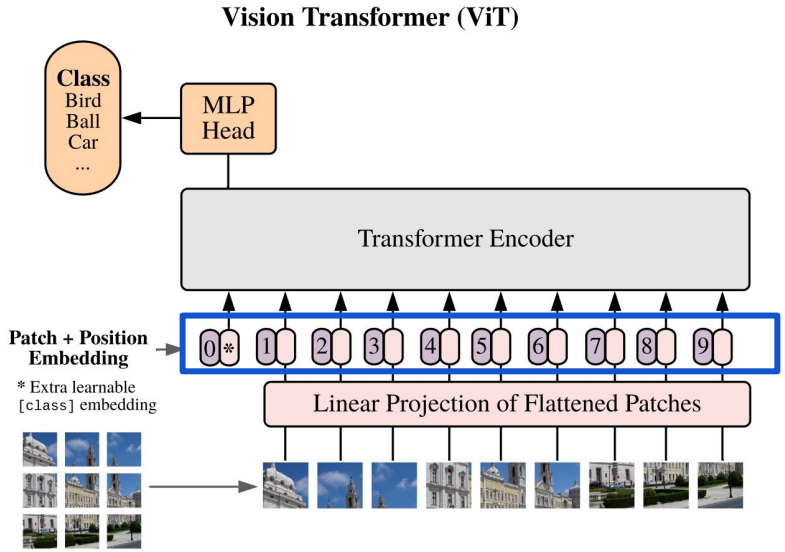
During training and inference, global, image-level predictions will come from the encoder output at the class token position



Dosovitsky et al. 2021

A more detailed look at Vision Transformers (ViT)

Key idea: Convert image into sequence of patches. Can then benefit from Transformer architecture and self-attention, which jointly attends over all patches

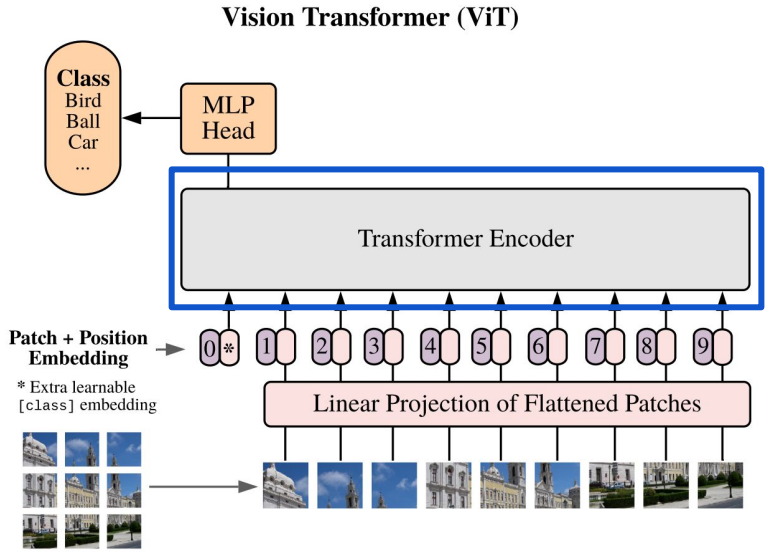


Converted image sequence ready for input to Transformer encoder (analogous to text sequence)

Dosovitsky et al. 2021

A more detailed look at Vision Transformers (ViT)

Key idea: Convert image into sequence of patches. Can then benefit from Transformer architecture and self-attention, which jointly attends over all patches

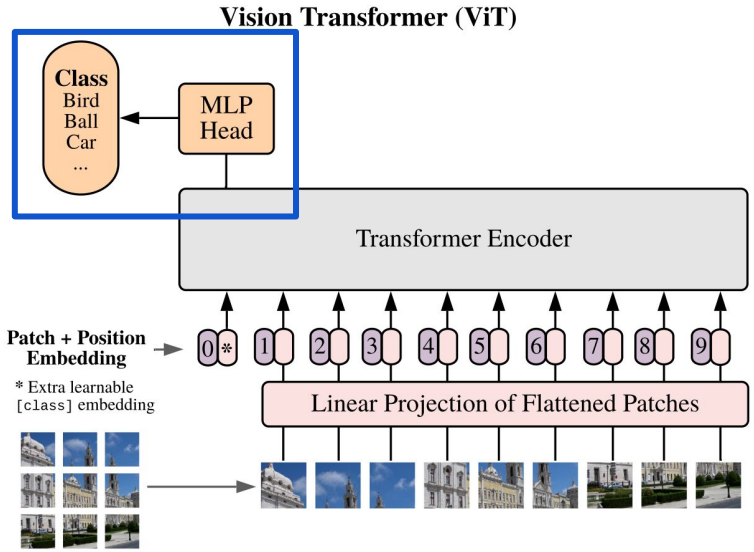


Transformer neural network layers

Dosovitsky et al. 2021

A more detailed look at Vision Transformers (ViT)

Key idea: Convert image into sequence of patches. Can then benefit from Transformer architecture and self-attention, which jointly attends over all patches



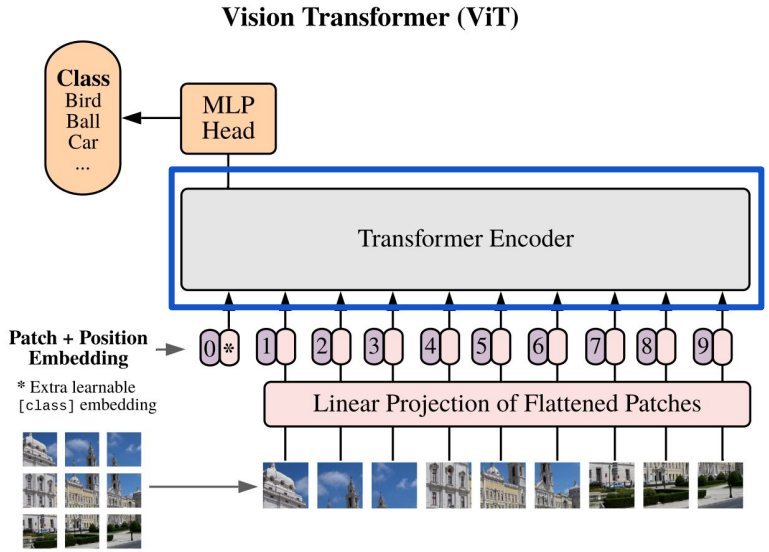
Output of the Transformer encoder can be attached to additional neural network layers to perform a prediction task and receive training signal.

MLP = multilayer perceptron, i.e., a fully connected neural network with 1 or 2 layers.

Dosovitsky et al. 2021

A more detailed look at Vision Transformers (ViT)

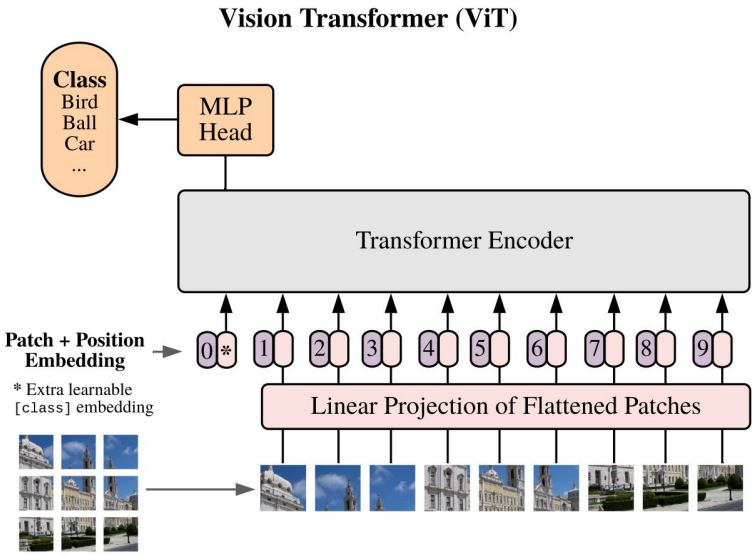
Key idea: Convert image into sequence of patches. Can then benefit from Transformer architecture and self-attention, which jointly attends over all patches



Let's look more carefully at the Transformer architecture…

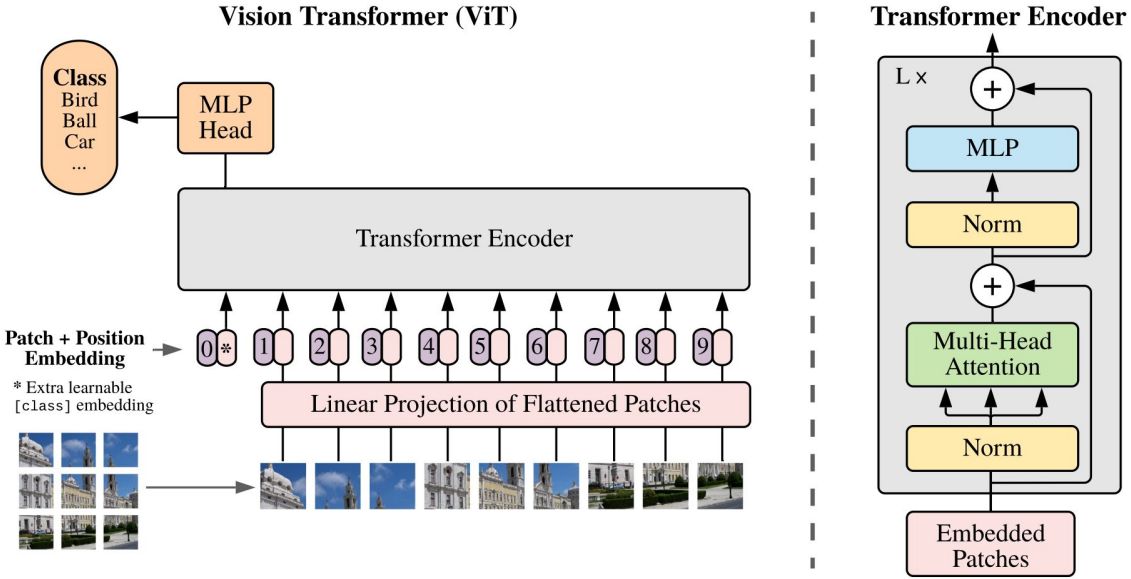
Dosovitsky et al. 2021

Transformer Encoder



Dosovitsky et al. 2021

Transformer Encoder

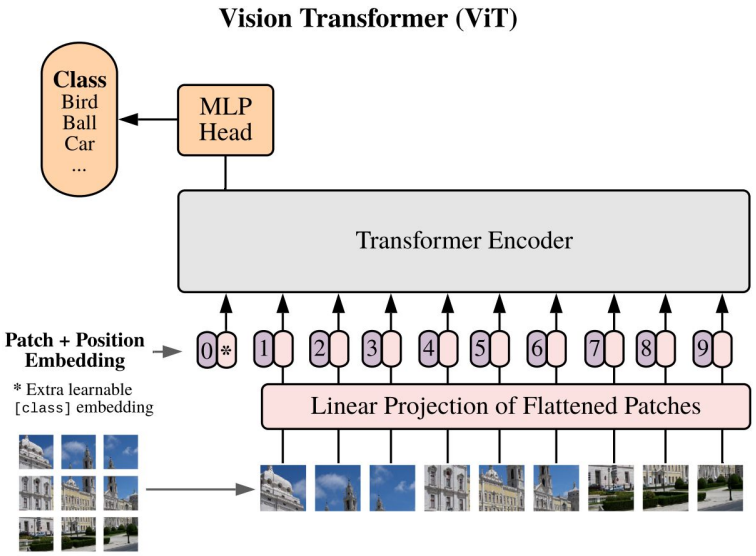


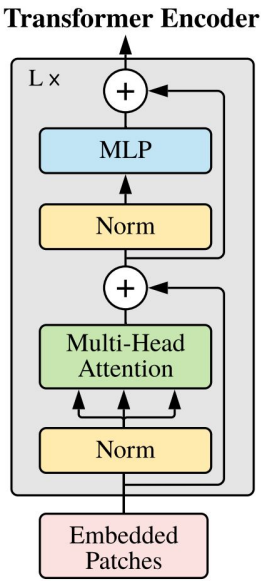
Dosovitsky et al. 2021

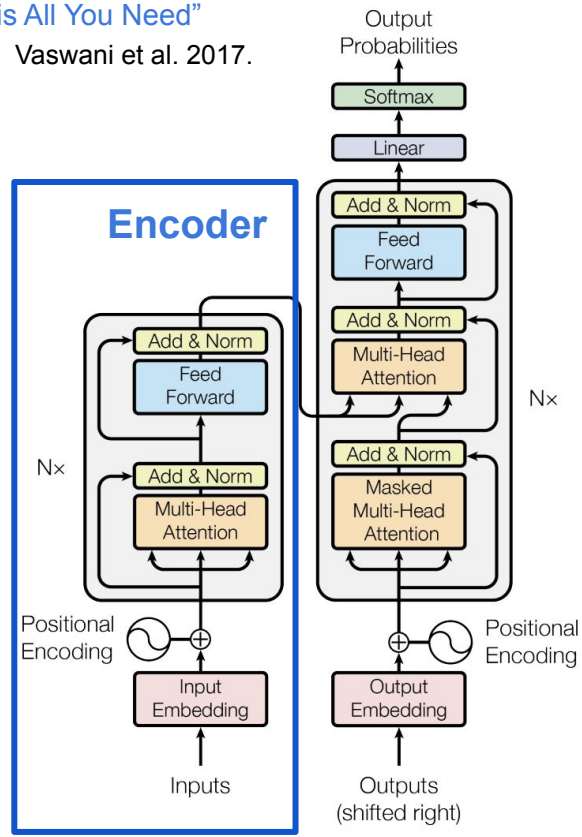
Transformer Encoder

Essentially the same as the original Transformer Encoder for text sequences from "Attention is All You Need"

Vaswani et al. 2017.

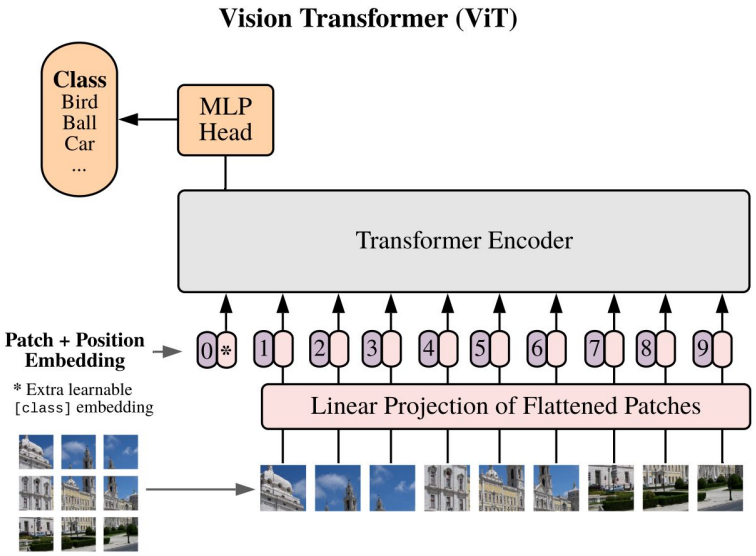


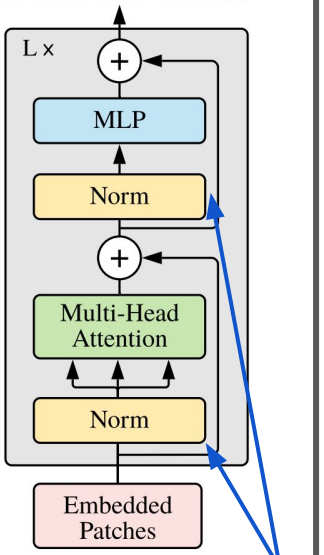




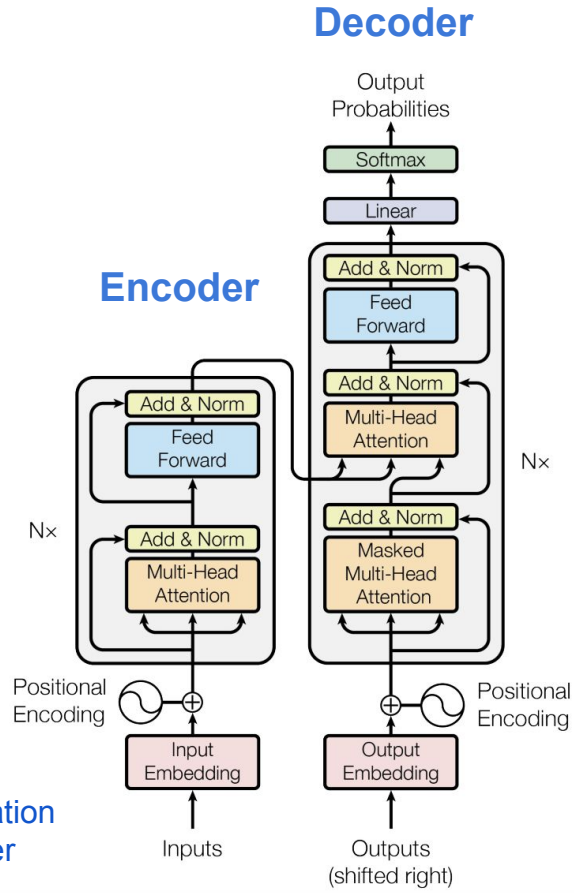
Dosovitsky et al. 2021

Transformer Encoder





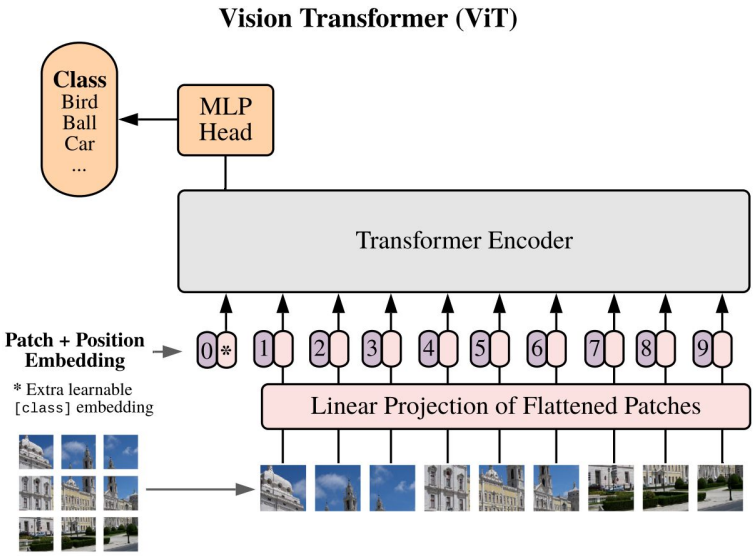
Minor difference: normalization now occurs before, not after attention and MLP layers

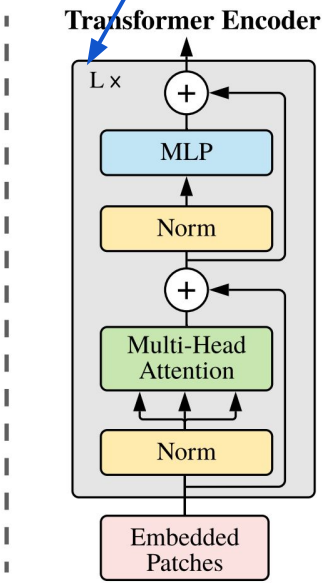


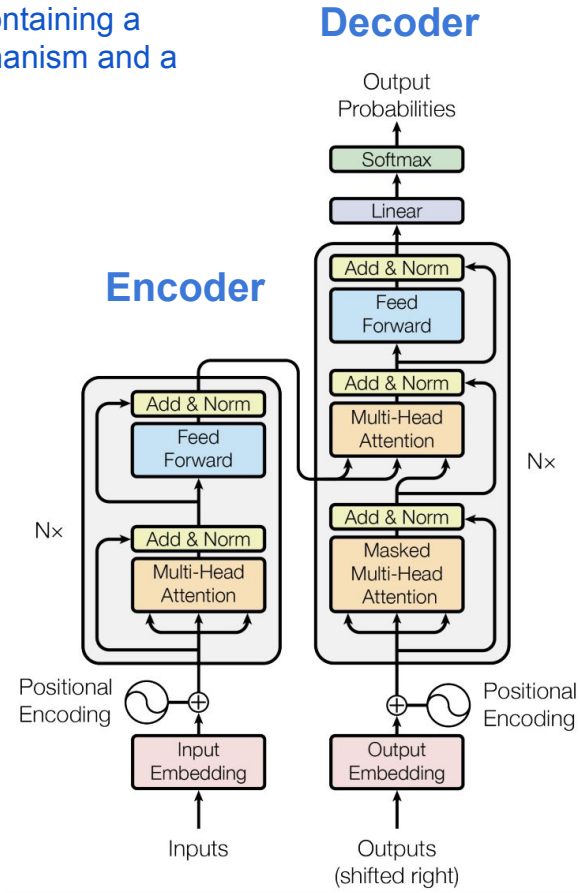
Dosovitsky et al. 2021

Transformer Encoder

The Encoder consists of L encoder blocks (e.g. L=12 or 24), each containing a multi-head attention mechanism and a 2-layer MLP



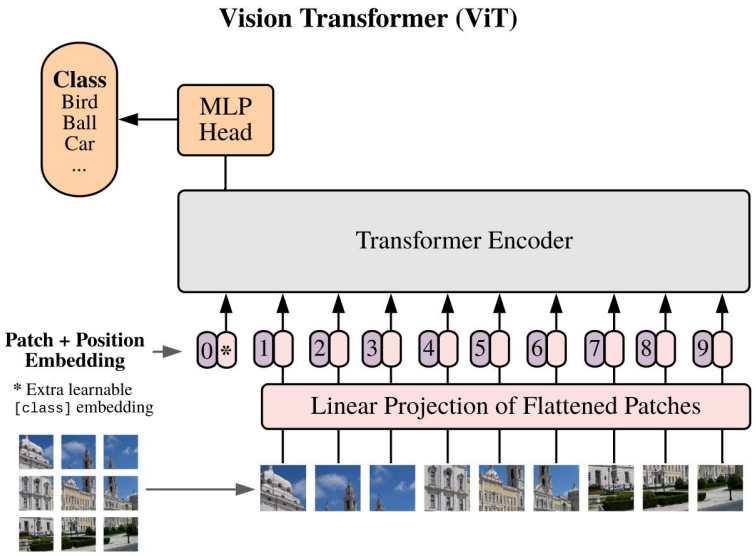


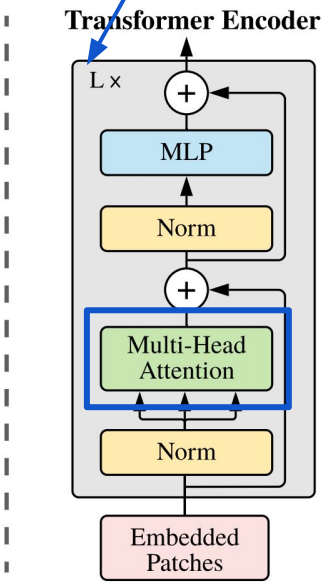


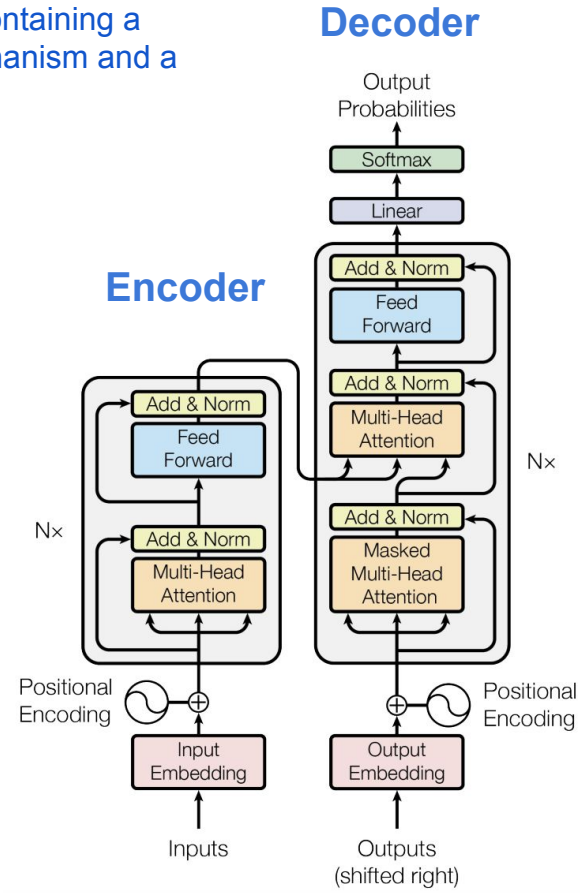
Dosovitsky et al. 2021

Transformer Encoder

The Encoder consists of L encoder blocks (e.g. L=12 or 24), each containing a multi-head attention mechanism and a 2-layer MLP







Dosovitsky et al. 2021

Transformer “attention” mechanism

Consider first attention between a sequence x (of length num_x), and a sequence y (of length num_y):

$$a_j = \text{softmax}\left(\frac{Q_j(x)K_j(y)^T}{\sqrt{d_c}}\right)V_j(y)$$

Vaswani et al. Attention is All You Need, 2017.
Devlin et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, 2018.

Transformer “attention” mechanism

Consider first attention between a sequence x (of length num_x), and a sequence y (of length num_y):

“Query” embedding: [num_x, d_c] where
d_c is embedding dimension

$$a_j = \text{softmax}\left(\frac{Q_j(x)K_j(y)^T}{\sqrt{d_c}}\right)V_j(y)$$

Vaswani et al. Attention is All You Need, 2017.
Devlin et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, 2018.

Transformer “attention” mechanism

Consider first attention between a sequence x (of length num_x), and a sequence y (of length num_y):

“Query” embedding: [num_x, d_c] where d_c is embedding dimension

“Key” embedding: [num_y, d_c]

$$a_j = \text{softmax}\left(\frac{Q_j(x)K_j(y)^T}{\sqrt{d_c}}\right)V_j(y)$$

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

Devlin et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, 2018.

Transformer “attention” mechanism

Consider first attention between a sequence x (of length num_x), and a sequence y (of length num_y):

“Query” embedding: [num_x, d_c] where d_c is embedding dimension

“Key” embedding: [num_y, d_c]

$$a_j = \text{softmax}\left(\frac{Q_j(x)K_j(y)^T}{\sqrt{d_c}}\right)V_j(y)$$

Embedding dimension

Vaswani et al. Attention is All You Need, 2017.
Devlin et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, 2018.

Transformer “attention” mechanism

Consider first attention between a sequence x (of length num_x), and a sequence y (of length num_y):

“Query” embedding: [num_x, d_c] where d_c is embedding dimension

“Key” embedding: [num_y, d_c]

$$a_j = \text{softmax}\left(\frac{Q_j(x)K_j(y)^T}{\sqrt{d_c}}\right)V_j(y)$$

Embedding dimension

Attention weights

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

Devlin et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, 2018.

Transformer “attention” mechanism

Consider first attention between a sequence x (of length num_x), and a sequence y (of length num_y):

“Query” embedding: [num_x, d_c] where d_c is embedding dimension

“Key” embedding: [num_y, d_c]

$$a_j = \text{softmax}\left(\frac{Q_j(x)K_j(y)^T}{\sqrt{d_c}}\right)V_j(y)$$

Embedding dimension

“Value” embedding: [num_y, d_c]

Attention weights

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

Devlin et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, 2018.

Transformer “attention” mechanism

Consider first attention between a sequence x (of length num_x), and a sequence y (of length num_y):

“Query” embedding: [num_x, d_c] where d_c is embedding dimension

“Key” embedding: [num_y, d_c]

$$a_j = \text{softmax}\left(\frac{Q_j(x)K_j(y)^T}{\sqrt{d_c}}\right)V_j(y)$$

Attention-weighted outputs of jth attention head: [num_x, d_c]

Embedding dimension

“Value” embedding: [num_y, d_c]

Attention weights

Vaswani et al. Attention is All You Need, 2017.
Devlin et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, 2018.

Transformer “attention” mechanism

Consider first attention between a sequence x (of length num_x), and a sequence y (of length num_y):

“Query” embedding: [num_x, d_c] where d_c is embedding dimension

“Key” embedding: [num_y, d_c]

$$a_j = \text{softmax}\left(\frac{Q_j(x)K_j(y)^T}{\sqrt{d_c}}\right)V_j(y)$$

Attention-weighted outputs of jth attention head: [num_x, d_c]

Embedding dimension

“Value” embedding: [num_y, d_c]

Attention weights

If using multiple attention heads, combine outputs with another matrix multiply

Vaswani et al. Attention is All You Need, 2017.
Devlin et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, 2018.

Transformer "self-attention" mechanism

"Self-attention" is just this attention mechanism with x = y!

Consider first attention between a sequence x (of length num_x), and a sequence y (of length num_y):

"Query" embedding: [num_x, d_c] where d_c is embedding dimension

"Key" embedding: [num_y, d_c]

$$a_j = \text{softmax}\left(\frac{Q_j(x)K_j(y)^T}{\sqrt{d_c}}\right)V_j(y)$$

Attention-weighted outputs of jth attention head: [num_x, d_c]

Embedding dimension

"Value" embedding: [num_y, d_c]

Attention weights

If using multiple attention heads, combine outputs with another matrix multiply

Vaswani et al. Attention is All You Need, 2017.
Devlin et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, 2018.

Transformer "self-attention" mechanism

"Self-attention" is just this attention mechanism with x = y!

Consider first attention between a sequence x (of length num_x), and a sequence y (of length num_y):

"Query" embedding: [num_x, d_c] where d_c is embedding dimension

"Key" embedding: [num_y, d_c]

Recommended reading: Check out this well-known blog post for a great (and illustrated) in-depth explanation of the self-attention mechanism and Transformers: https://jalammar.github.io/illustrated-transformer/

$$a_j =$$

Attention-weighted outputs of jth attention head: [num_x, d_c]

Embedding dimension

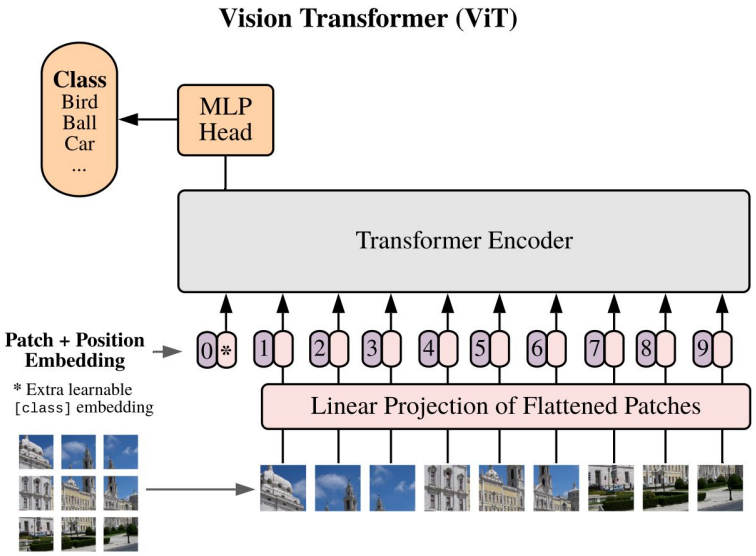
"Value" embedding: [num_y, d_c]

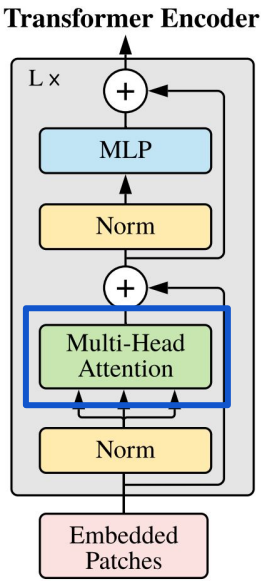
Attention weights

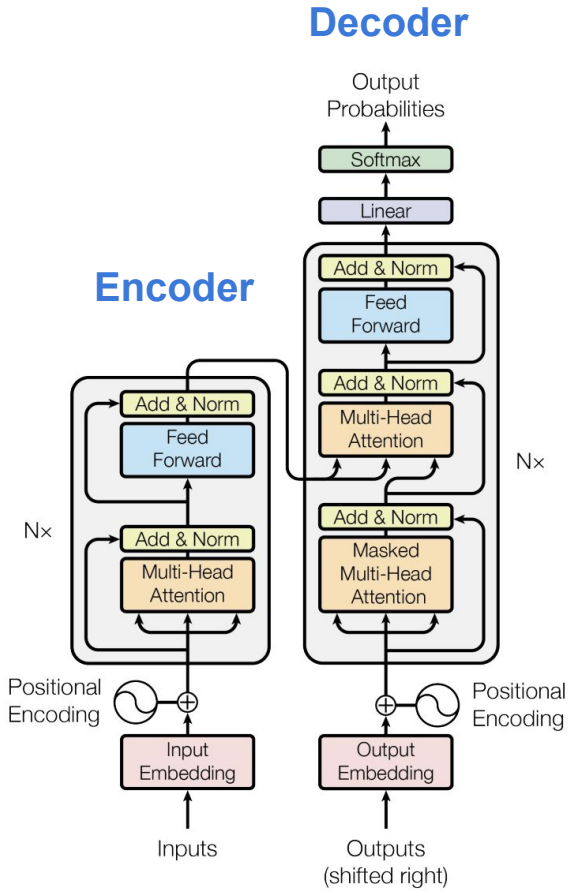
If using multiple attention heads, combine outputs with another matrix multiply

Vaswani et al. Attention is All You Need, 2017.
Devlin et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, 2018.

Back to the whole picture







Dosovitsky et al. 2021

Original ViT model variants

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

Dosovitsky et al. 2021

Original ViT model variants

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

Will often see models referred to such as “ViT-L/16”, which stands for ViT-Large with 16x16 input patches

Dosovitsky et al. 2021

Do vision transformers work well?

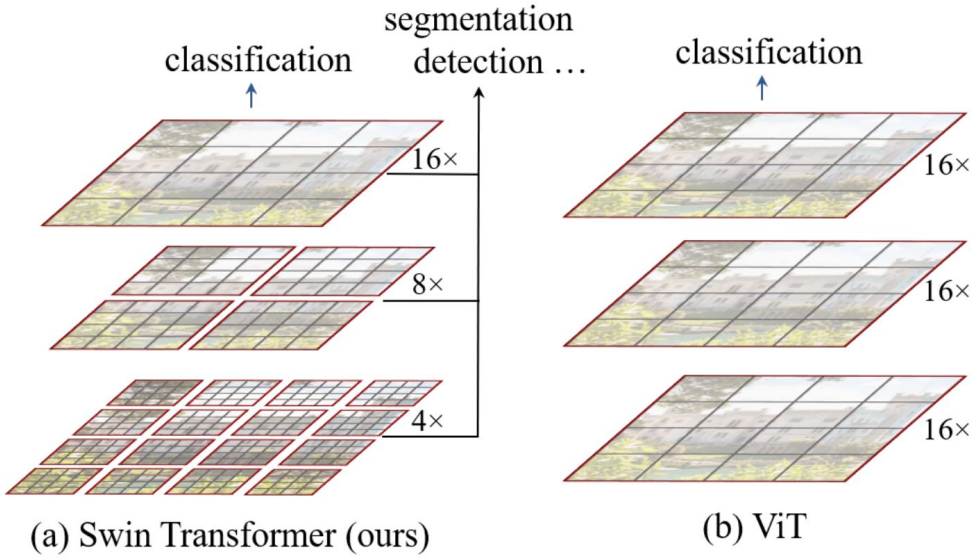
- ViT first Transformer-based vision model to achieve comparable or superior results to state-of-the-art CNNs
- Transformer architecture has less inductive biases than CNNs (i.e., assumes less about the spatial structure than convolutional filter design does)
    - Consequence: Transformers work well when trained on very large amounts of data, less so when there are smaller / medium amounts of data (in this case, leveraging CNN's assumptions about data structure can be helpful)
- Weakness: ViT has quadratic computational complexity of self-attention, which can cause challenges e.g. scaling to high-resolution images, and can be more memory intensive

Some areas of ongoing research

- Improving computational efficiency
- Handling high-resolution and large-scale images, and multi-scale Transformers
- Architectures that model more complex vision settings such as video, detection, segmentation, etc.
- Improving positional encoding mechanisms

Swin Transformer: Hierarchical Vision Transformer using Shifted Windows

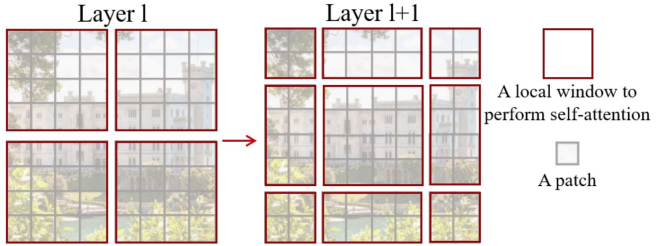
- Constructs a hierarchical feature representation by starting from small-size patches and gradually merging in deeper layers
  -> suitable for denser vision tasks
- Maintains low computational complexity by computing self-attention only within non-overlapping windows that partition an image

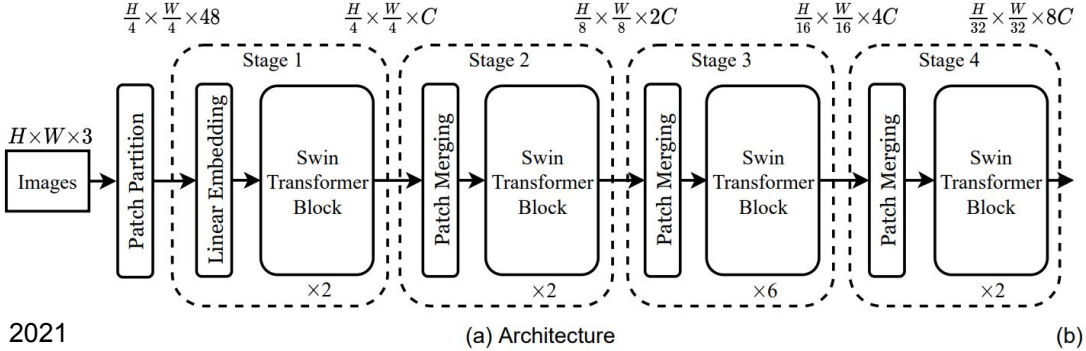


Liu et al. 2021

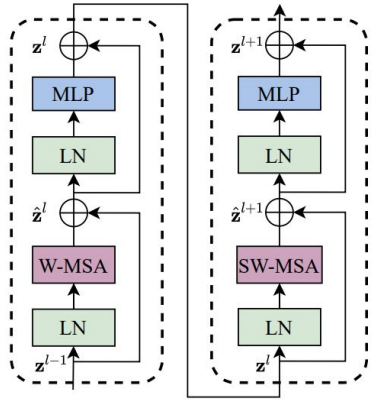
Swin Transformer: Hierarchical Vision Transformer using Shifted Windows

An important element: shifted windows for local self-attention at different layers, to provide connections across local regions





(a) Architecture

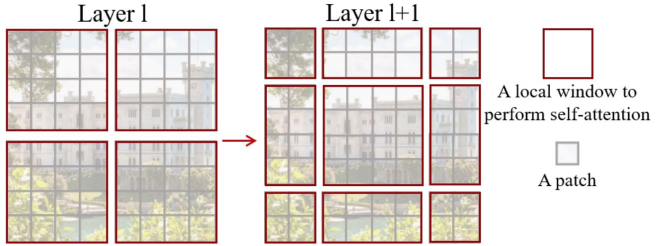


(b) Two Successive Swin Transformer Blocks

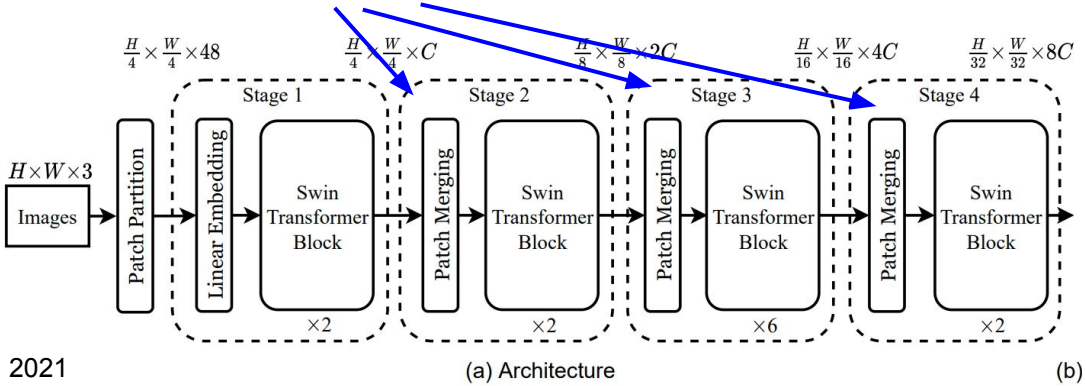
Liu et al. 2021

Swin Transformer: Hierarchical Vision Transformer using Shifted Windows

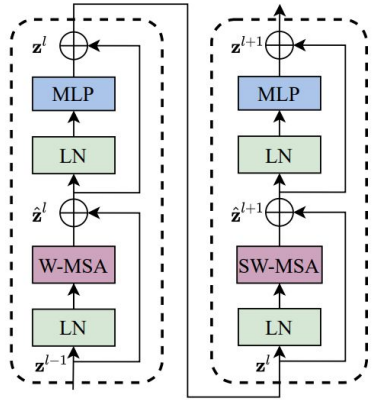
An important element: shifted windows for local self-attention at different layers, to provide connections across local regions



Increasing patch size through merging (earlier layers allow more localized features that can be useful for dense prediction tasks)



(a) Architecture

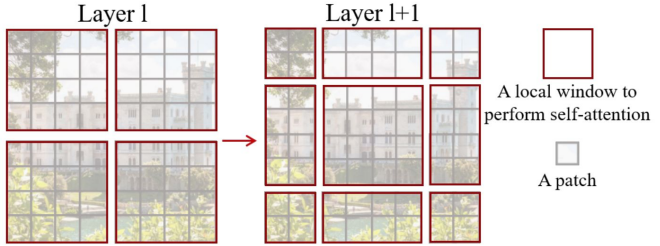


(b) Two Successive Swin Transformer Blocks

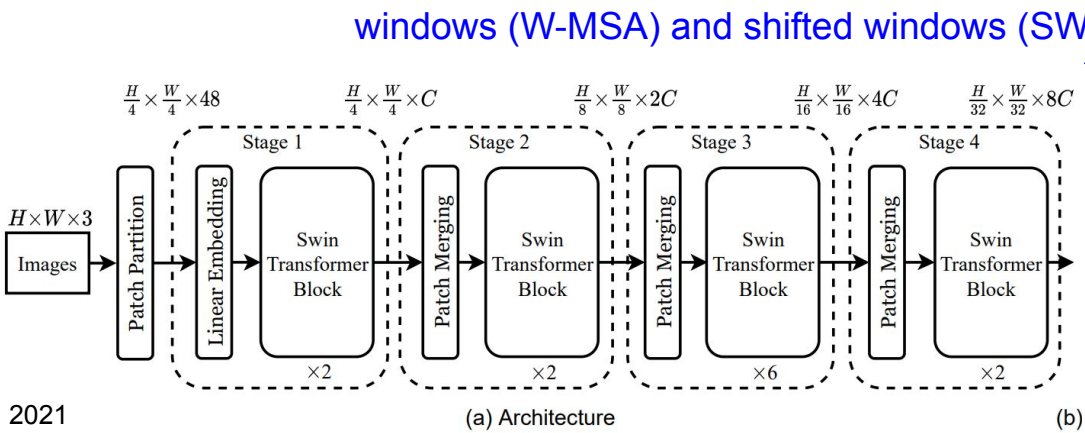
Liu et al. 2021

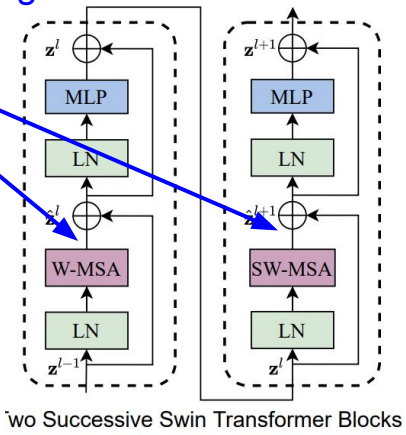
Swin Transformer: Hierarchical Vision Transformer using Shifted Windows

An important element: shifted windows for local self-attention at different layers, to provide connections across local regions



Consecutive multi-head self attention modules with regular windows (W-MSA) and shifted windows (SW-MSA)





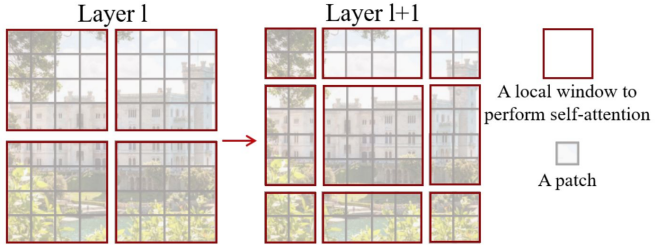
Liu et al. 2021

(a) Architecture

(b) Two Successive Swin Transformer Blocks

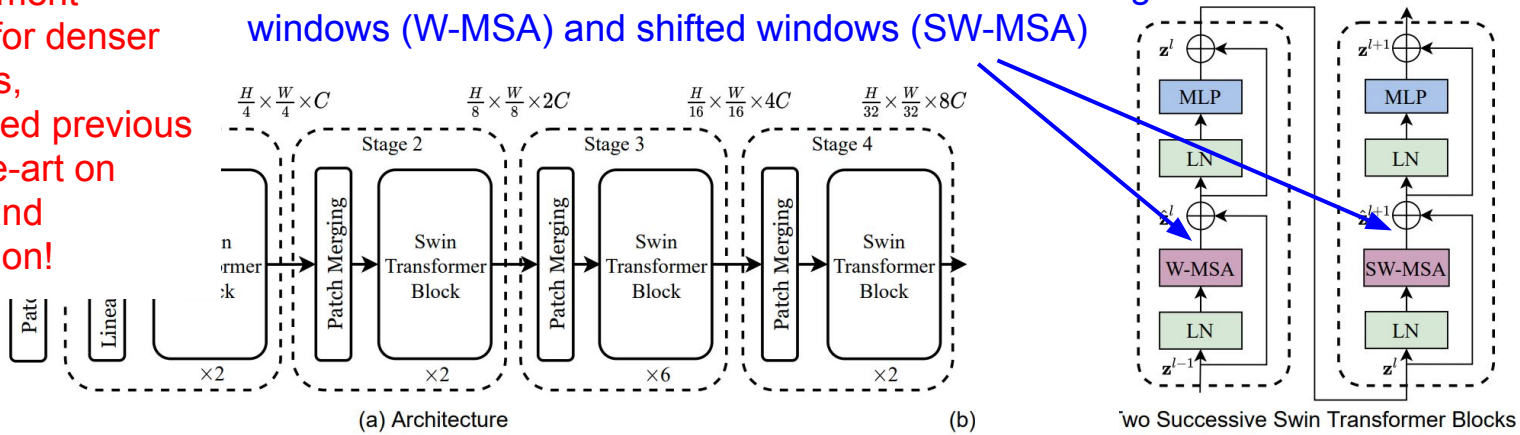
Swin Transformer: Hierarchical Vision Transformer using Shifted Windows

An important element: shifted windows for local self-attention at different layers, to provide connections across local regions



As replacement backbone for denser vision tasks, outperformed previous state-of-the-art on detection and segmentation!

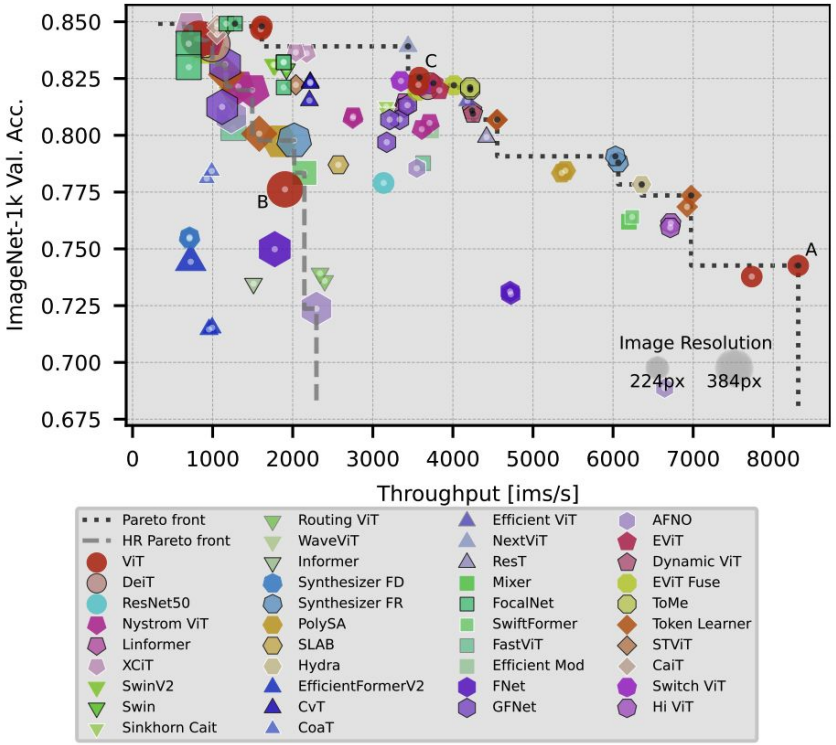
Consecutive multi-head self attention modules with regular windows (W-MSA) and shifted windows (SW-MSA)



Liu et al. 2021

However vanilla ViT is still widely used

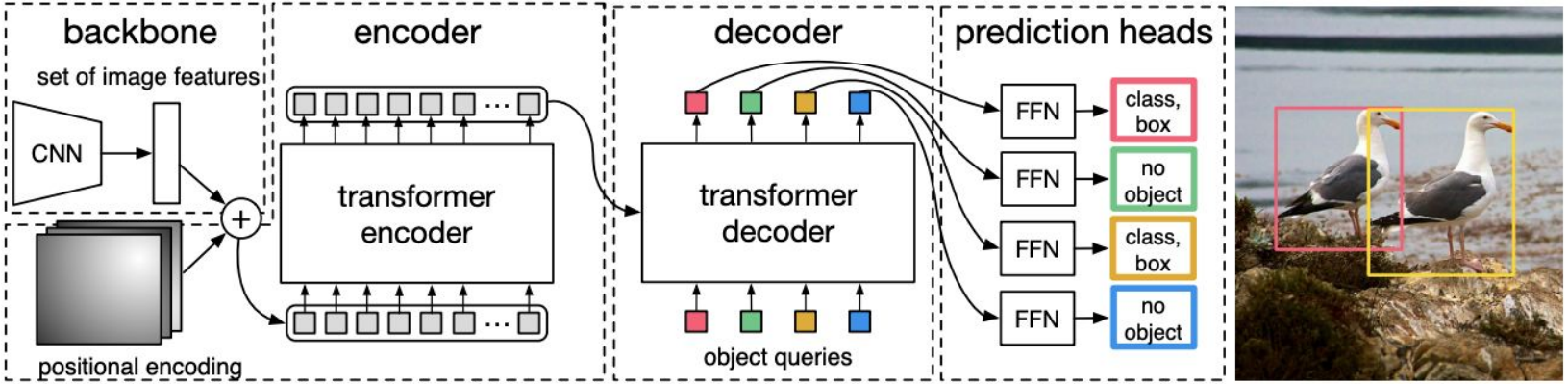
Vanilla ViT is simple, directly analogous to Transformers for text, and benchmarks show it to be still very strong on the Pareto optimal curve of ViT variants when considering various metrics of accuracy, speed, memory, etc.



Nauen et al. Which Transformer to Favor: A Comparative Analysis of Efficiency in Vision Transformers, 2024

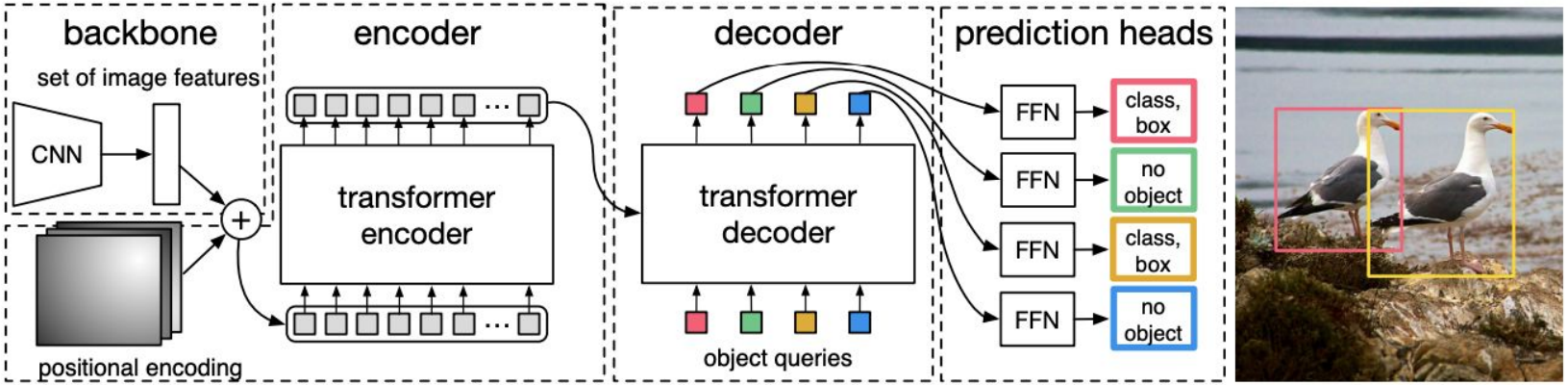
DETR: Example of Transformer-based architecture designed for more complex vision tasks

- Uses a Transformer encoder-decoder model to perform detection and segmentation
- Trained directly end-to-end to predict all objects at once, with a set loss function that performs bipartite matching
- Allows avoiding previous hand-designed components of object detection models, like spatial anchors and non-maximal suppression!



Carion et al. 2020

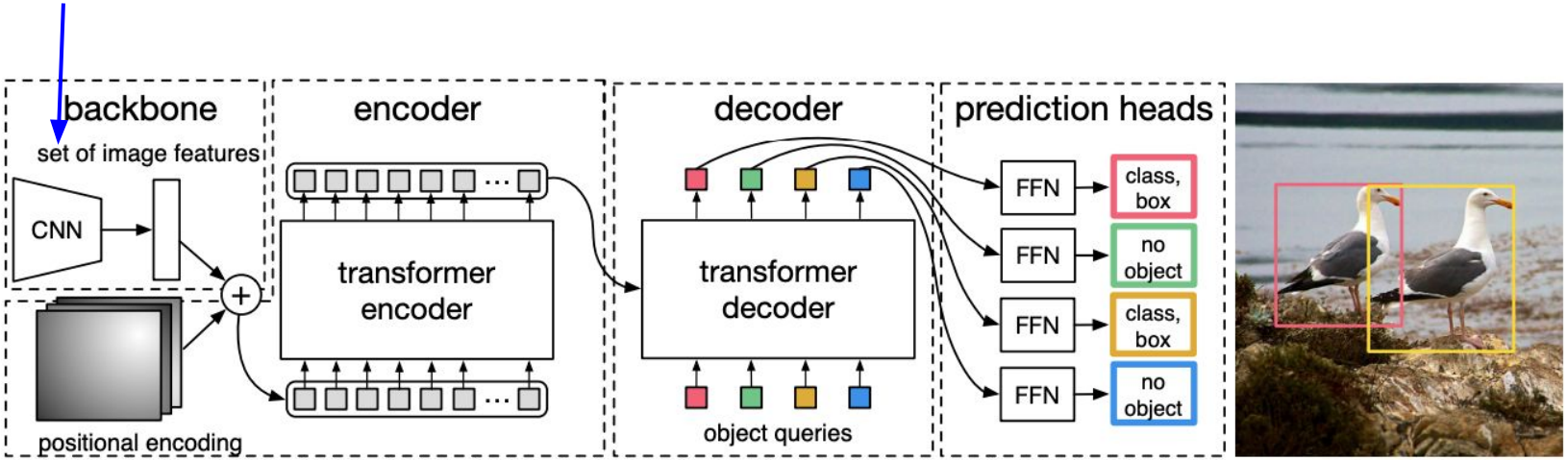
DETR: Example of Transformer-based architecture designed for more complex vision tasks



Carion et al. 2020

DETR: Example of Transformer-based architecture designed for more complex vision tasks

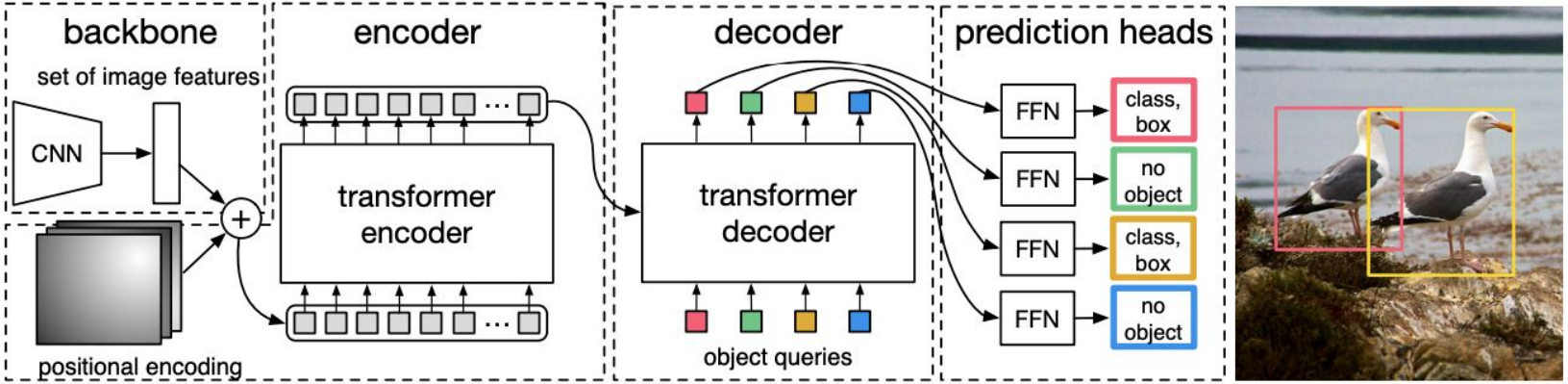
First extract image features with a CNN



Carion et al. 2020

DETR: Example of Transformer-based architecture designed for more complex vision tasks

Transformer encoder-decoder part of the model is used for predicting a set of objects from the image features



Carion et al. 2020

DETR: Example of Transformer-based architecture designed for more complex vision tasks

Transformer encoder-decoder part of the model is used for predicting a set of objects from the image features

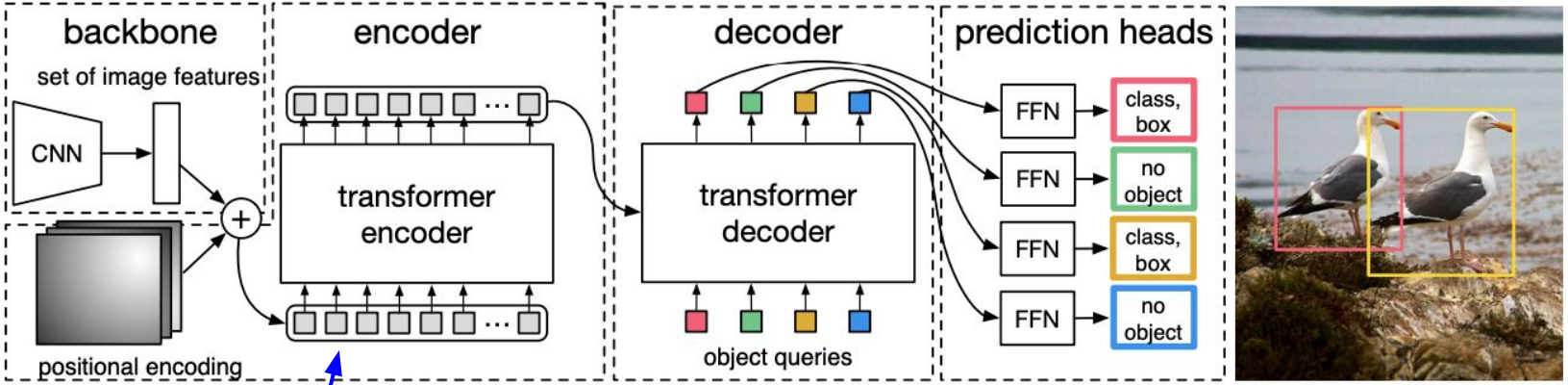
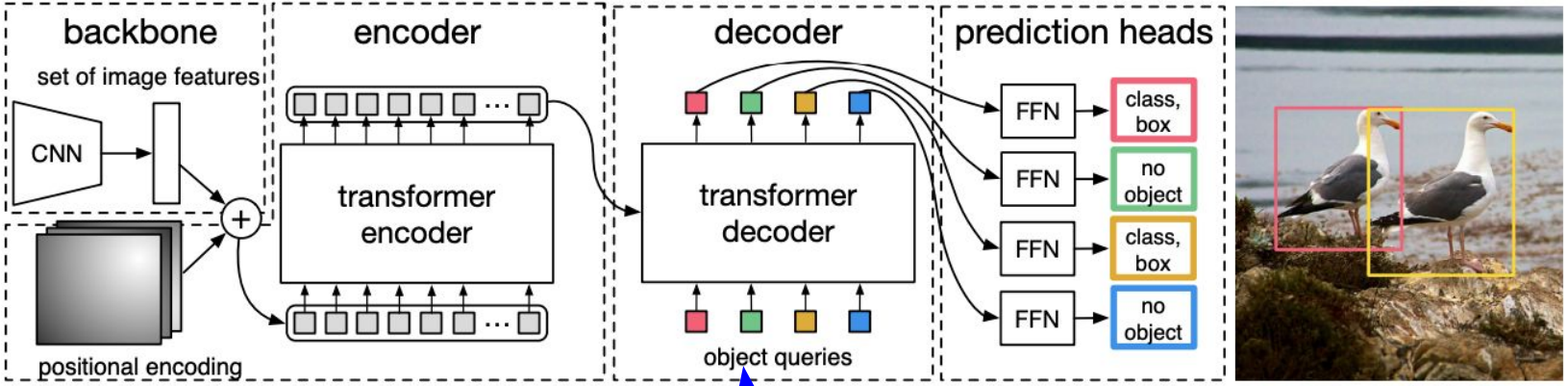


Image features are converted to sequence and input to encoder

Carion et al. 2020

DETR: Example of Transformer-based architecture designed for more complex vision tasks

Transformer encoder-decoder part of the model is used for predicting a set of objects from the image features



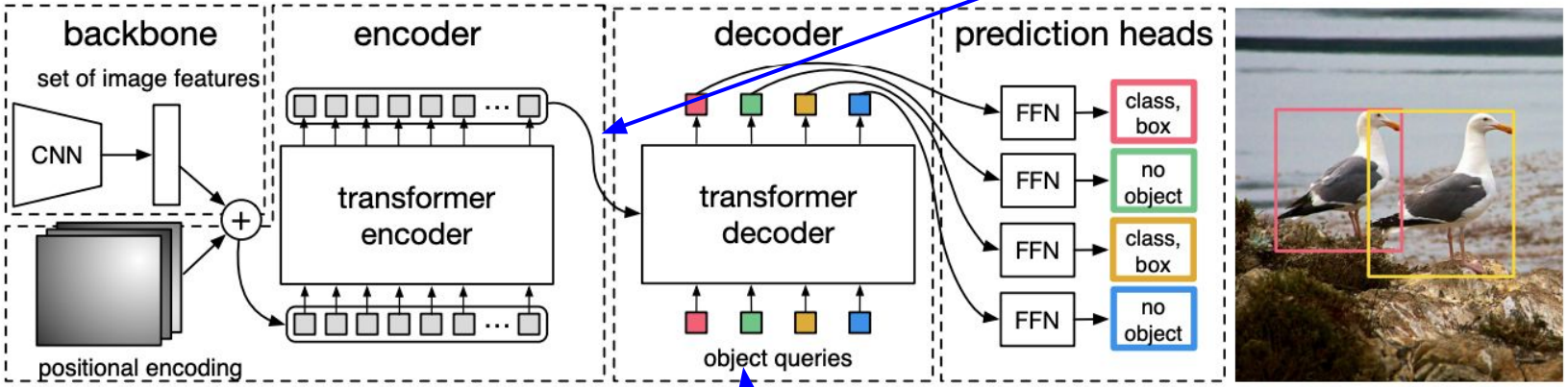
Decoder produces outputs that are fed into simple feedforward network for predicting bounding box locations and classes. Input to decoder is a set of learned positional encodings (can be understood as “object queries”) => each one will correspond to a predicted box at output. Use N object queries > total expected number of boxes in the image, class output can also be “no object” to predict variable # of objects.

Carion et al. 2020

DETR: Example of Transformer-based architecture designed for more complex vision tasks

Transformer encoder-decoder part of the model is used for predicting a set of objects from the image features

Encoder-decoder attention (remember how Transformer attention mechanism can be defined between any two sequences)



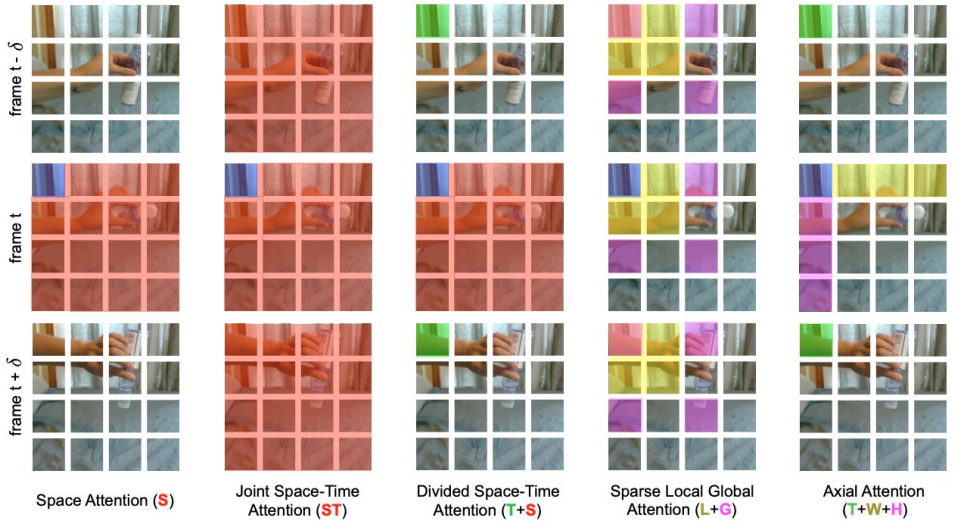
Decoder produces outputs that are fed into simple feedforward network for predicting bounding box locations and classes. Input to decoder is a set of learned positional encodings (can be understood as “object queries”) => each one will correspond to a predicted box at output. Use N object queries > total expected number of boxes in the image, class output can also be “no object” to predict variable # of objects.

Carion et al. 2020

Transformers for video: attention across space-time

Example: TimeSformer model (Bertasius 2021)

Extension of what we have already seen, but can compute attention of query position (blue patch) over sequences corresponding to different neighborhoods (other colored patches)

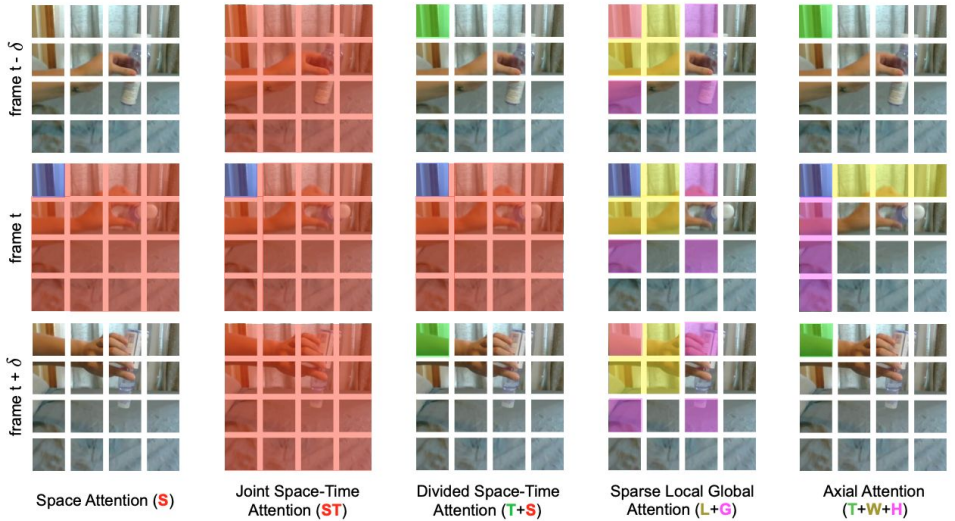


Bertasius et al. 2021

Transformers for video: attention across space-time

Example: TimeSformer model (Bertasius 2021)

Extension of what we have already seen, but can compute attention of query position (blue patch) over sequences corresponding to different neighborhoods (other colored patches)



Standard image-level attention

Attention over different spatiotemporal neighborhoods

Bertasius et al. 2021

Next time

- Vision representation learners