Fundamental of Visual Transformer

Lecture 8

Content of this course

- Difference between CNN and ViT
  - What is the inductive bias?
  - Which one is better?

- Vision transformer
  - Case study: Image captioning
  - Basic algorithm, analysis, and architectures

A brief overview of CNNs

- CNNs: most remarkable architecture in deep learning filed.

A bit of history:

ImageNet Classification with Deep Convolutional Neural Networks

[Krizhevsky, Sutskever, Hinton, 2012]

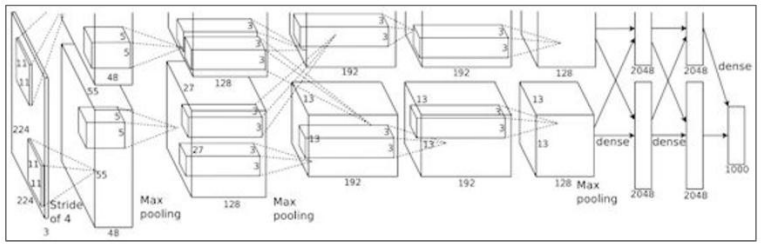


Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

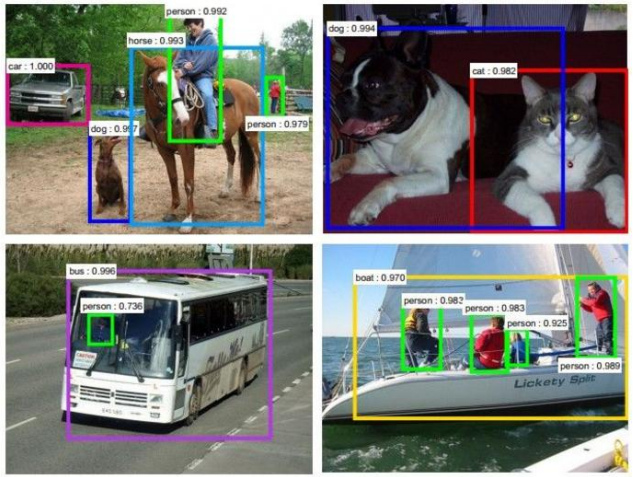
"AlexNet"

A brief overview of CNNs

- CNN: we use it to many applications

Fast-forward to today: ConvNets are everywhere

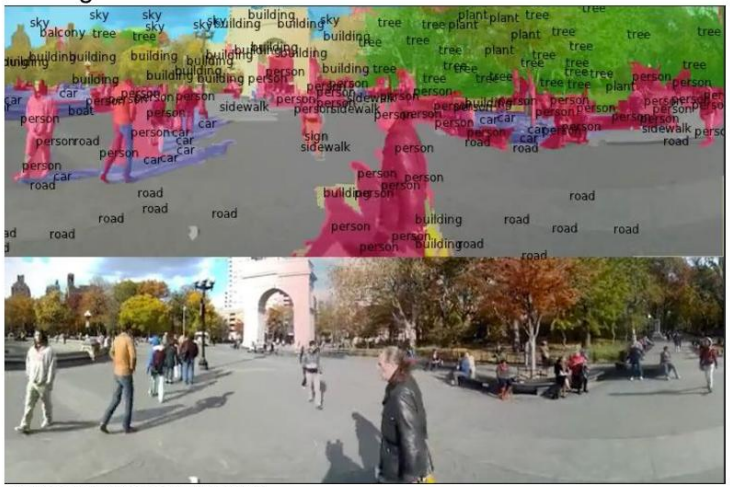
Detection



Figures copyright Shaoqing Ren, Kaiming He, Ross Girschick, Jian Sun, 2015. Reproduced with permission.

[Faster R-CNN: Ren, He, Girshick, Sun 2015]

Segmentation



Figures copyright Clement Farabet, 2012. Reproduced with permission.

[Farabet et al., 2012]

A brief overview of CNNs

- CNN: we use it to many applications

No errors

A white teddy bear sitting in the grass

Minor errors

A man in a baseball uniform throwing a ball

Somewhat related

A woman is holding a cat in her hand

A man riding a wave on top of a surfboard

A cat sitting on a suitcase on the floor

A woman standing on a beach holding a surfboard

Image Captioning

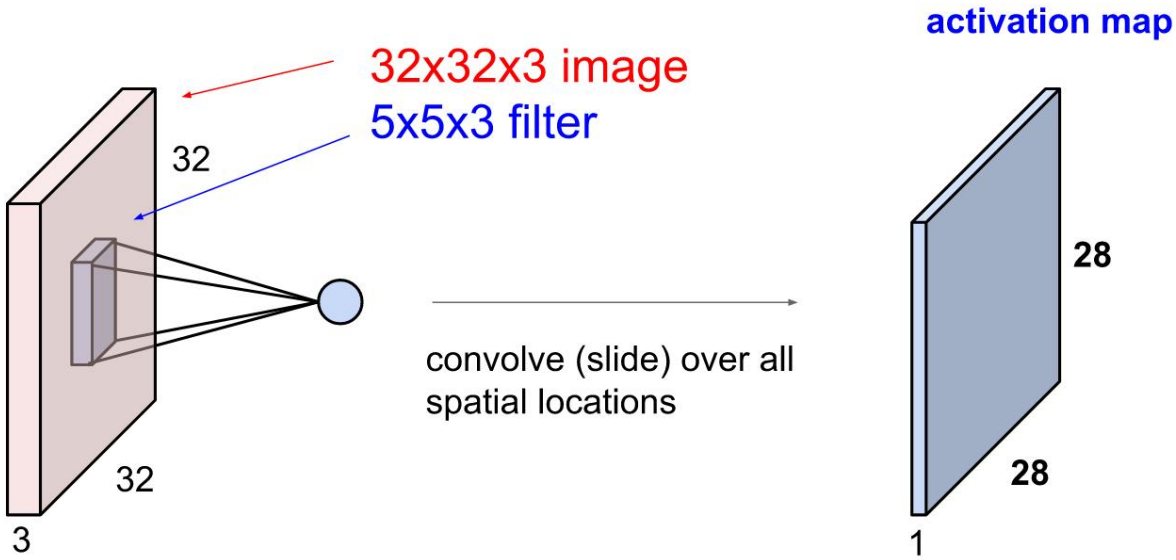
[Vinyals et al., 2015]
[Karpathy and Fei-Fei, 2015]

Captions generated by Justin Johnson using Neuraltalk2

A brief overview of CNNs

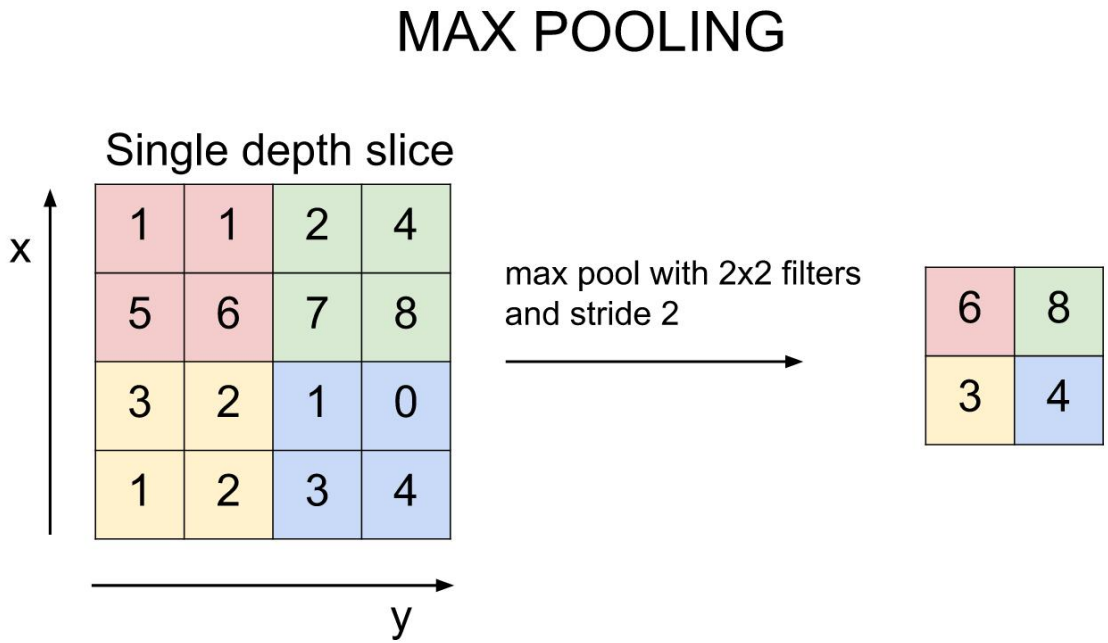
- The heart of CNN is the convolution operation in the spatial dimension.

A closer look at spatial dimensions:



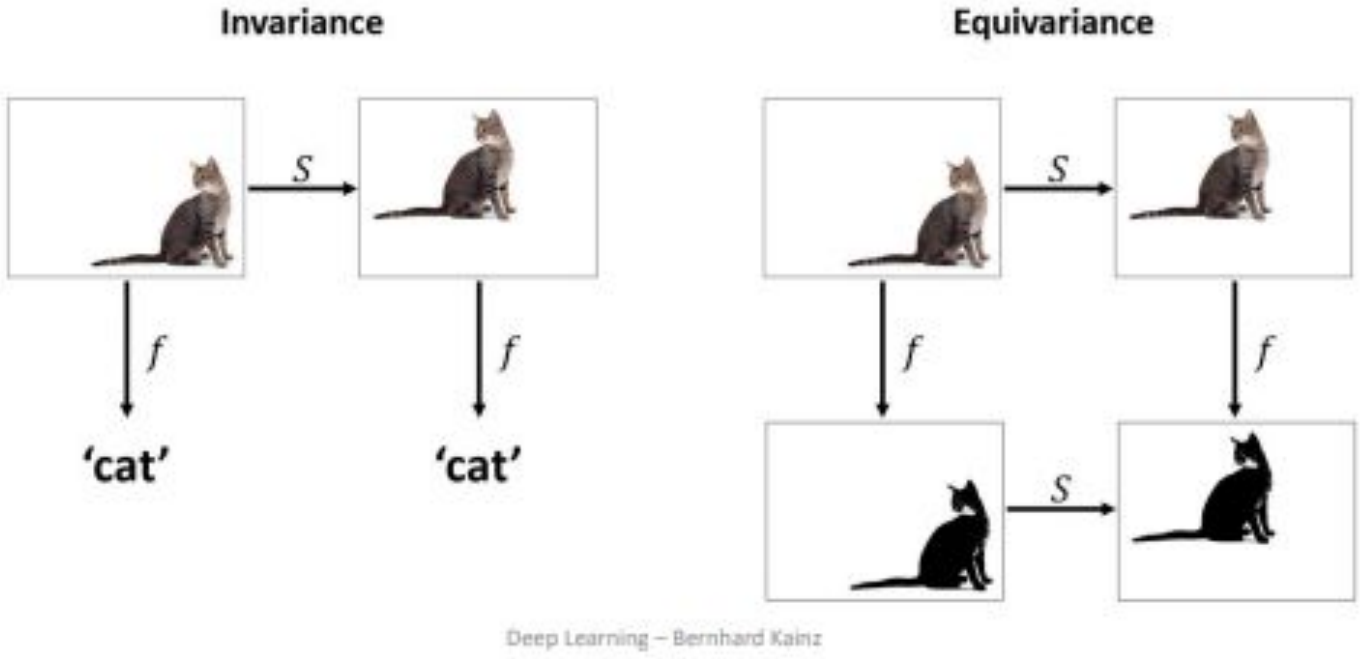
A brief overview of CNNs

- Another important thing is the pooling layer in CNN.



A brief overview of CNNs

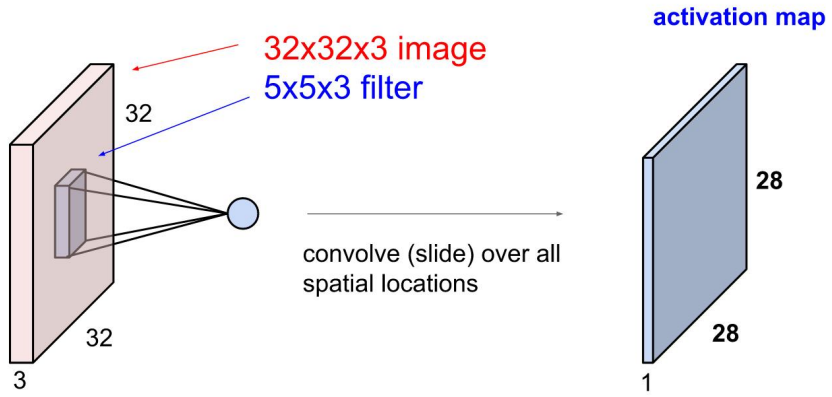
- Two important properties of CNN: Translation equivariance & invariance

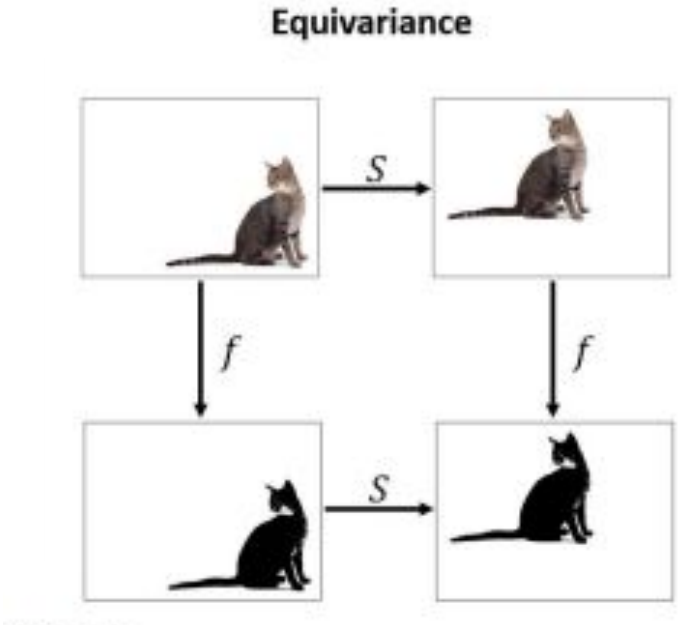


Translation equivariance

- Translation equivariance is achieved by the convolution operation

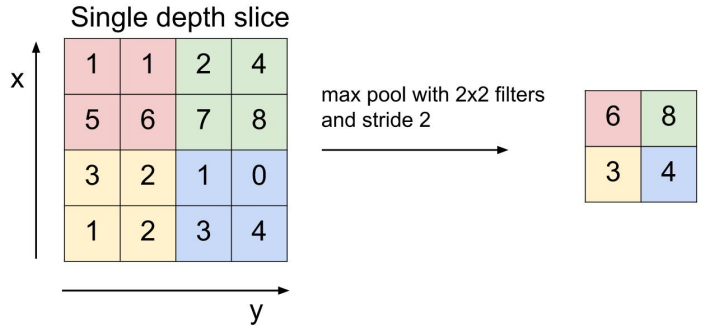
A closer look at spatial dimensions:

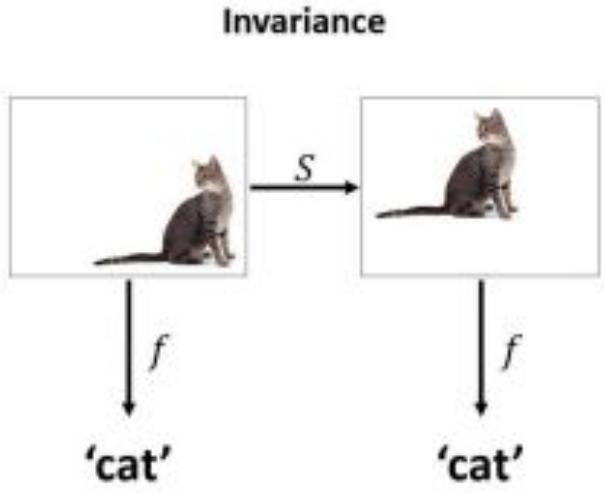




Translation invariance

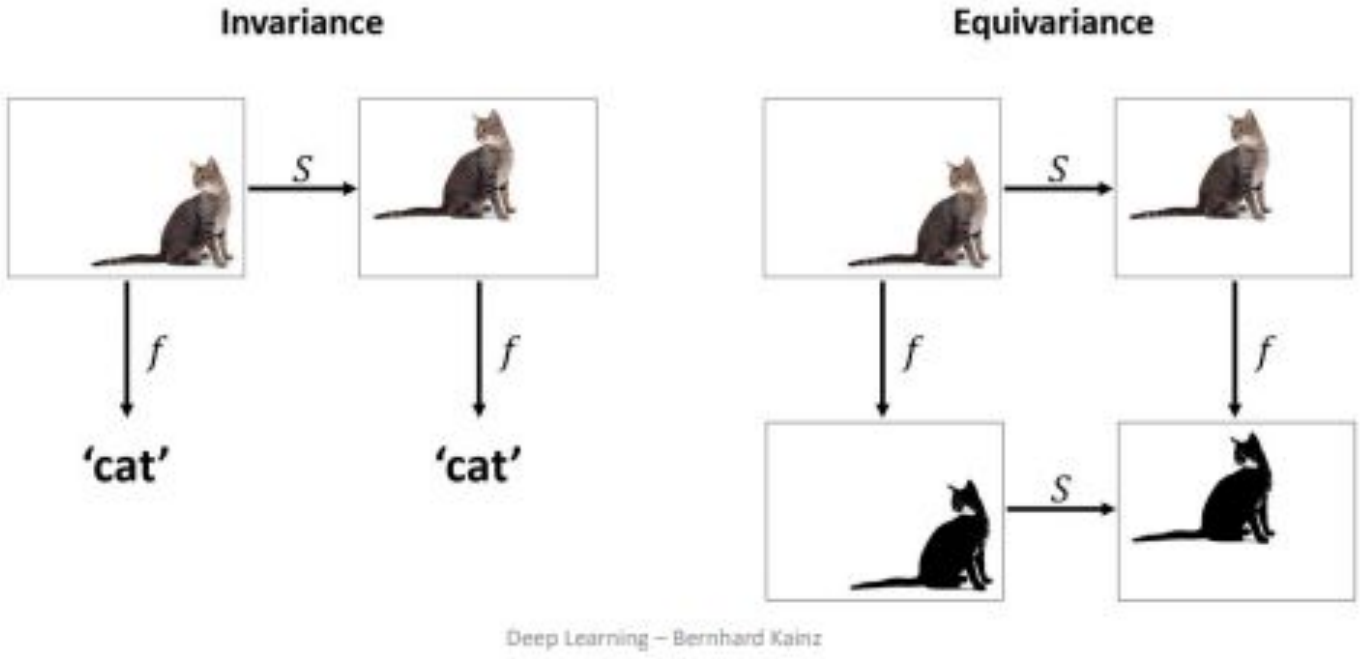
- Translation invariance is achieved by the pooling layer





Inductive bias in CNN

- We refer these translation equivariance and invariance as the inductive bias.



Inductive bias in CNN

- (+) Strong points
  - Effectively predict most unseen data (image)
  - Good sample efficiency

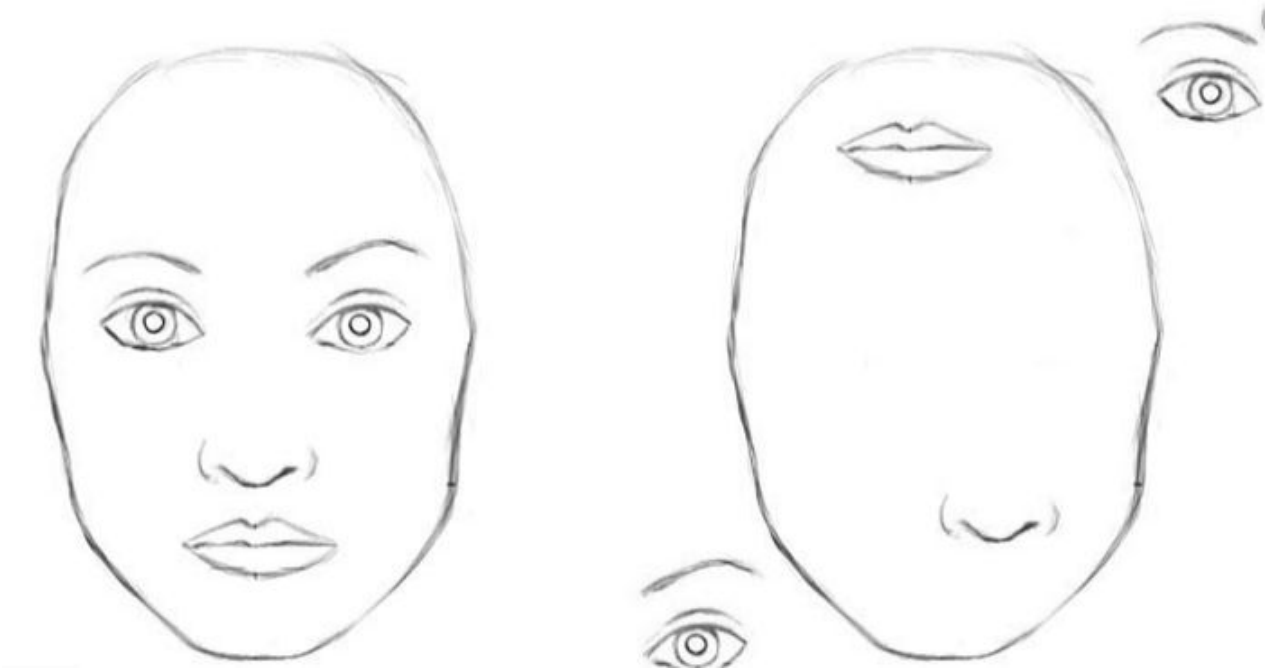
Inductive bias in CNN

- (+) Strong points
  - Effectively predict most unseen data(image) with pre-defined induction
  - Good sample efficiency

- (-) Weak points
  - Poorly predict some data(image) out of the pre-defined induction
  - Loss of features due to pooling layer

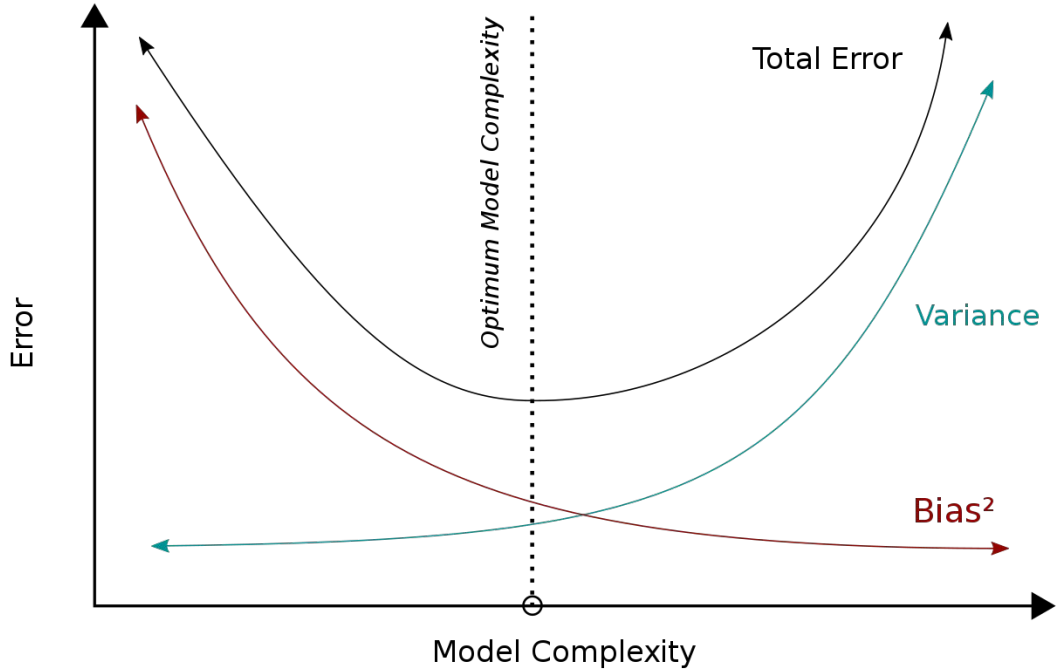
Inductive bias in CNN

- Are they both the same face?



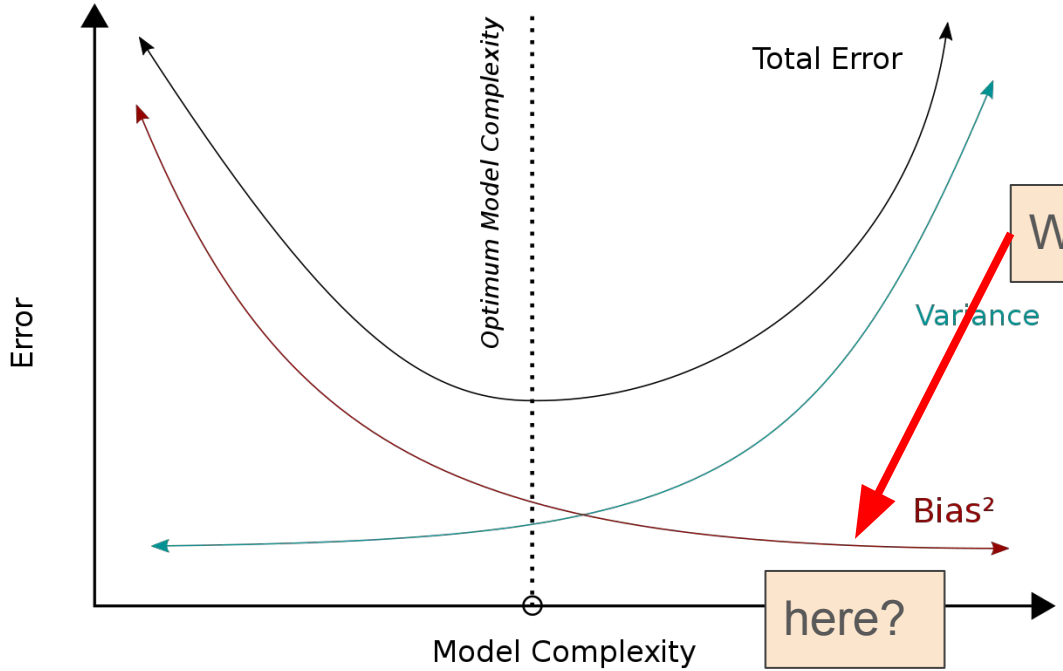
Inductive bias in CNN

- If so, should the bias be larger? (= Should we consider less bias?)



Inductive bias in CNN

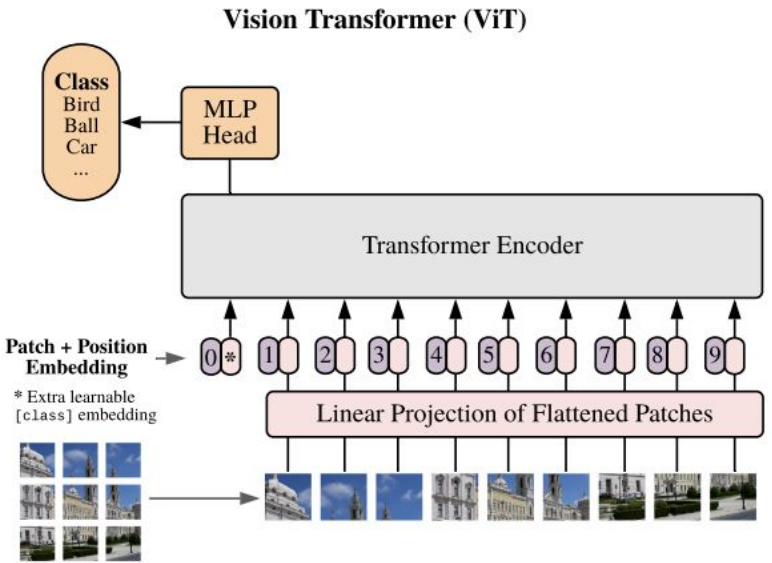
- If so, should the bias be larger? (= Should we consider less bias?)



Where is the CNN on the graph?

Less inductive bias in ViT

- Vision transformer(ViT) has no inductive bias of translation equivariance and invariance.
  - No convolution operation
  - No pooling layer



Less inductive bias in ViT

- So, how can we effectively recognize images without inductive bias?
  - Exploit huge amount of training data (>300M)

JFT-300M

Introduced by Sun et al. in Revisiting Unreasonable Effectiveness of Data in Deep Learning Era

JFT-300M is an internal Google dataset used for training image classification models. Images are labeled using an algorithm that uses complex mixture of raw web signals, connections between web-pages and user feedback. This results in over one billion labels for the 300M images (a single image can have multiple labels). Of the billion image labels, approximately 375M are selected via an algorithm that aims to maximize label precision of selected images.

https://paperswithcode.com/dataset/jft-300m

Less inductive bias in ViT

- So, how can we effectively recognize images without inductive bias?
  - Exploit huge amount of training data (>300M)
  - **All you need is attention!**

https://sh-tsang.medium.com/review-attention-is-all-you-need-transformer-96c787ecdec1

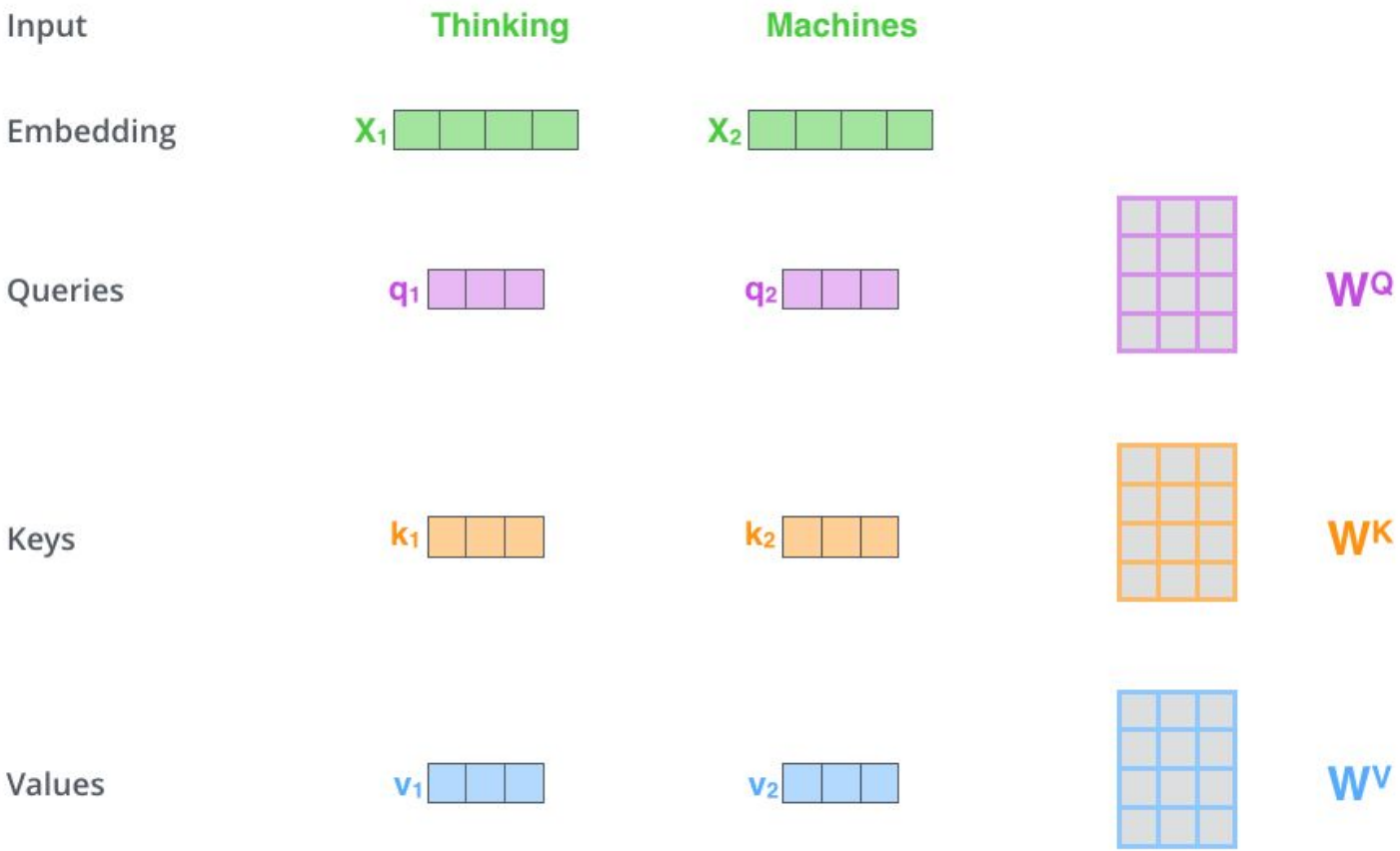
Summary!

- CNN V.S ViT

Architecture	Key component	Inductive bias	Training data	SOTA performance
CNN	Convolution, Pooling	Translation equivariance / invariance	Need less data (Good sample efficiency)	90.2% (Metal Pseudo Labels with EfficientNet-L2)
ViT	Attention	(Less than CNN…)	Need more data	90.45% (ViT-G/14)

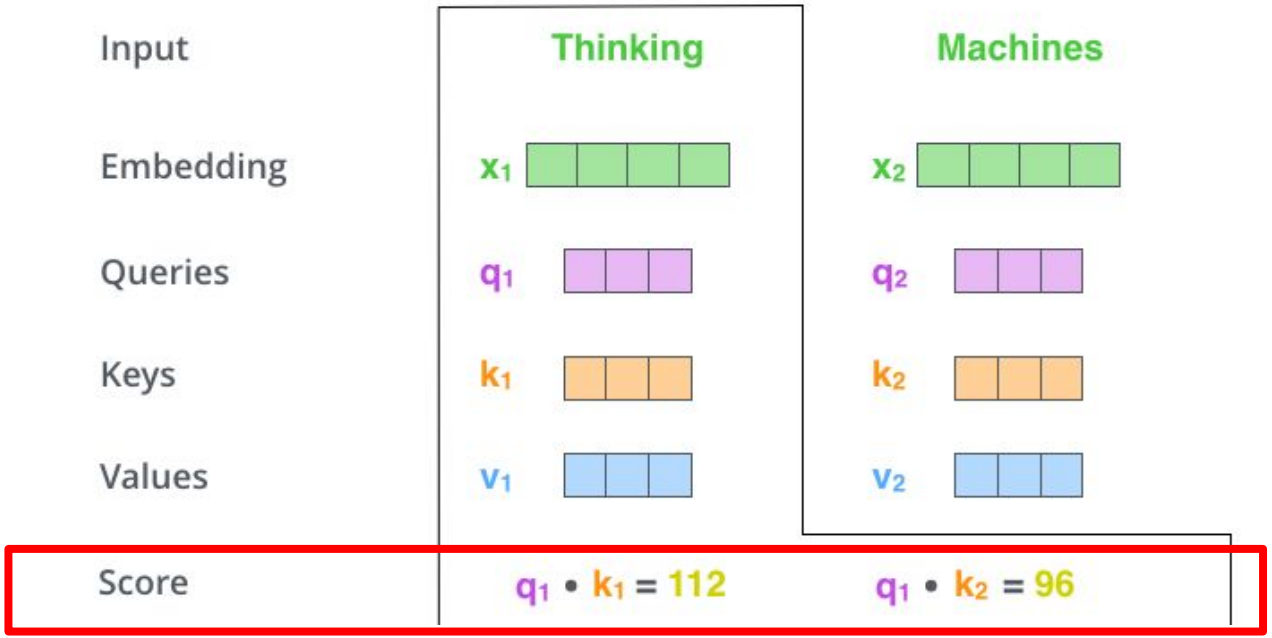
https://paperswithcode.com/sota/image-classification-on-imagenet

Attention!


Input
Thinking
Machines
Embedding
X1
X2
Queries
q1
q2
WQ
Keys
k1
k2
WK
Values
v1
v2
WV

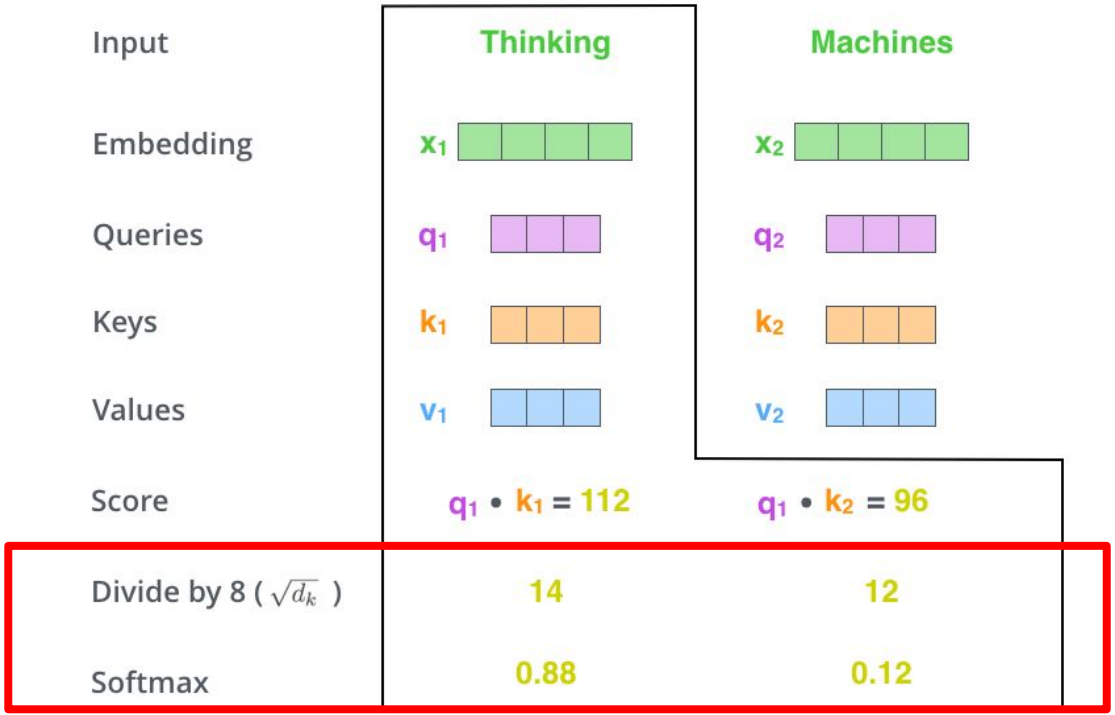
Attention! (Query, key, and value)

- Get score of “Thinking” query.



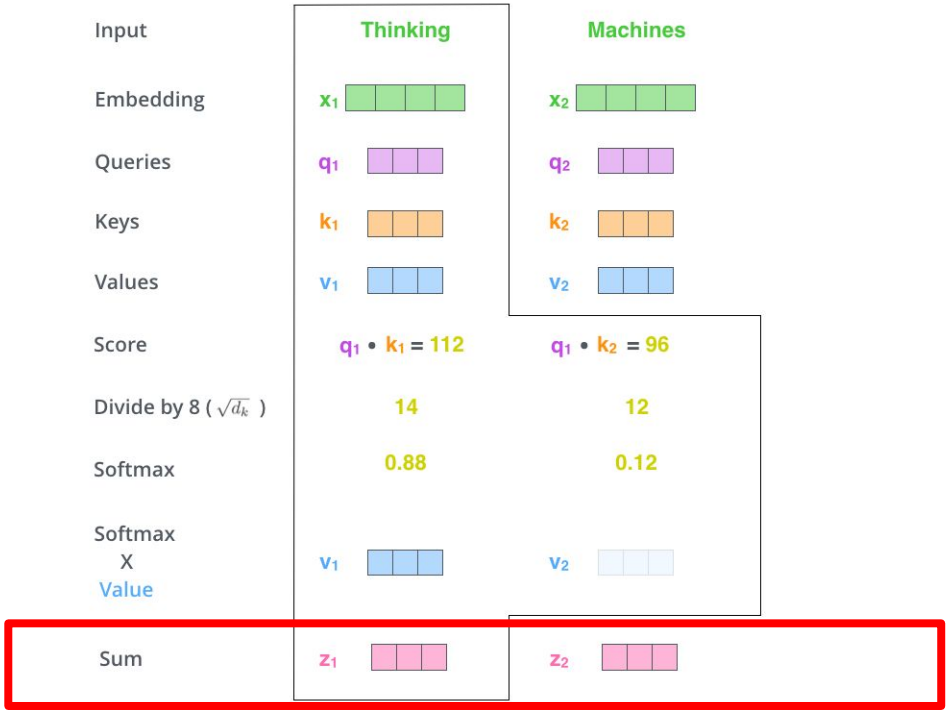
Attention! (Query, key, and value)

- Normalize the score



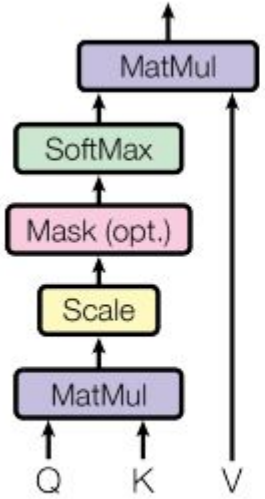
Attention! (Query, key, and value)

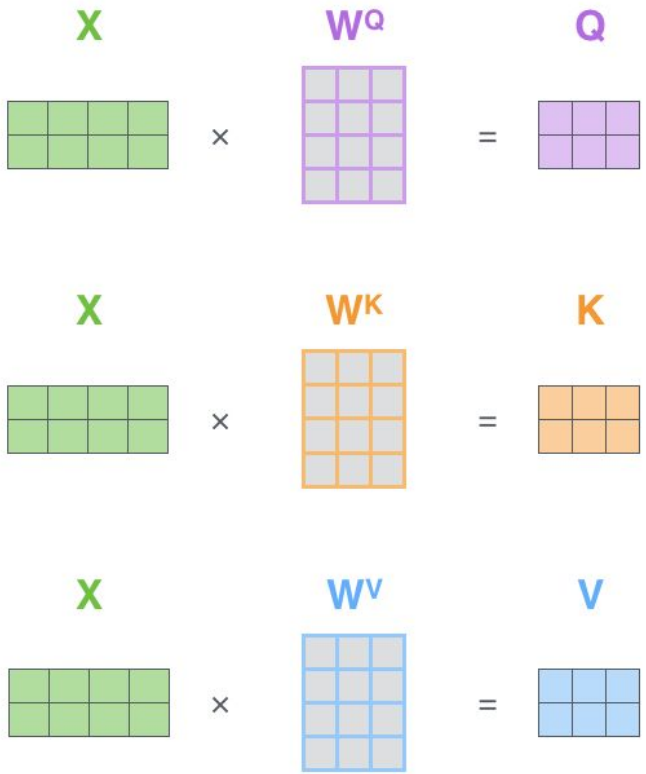
- Get final self-attention vector



Attention! (Query, key, and value)

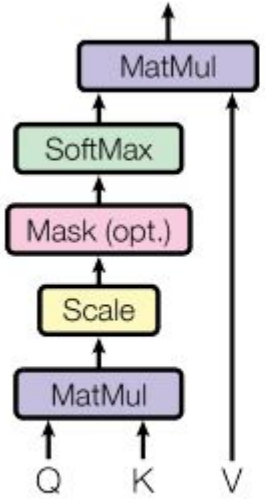
- Take a closer look





Attention! (Query, key, and value)

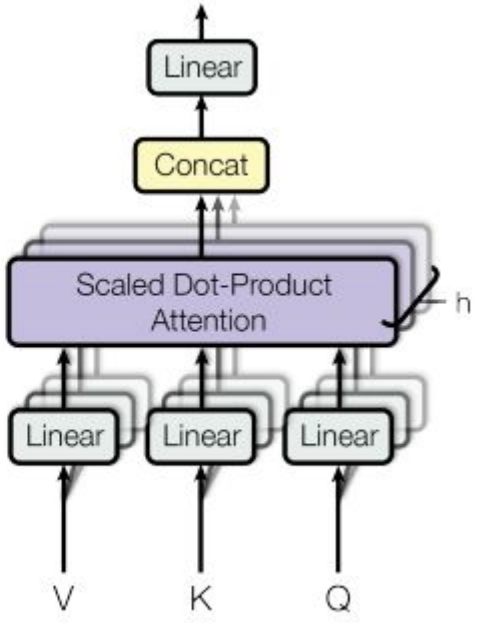
- Take a closer look

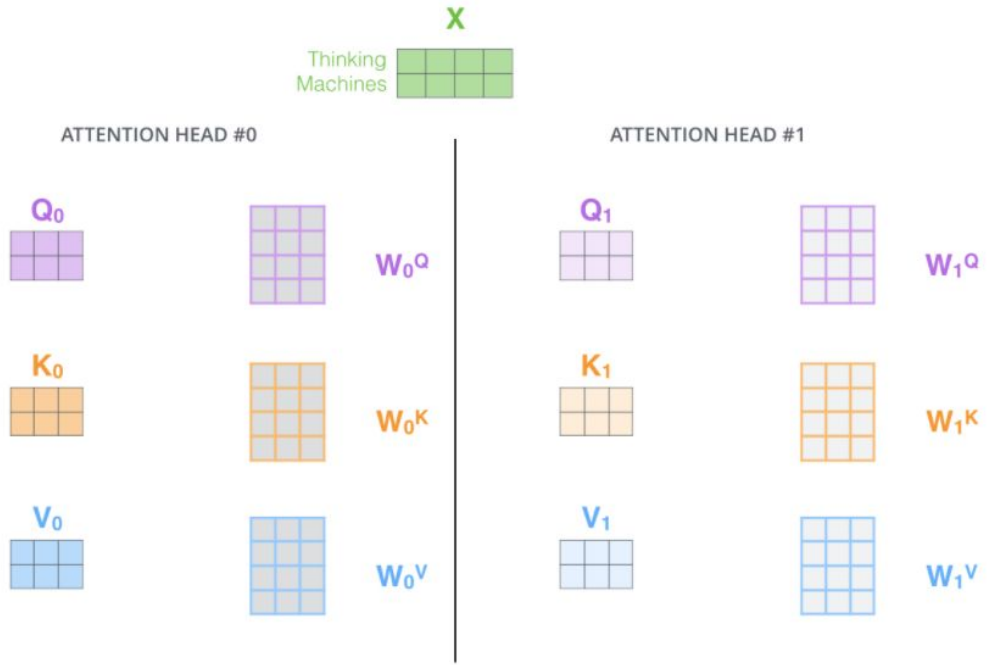


$$\text{softmax}\left(\frac{Q \times K^T}{\sqrt{d_k}}\right) V = Z$$

Attention! (Query, key, and value)

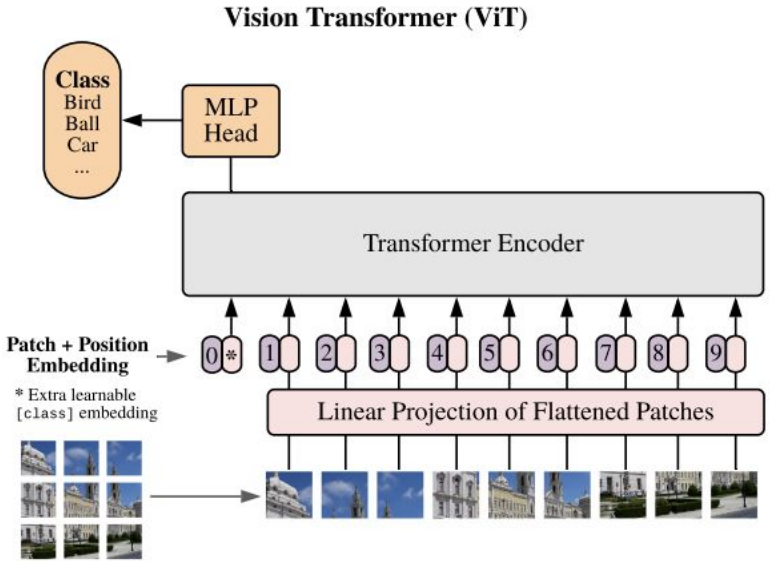
- Extend it to multi-head attention architecture

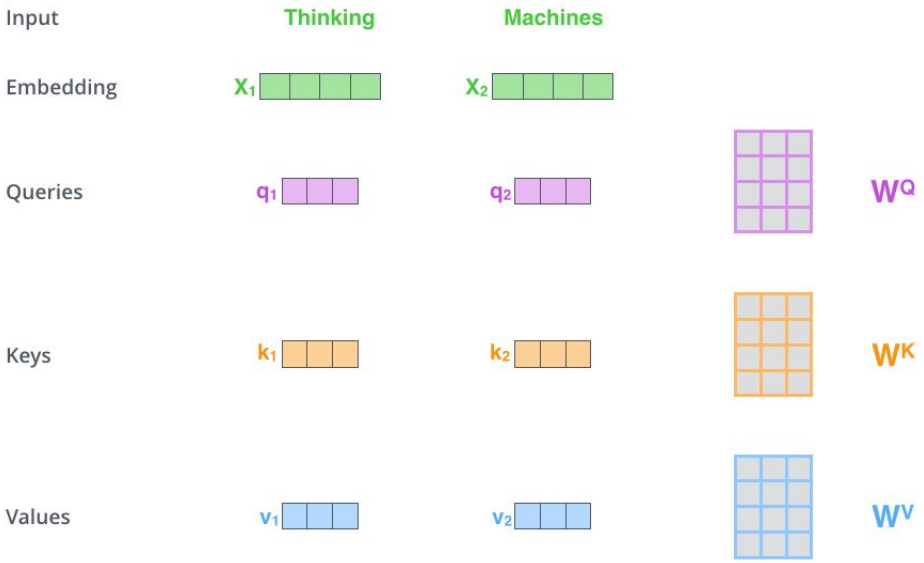




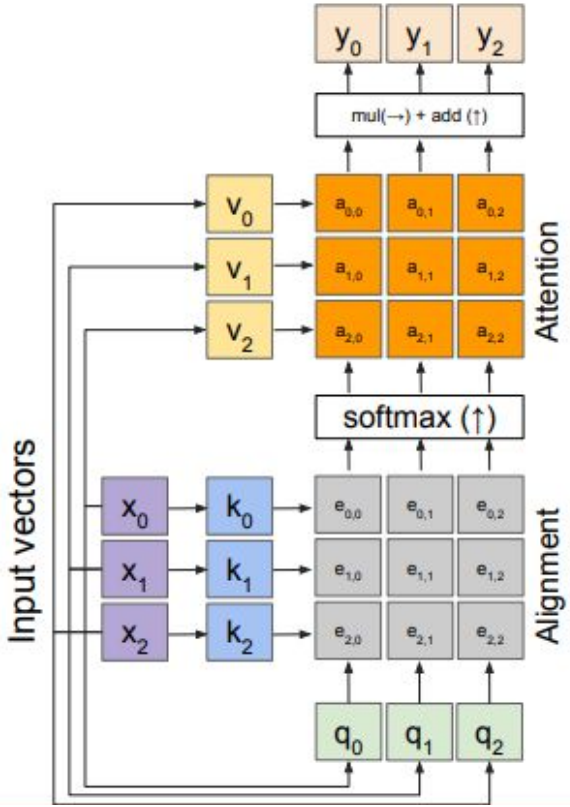
Attention in ViT

- The same attention process in image patches.
  - No convolution operation
  - No pooling layer





Attention in ViT

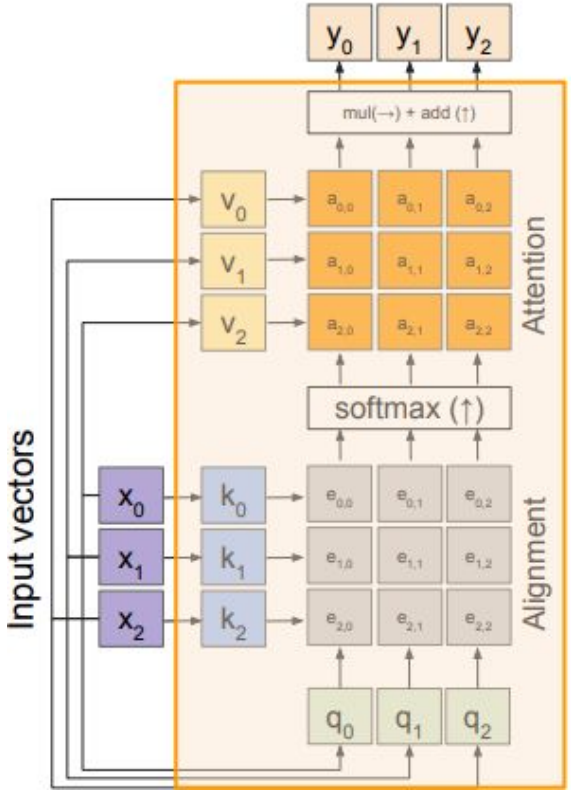


Outputs:
context vectors: **y** (shape: D_v)

Operations:
Key vectors: $\mathbf{k} = \mathbf{x}W_k$
Value vectors: $\mathbf{v} = \mathbf{x}W_v$
Query vectors: $\mathbf{q} = \mathbf{x}W_q$
Alignment: $e_{i,j} = q_j \cdot k_i / \sqrt{D}$
Attention: $\mathbf{a} = \text{softmax}(\mathbf{e})$
Output: $y_j = \sum_i a_{i,j} v_i$

Inputs:
Input vectors: **x** (shape: N x D)

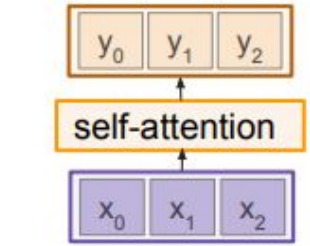
Attention in ViT

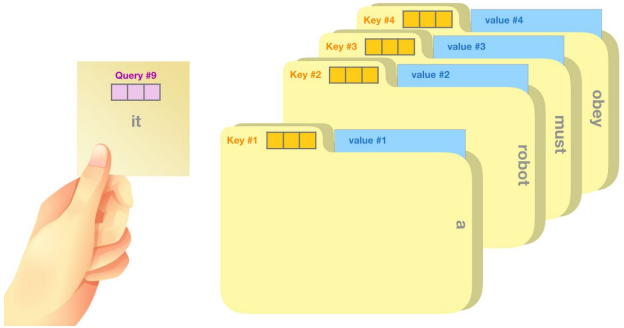


Outputs:
context vectors: **y** (shape: D_v)

Operations:
Key vectors: $\mathbf{k} = \mathbf{x}W_k$
Value vectors: $\mathbf{v} = \mathbf{x}W_v$
Query vectors: $\mathbf{q} = \mathbf{x}W_q$
Alignment: $e_{i,j} = q_j \cdot k_i / \sqrt{D}$
Attention: $\mathbf{a} = \text{softmax}(\mathbf{e})$
Output: $y_j = \sum_i a_{i,j} v_i$

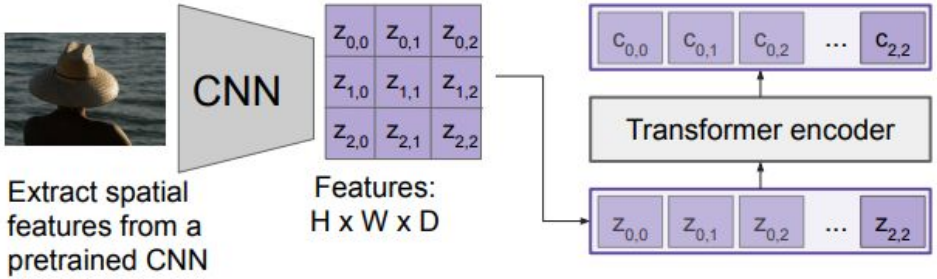
Inputs:
Input vectors: **x** (shape: N x D)



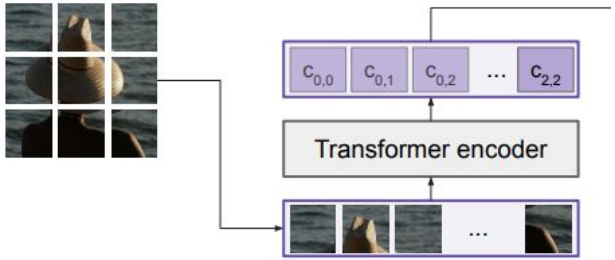


Two types of ViT

- Hybrid ViT

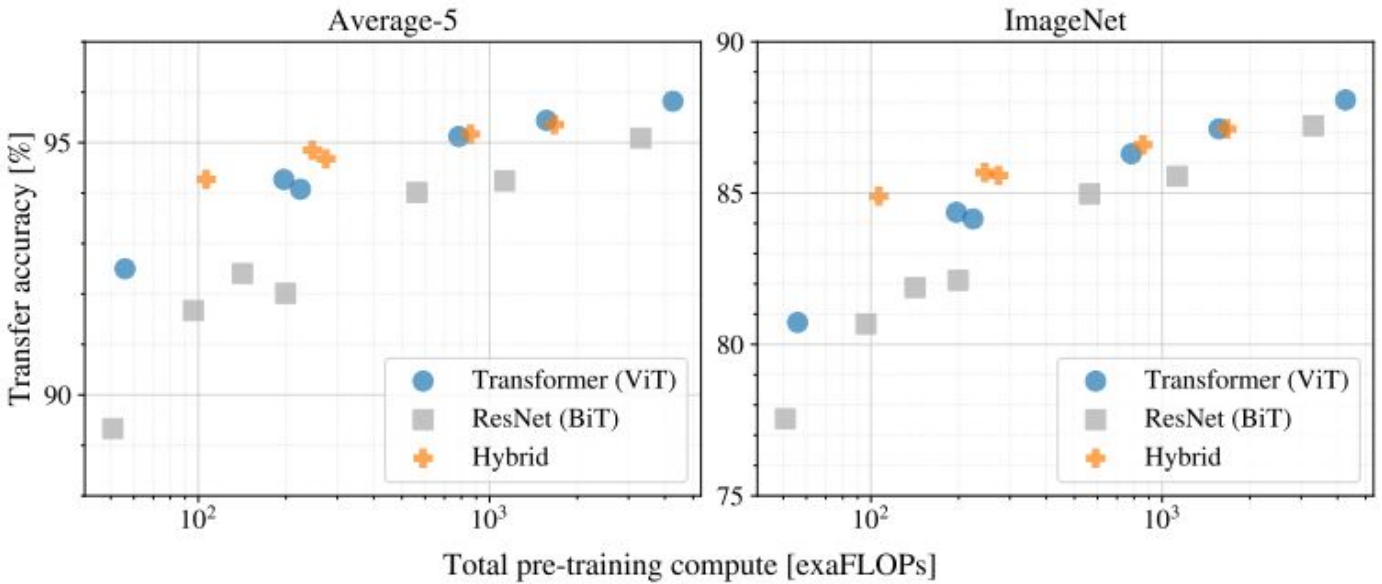


- Naive ViT (Using only pixels)



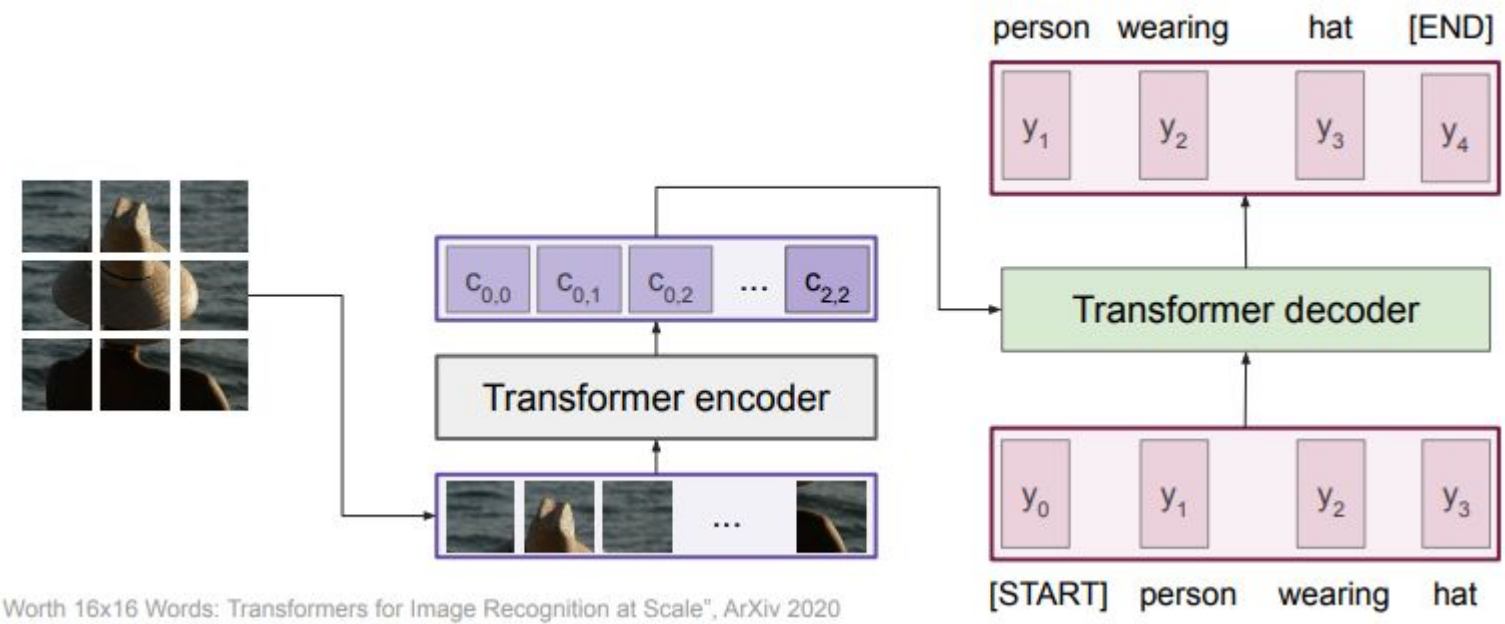
Two types of ViT

- Performance comparison
  - The more it learns, the better the performance of naive ViT.



Let’s dig into the transformer!

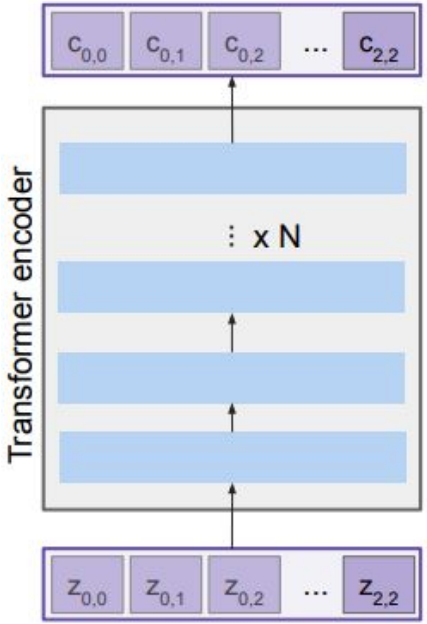
- A case study: “Image Captioning”



nage is Worth 16x16 Words: Transformers for Image Recognition at Scale", ArXiv 2020

Encoder

The Transformer encoder block

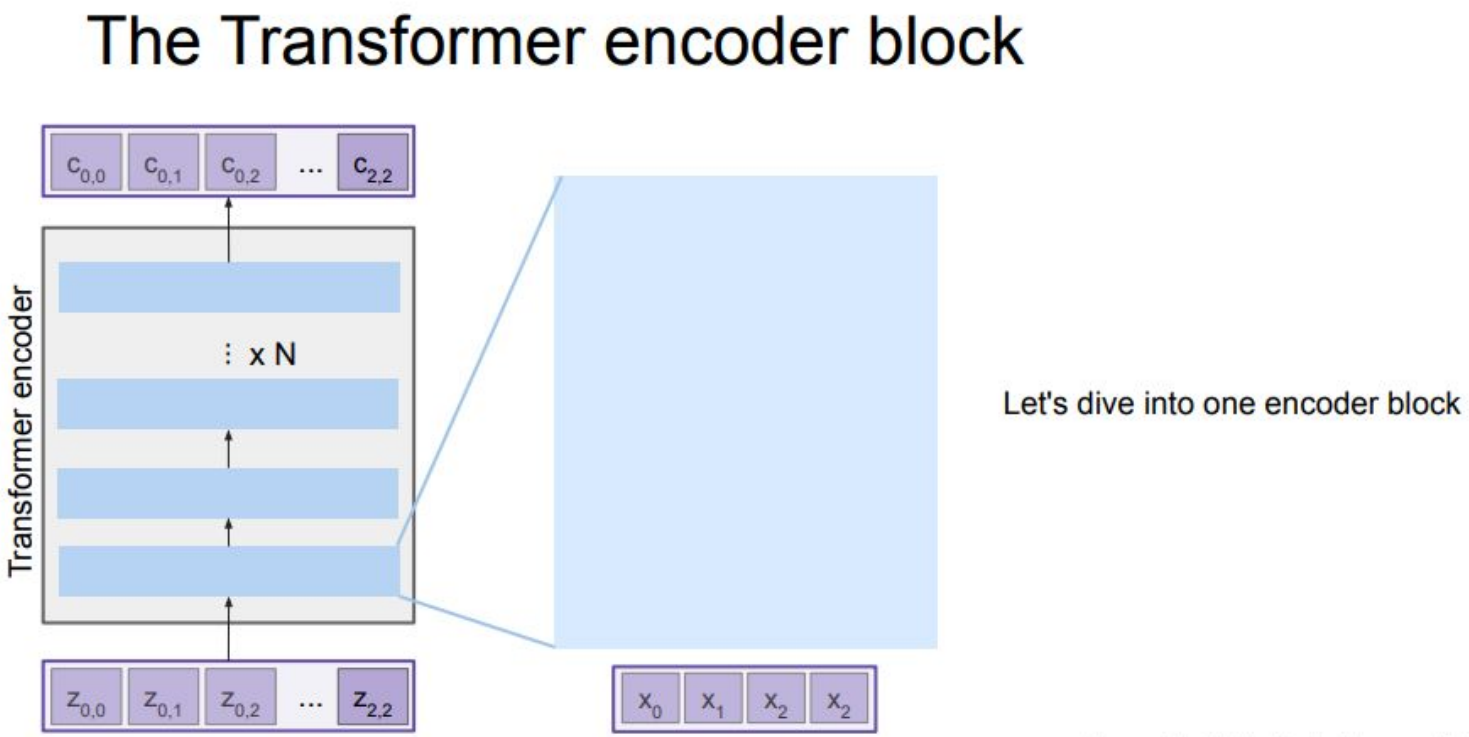


Made up of N encoder blocks.

In vaswani et al. N = 6, D_q = 512

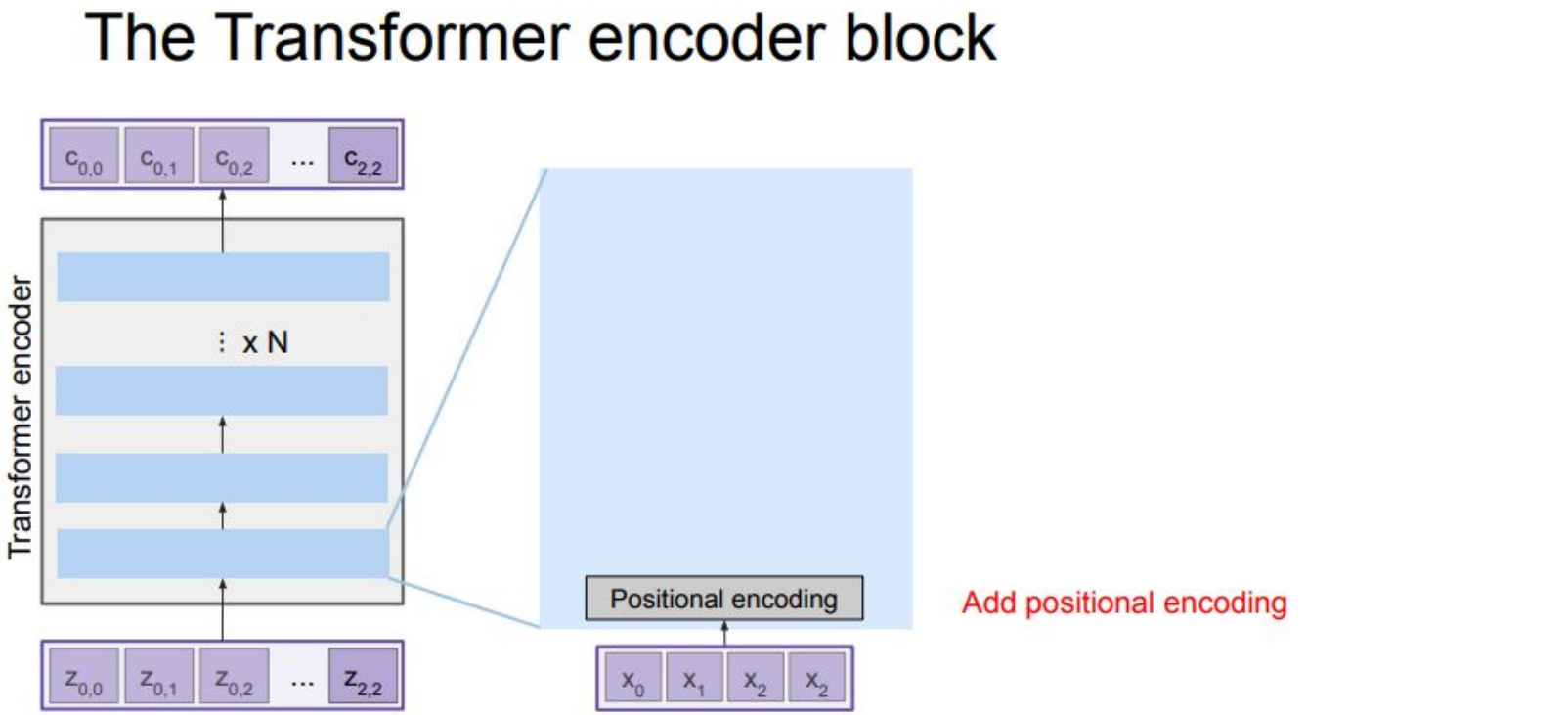
Vaswani et al, "Attention is all you need", NeurIPS 2017

Encoder



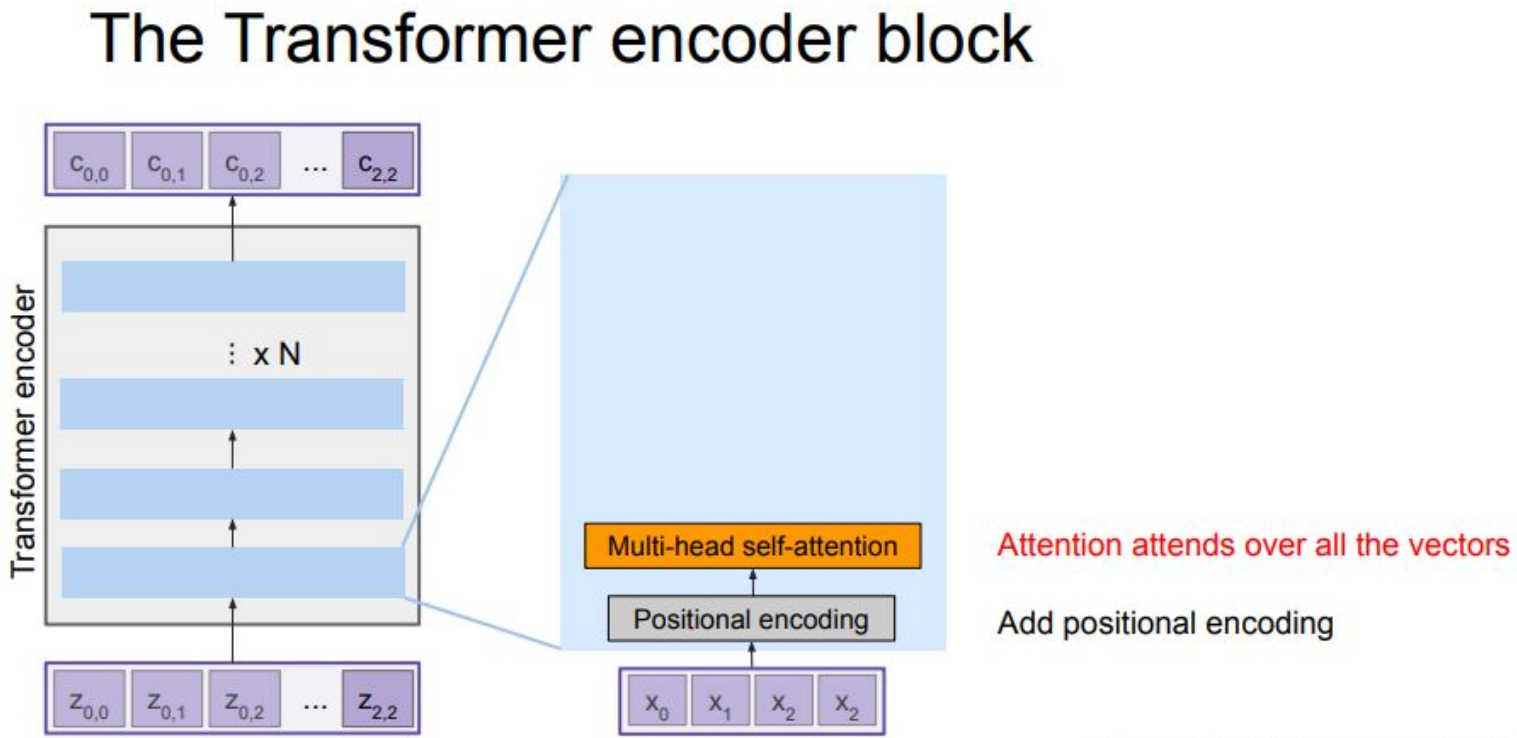
Vaswani et al, "Attention is all you need", NeurIPS 2017

Encoder

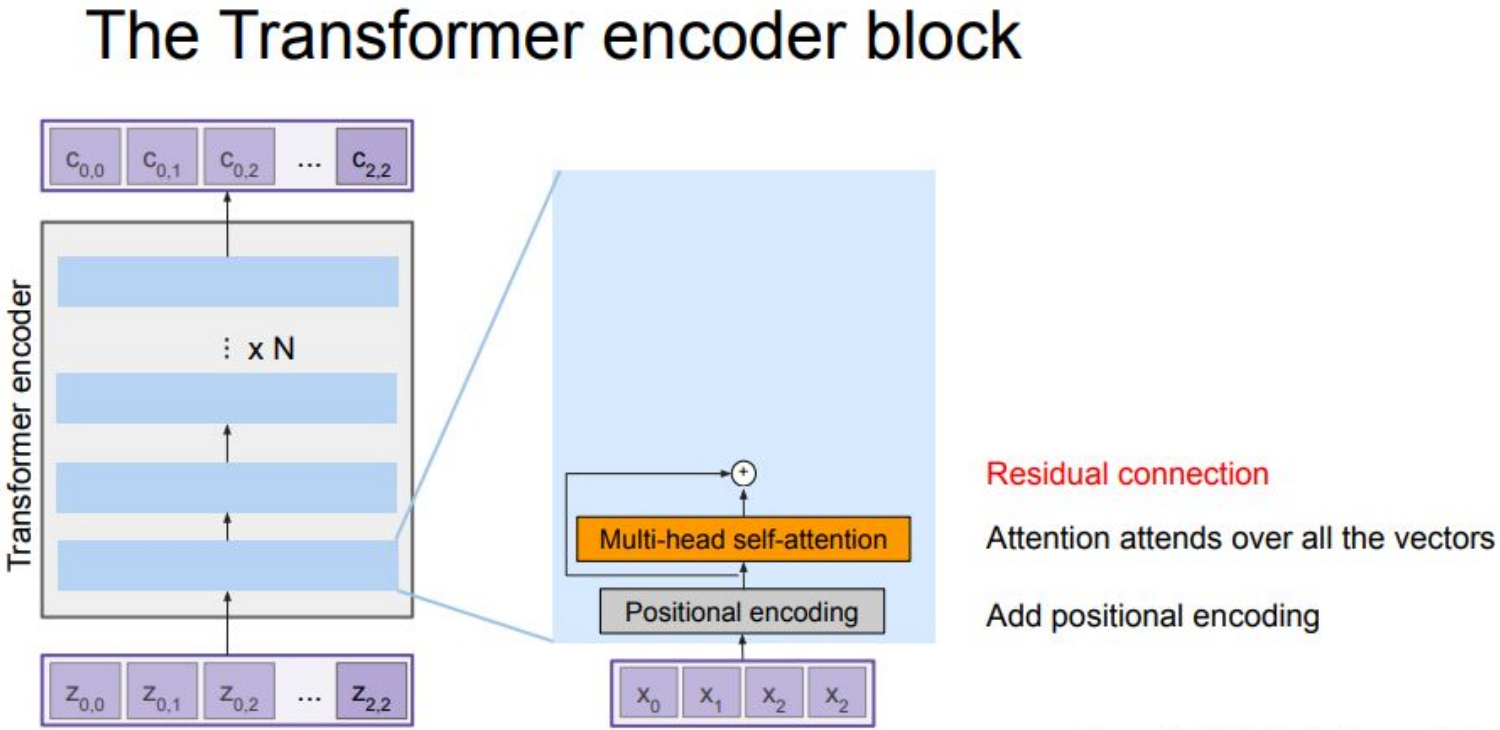


Vaswani et al, "Attention is all you need", NeurIPS 2017

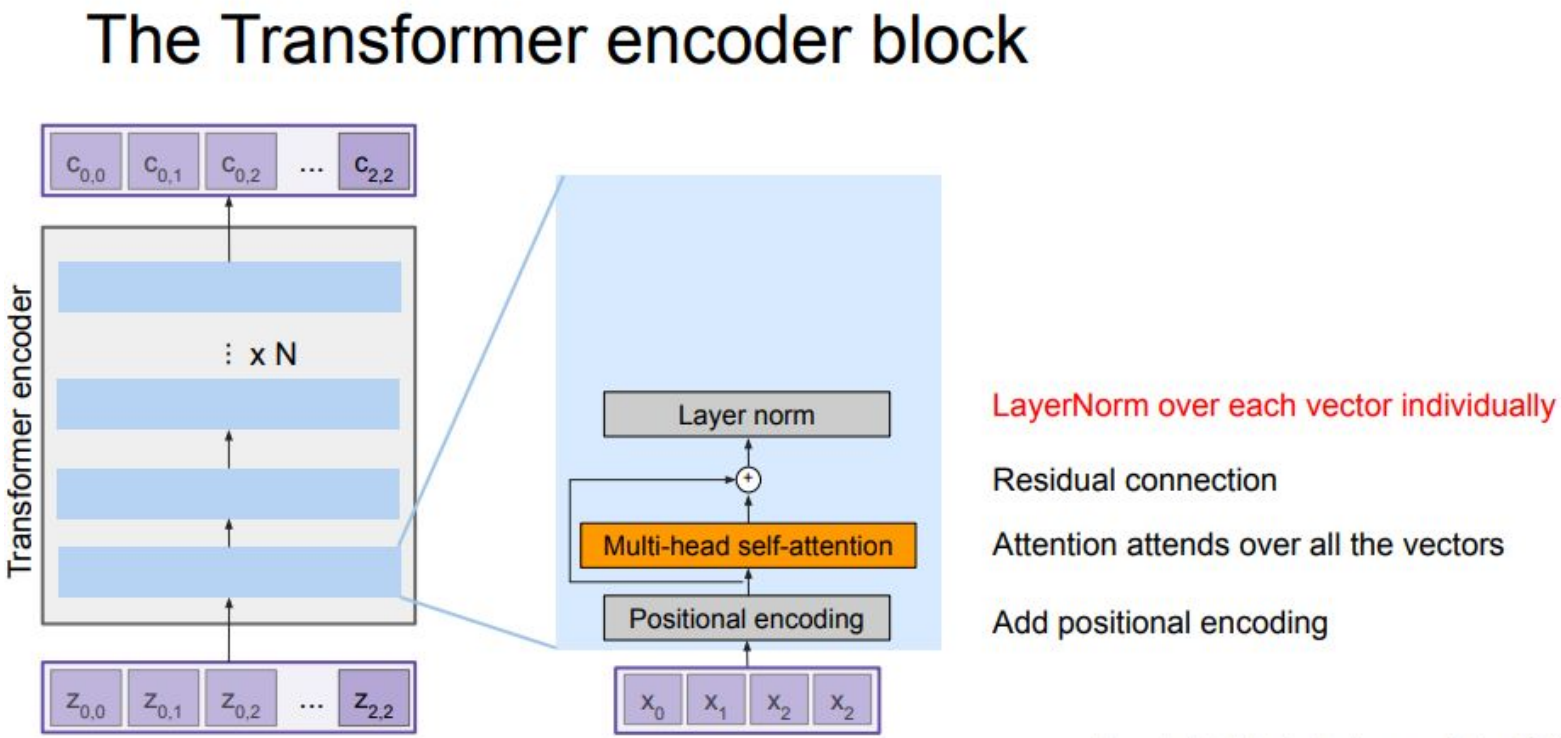
Encoder


The Transformer encoder block
Transformer encoder
x N
Multi-head self-attention
Positional encoding
Attention attends over all the vectors
Add positional encoding
Vaswani et al, "Attention is all you need", NeurIPS 2017

Encoder

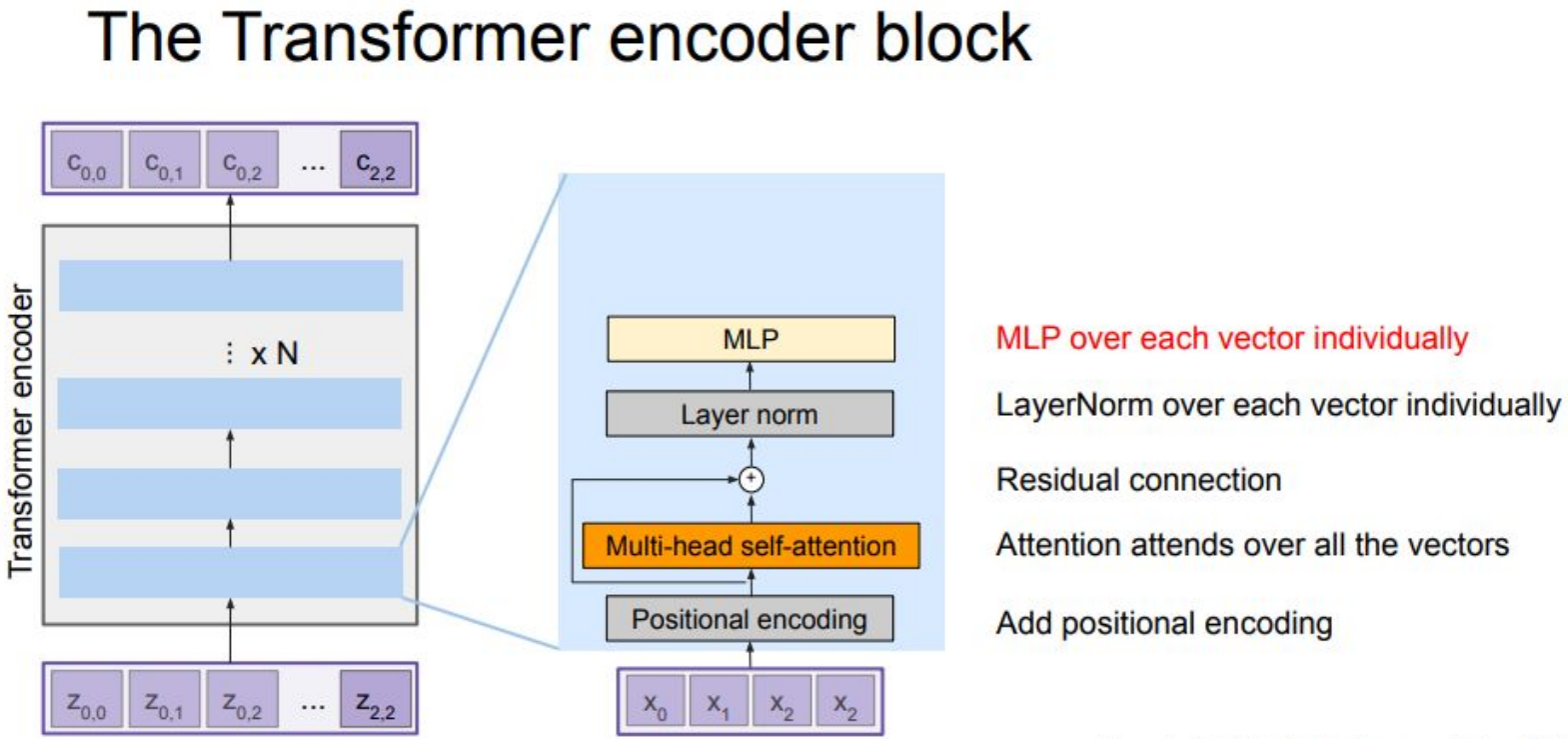

The Transformer encoder block
Transformer encoder
x N
Multi-head self-attention
Positional encoding
Residual connection
Attention attends over all the vectors
Add positional encoding
Vaswani et al, "Attention is all you need", NeurIPS 2017

Encoder



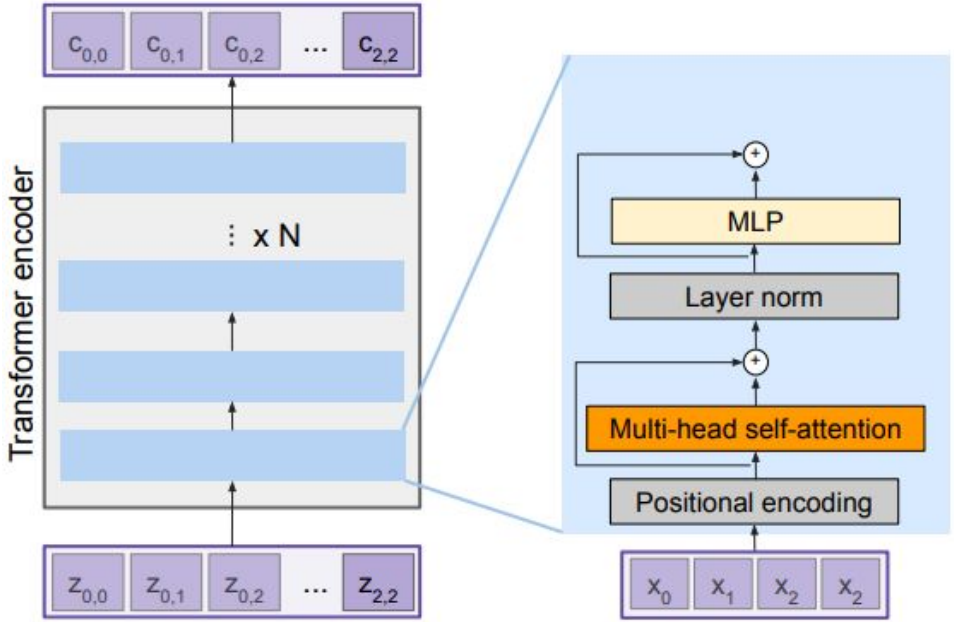
Vaswani et al, "Attention is all you need", NeurIPS 2017

Encoder


The Transformer encoder block
Transformer encoder
x N
MLP
Layer norm
Multi-head self-attention
Positional encoding
MLP over each vector individually
LayerNorm over each vector individually
Residual connection
Attention attends over all the vectors
Add positional encoding
Vaswani et al, "Attention is all you need", NeurIPS 2017

Encoder

The Transformer encoder block



Residual connection

MLP over each vector individually

LayerNorm over each vector individually

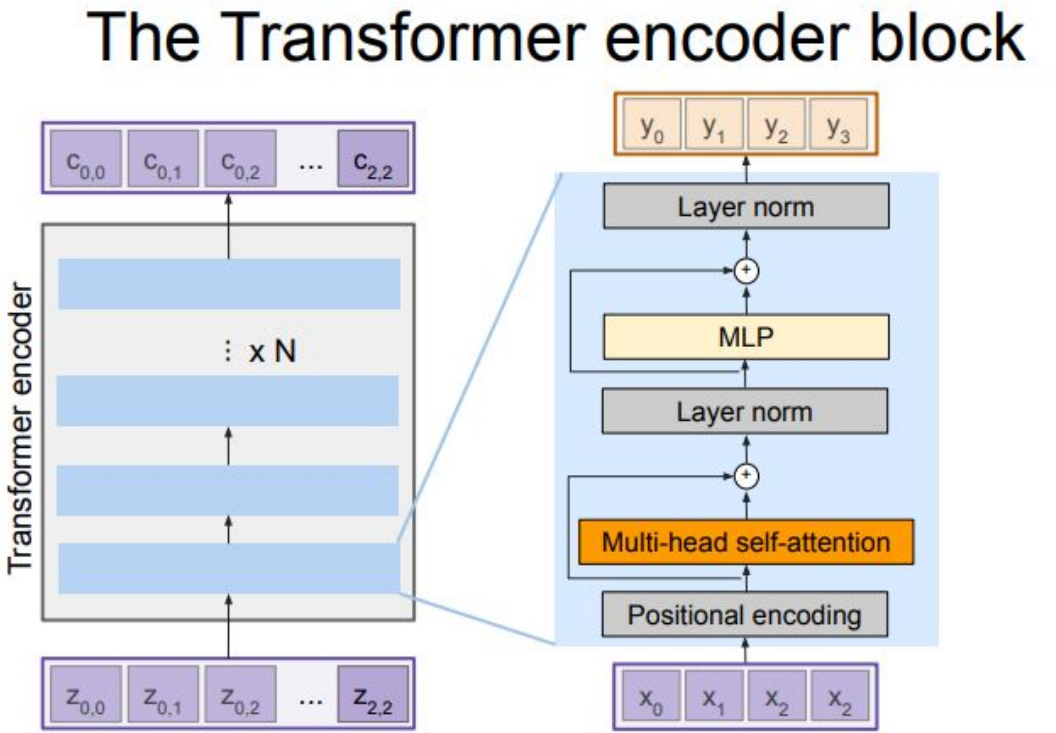
Residual connection

Attention attends over all the vectors

Add positional encoding

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

Encoder



Transformer Encoder Block:

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

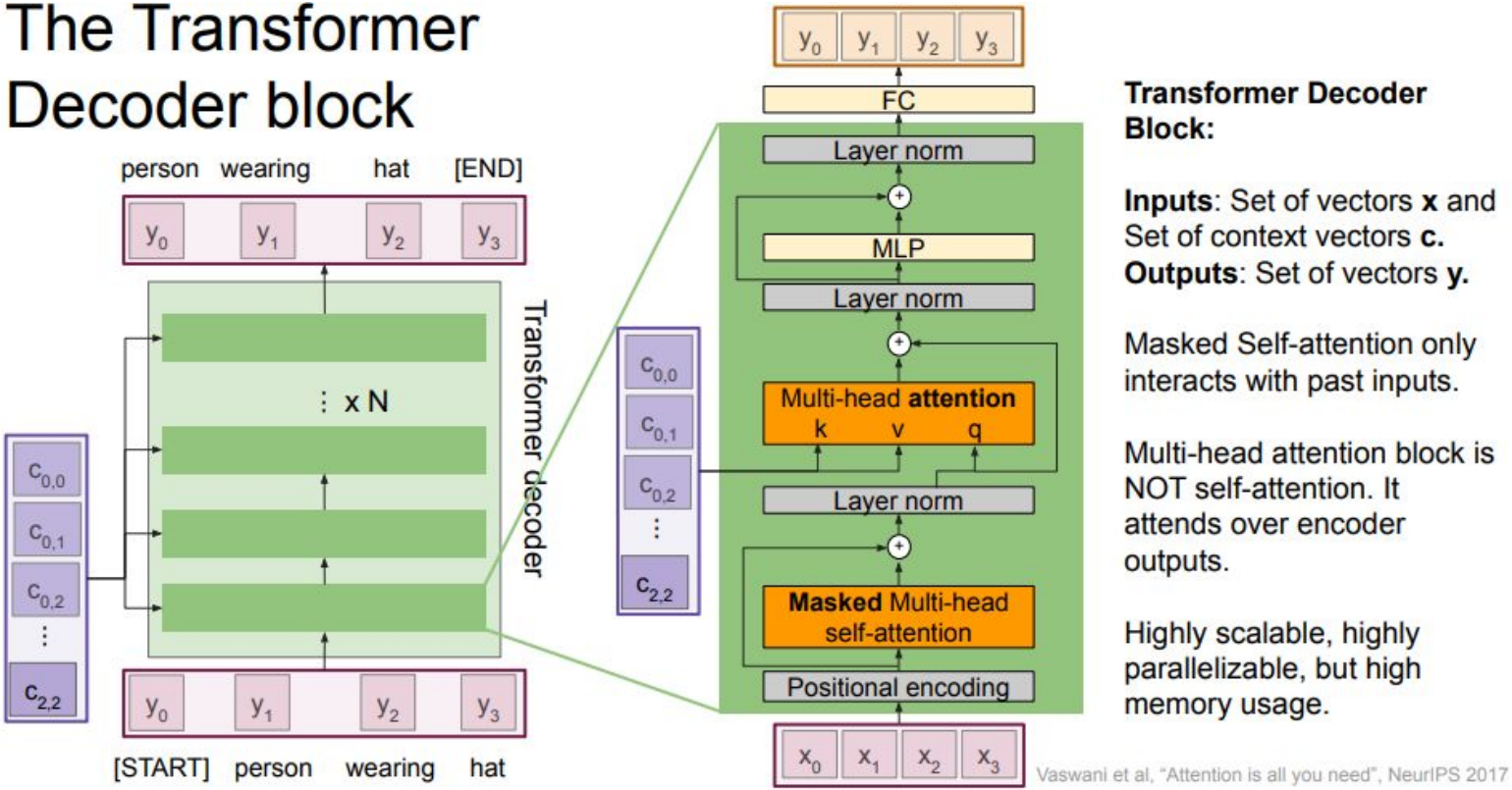
Self-attention is the only interaction between vectors.

Layer norm and MLP operate independently per vector.

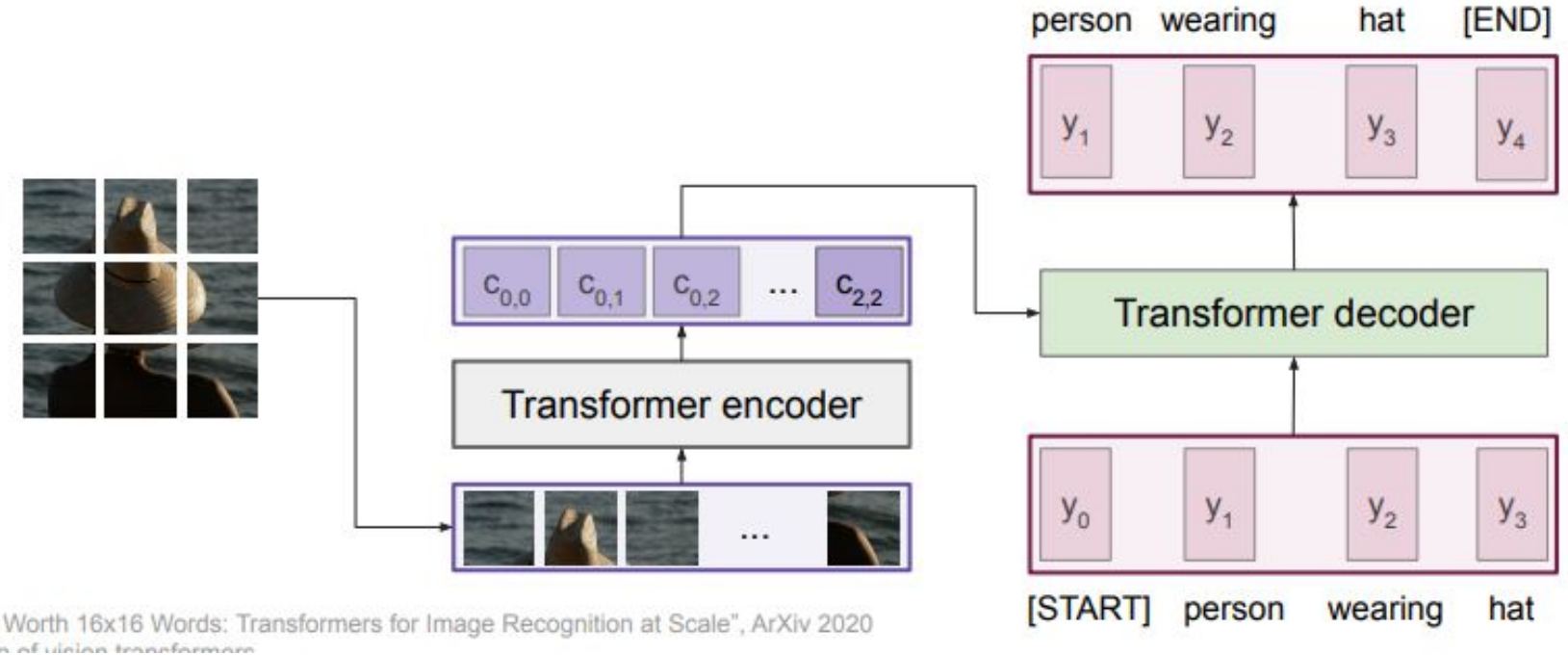
Highly scalable, highly parallelizable, but high memory usage.

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

Decoder


The Transformer Decoder block
person wearing hat [END]
y_0 y_1 y_2 y_3
⋮ x N
Transformer decoder
$c_{0,0}$ $c_{0,1}$ $c_{0,2}$ ⋮ $c_{2,2}$
[START] person wearing hat
FC
Layer norm
MLP
Layer norm
Multi-head attention
k v q
Layer norm
Masked Multi-head self-attention
Positional encoding
x_0 x_1 x_2 x_3
Transformer Decoder Block:
Inputs: Set of vectors x and Set of context vectors c.
Outputs: Set of vectors y.
Masked Self-attention only interacts with past inputs.
Multi-head attention block is NOT self-attention. It attends over encoder outputs.
Highly scalable, highly parallelizable, but high memory usage.
Vaswani et al, "Attention is all you need", NeurIPS 2017

The transformer!



e is Worth 16x16 Words: Transformers for Image Recognition at Scale", ArXiv 2020

Recap

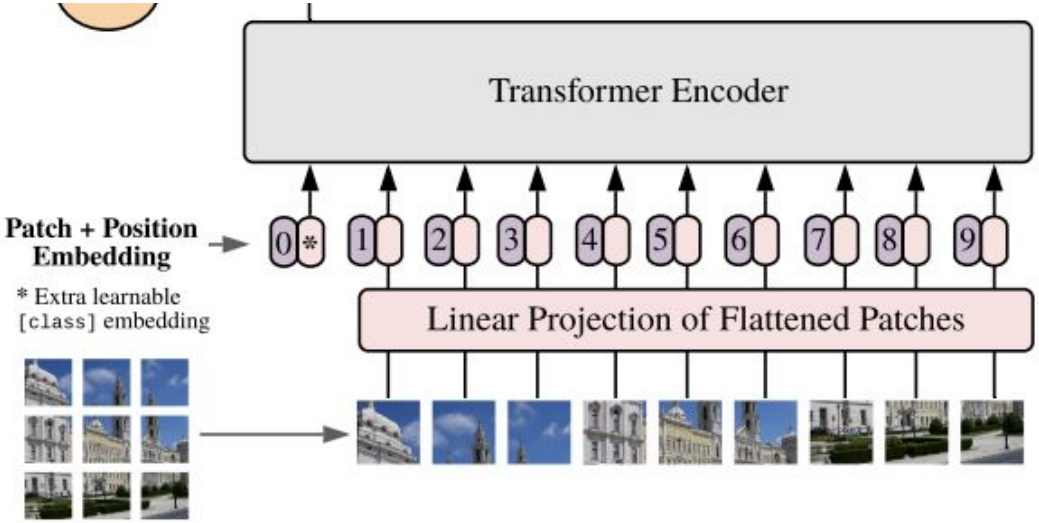
- CNN V.S ViT

Architecture	Key component	Inductive bias	Training data	SOTA performance
CNN	Convolution, Pooling	Translation equivariance / invariance	Need less data (Good sample efficiency)	90.2% (Metal Pseudo Labels with EfficientNet-L2)
ViT	Attention	(Less than CNN…)	Need more data	90.45% (ViT-G/14)

https://paperswithcode.com/sota/image-classification-on-imagenet

Inductive bias in ViT

- There are also inductive biases in ViT
  - Cropping neighboring pixels as a patch!



Transformer

- Non-vision specific model
  - Typically applied to 1-D sequence data
- Transformer “encoder”
  - A stack of alternating self-attention and MLP blocks
  - Residuals and LayerNorm
- Transformer “decoder” (not shown)
  - A slightly more involved architecture useful when the output space is different from the input space (e.g. translation)

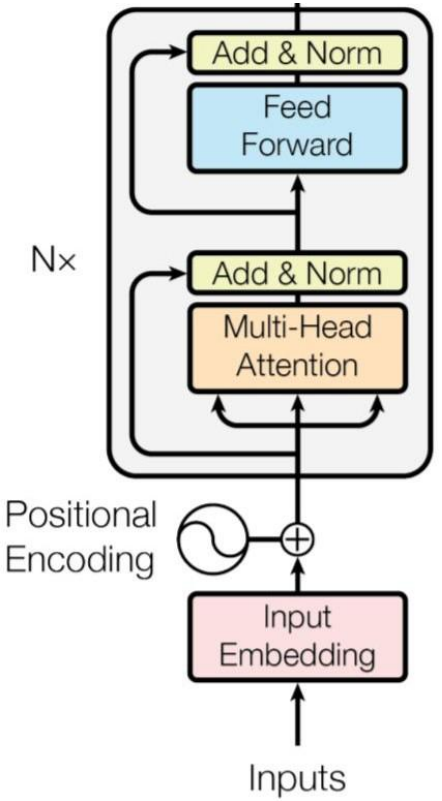
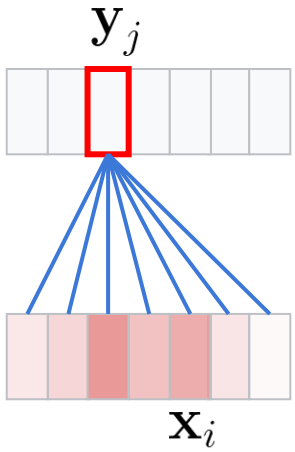


Figure from Vaswani et al.

Vaswani et al., *Attention Is All You Need*, NeurIPS 2017

Self-attention

- Each of the tokens (=vectors) attends to all tokens
  - Extra tricks: learned key, query, and value projections, inverse-sqrt scaling in the softmax, and multi-headed attention (omit for simplicity)
- It's a set operation (permutation-invariant)
  - ...and hence need "position embeddings" to "remember" the spatial structure
- It's a global operation
  - Aggregates information from all tokens



$$\boldsymbol{\alpha}_j = softmax(\frac{K\mathbf{x}_1 \cdot Q\mathbf{x}_j}{\sqrt{d_K}}, \ldots, \frac{K\mathbf{x}_n \cdot Q\mathbf{x}_j}{\sqrt{d_K}})$$

$$\mathbf{y}_j = \sum_{i=1}^{n} \alpha_{ji} V\mathbf{x}_i$$

Simplified! Multi-headed attention not shown

Vaswani et al., *Attention Is All You Need*, NeurIPS 2017

Self-Attention with Queries, Keys, Values

Make three version of each input embedding x(i)

- **Query** vector $\mathbf{q}^{(i)} = \mathbf{W}_q \mathbf{x}^{(i)}$
- **Key** vector: $\mathbf{k}^{(i)} = \mathbf{W}_k \mathbf{x}^{(i)}$
- **Value** vector: $\mathbf{v}^{(i)} = \mathbf{W}_v \mathbf{x}^{(i)}$

The **attention weight of the *j*-th position** to compute the **new output for the *i*-th position** depends on the **query of i** and the **key of j (scaled):**

$$w_j^{(i)} = \frac{\exp(\mathbf{q}^{(i)}\mathbf{k}^{(j)})/\sqrt{k}}{\sum_j (\exp(\mathbf{q}^{(i)}\mathbf{k}^{(j)})/\sqrt{k})}$$

The **new output vector for the i-th position** depends on the **attention weights** and **value** vectors of all **input positions j**:

$$\mathbf{y}^{(i)} = \sum_{j=1..T} w_j^{(i)} \mathbf{v}^{(j)}$$

[Julia Hockenmaier, Lecture 9, UIUC]

Transformer self-attention layer

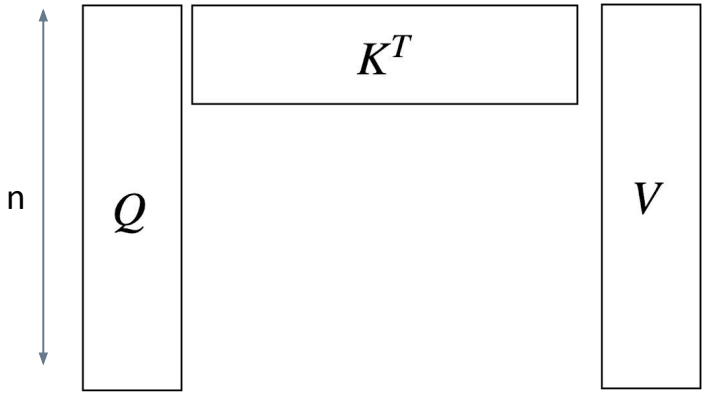
Input: X (matrix of n embedding vectors, each dim m)

Parameters (learned): W_Q, W_K, W_V

Compute:

$$Q = XW_Q$$
$$K = XW_K$$
$$V = XW_V$$

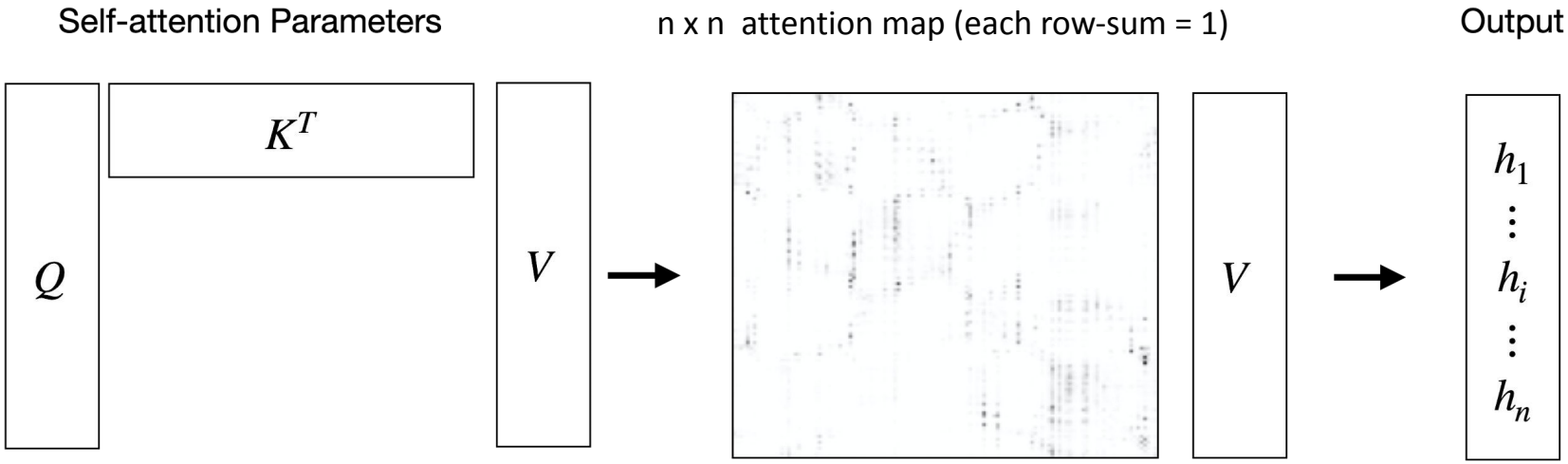
Self-attention Parameters



$$Q, K, V \in \mathbb{R}^{n \times m}$$

$$QK^T \in \mathbb{R}^{n \times n}$$

Transformer self-attention

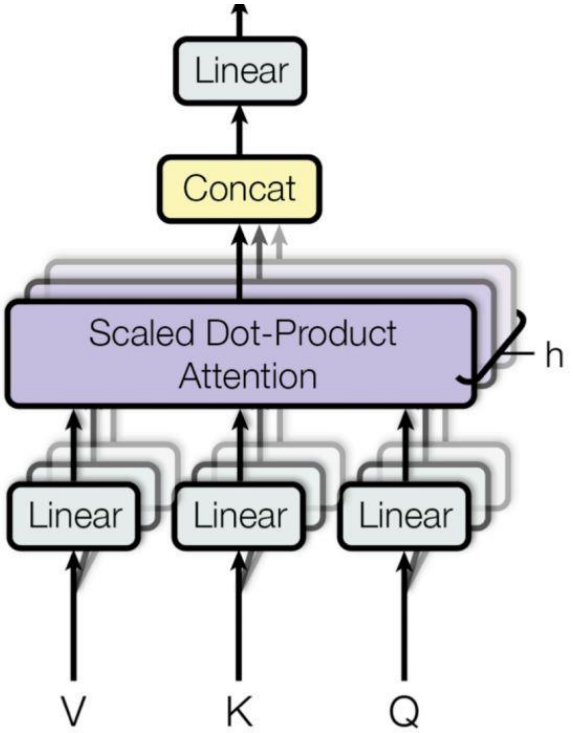


Output matrix H = $\mathrm{softmax}(\frac{1}{\sqrt{d}} QK^T) \cdot V$

Self-attention explicitly models interactions between all pairs of input embeddings

Multi-Head attention

- Learn *h* different linear projections of Q,K,V
- Compute attention separately on each of these *h* versions
- Concatenate and project the resultant vectors to a lower dimensionality.
- Each attention head can use low dimensionality



$$\text{MultiHead}(Q,K,V) = \text{Concat}(\text{head}_1,...,\text{head}_\text{h})W^O$$

$$\text{where head}_\text{i} = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

[Julia Hockenmaier, Lecture 9, UIUC]

Positional Encoding (1-D)

How to capture sequence order?

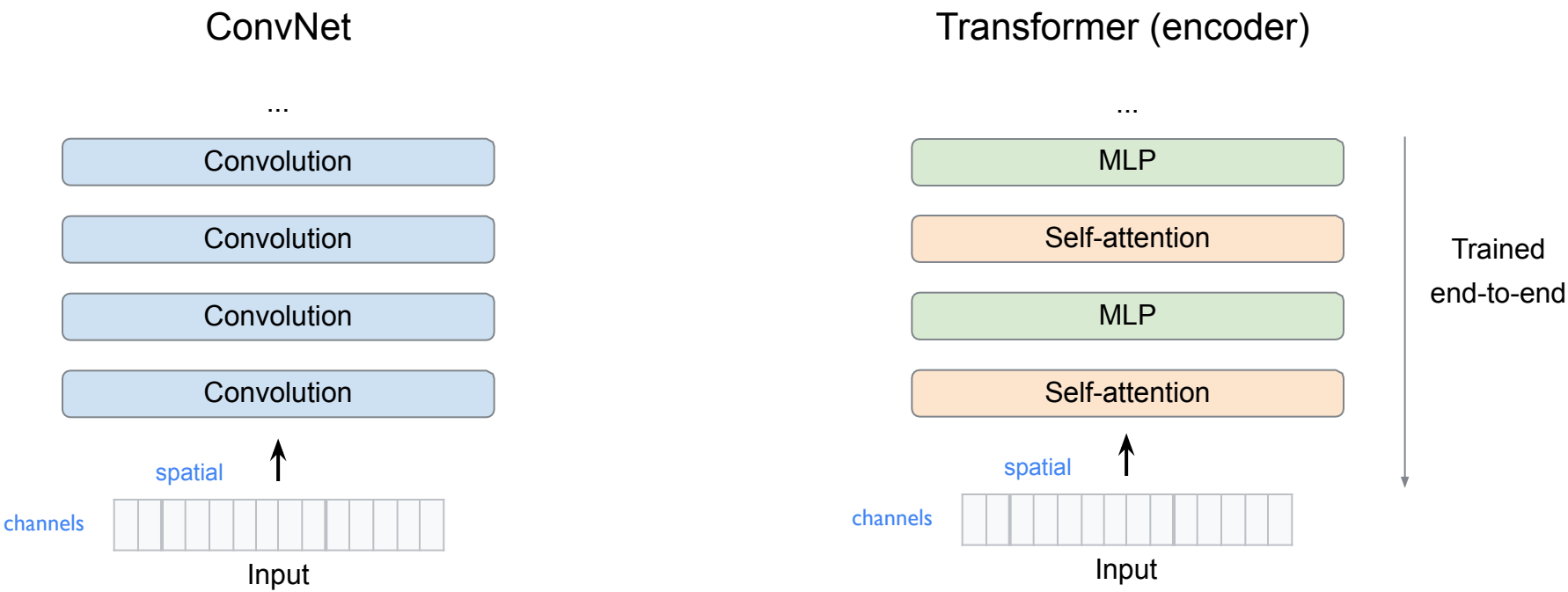
Add positional embeddings to input embeddings

-Same dimension
-Can be learned or fixed

Fixed encoding: sin / cos of different frequencies:

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

ConvNet vs Transformer



Convolutions (with kernels > 1x1) mix both the channels and the spatial locations

MLPs (=1x1 convs) only mix the channels, per location
Self-attention mixes the spatial locations (and channels a bit)

*ResNets have grouped of 1x1 convolutions that are nearly identical to transformer's MLPs

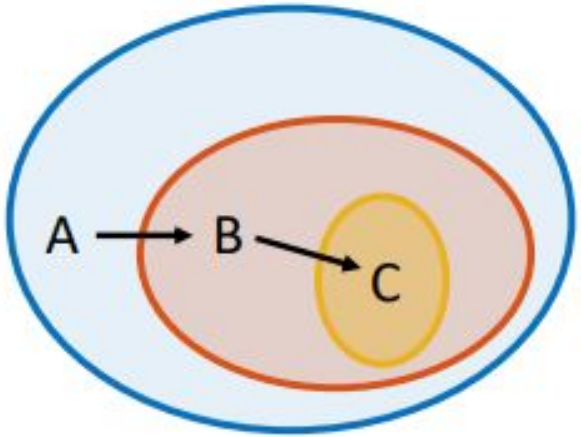
ResNet strikes back! (arXiv 2021)

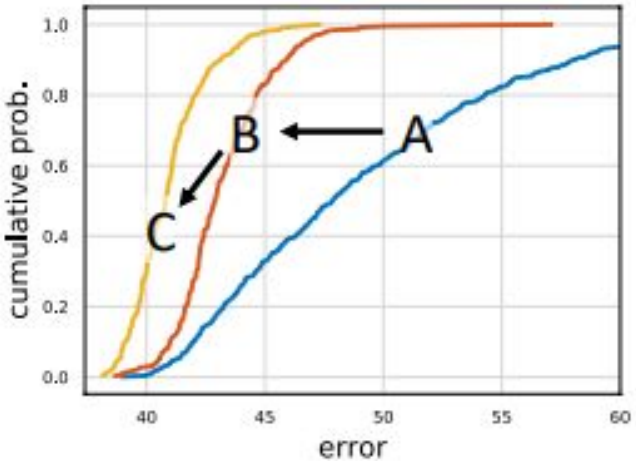
- CNN is still good in the same experimental settings.

	A1-A2-org.		A3		Cost							ImageNet-1k-val			
	train	test	train	test	A1	A2	A1-A2		A3			A1	A2	A3	org.
↓ Architecture	res.	res.	res.	res.	time (hour)		# GPU	Pmem	time	# GPU	Pmem	Accuracy(%)			
ResNet-18 [13]†	224	224	160	224	186	93	2	12.5	28	2	6.5	71.5	70.6	68.2	69.8
ResNet-34 [13]†	224	224	160	224	186	93	2	17.5	27	2	9.0	76.4	75.5	73.0	73.3
ResNet-50 [13]†	224	224	160	224	110	55	4	22.0	15	4	11.4	80.4	79.8	78.1	76.1
ResNet-101 [13]†	224	224	160	224	74	37	8	16.3	8	8	8.5	81.5	81.3	79.8	77.4
ResNet-152 [13]†	224	224	160	224	92	46	8	22.5	9	8	11.8	82.0	81.8	80.6	78.3
RegNetY-4GF [32]	224	224	160	224	130	65	4	27.1	15	4	13.9	81.5	81.3	79.0	79.4
RegNetY-8GF [32]	224	224	160	224	106	53	8	19.8	10	8	10.3	82.2	82.1	81.1	79.9
RegNetY-16GF [32]	224	224	160	224	150	75	8	25.6	13	8	13.4	82.0	82.2	81.7	80.4
RegNetY-32GF [32]	224	224	160	224	120	60	16	17.6	12	16	9.4	82.5	82.4	82.6	81.0
SE-ResNet-50 [20]	224	224	160	224	102	51	4	27.6	16	4	14.2	80.0	80.1	77.0	76.7
SENet-154 [20]	224	224	160	224	110	55	16	23.3	12	16	12.2	81.7	81.8	81.9	81.3
ResNet-50-D [14]	224	224	160	224	100	50	4	23.9	14	4	12.3	80.7	80.2	78.7	79.3
ResNeXt-50-32x4d [51]†	224	224	160	224	80	40	8	14.3	15	4	14.6	80.5	80.4	79.2	77.6
EfficientNet-B0 [41]	224	224	160	224	110	55	4	22.1	15	4	11.4	77.0	76.8	73.0	77.1
EfficientNet-B1 [41]	240	240	160	224	62	31	8	17.9	8	8	7.9	79.2	79.4	74.9	79.1
EfficientNet-B2 [41]	260	260	192	256	76	38	8	22.8	9	8	11.9	80.4	80.1	77.5	80.1
EfficientNet-B3 [41]	300	300	224	288	62	31	16	19.5	6	16	10.1	81.4	81.4	79.2	81.6
EfficientNet-B4 [41]	380	380	320	380	64	32	32	20.4	8	32	14.3	81.6	82.4	81.2	82.9
ViT-Ti [45]*	224	224	160	224	98	49	4	16.3	14	4	7.0	74.7	74.1	66.7	72.2
ViT-S [45]*	224	224	160	224	68	34	8	16.1	8	8	7.0	80.6	79.6	73.8	79.8
ViT-B [11]*	224	224	160	224	66	33	16	16.4	5	16	7.3	80.4	79.8	76.0	81.8
timm [50] specific architectures															
ECA-ResNet-50-T	224	224	160	224	112	56	4	29.3	15	4	15.0	81.3	80.9	79.6	-
EfficientNetV2-rw-S [42]	288	384	224	288	52	26	16	16.6	7	16	10.1	82.3	82.9	80.9	83.8
EfficientNetV2-rw-M [42]	320	384	256	352	64	32	32	18.5	9	32	12.1	80.6	81.9	82.3	84.8
ECA-ResNet-269-D	320	416	256	320	108	54	32	27.4	11	32	17.8	83.3	83.9	83.3	85.0

RegNet

- Designing Network Design Spaces, CVPR 2020





NFNet

- High-Performance Normalizer-Free ResNets, ICML 2021

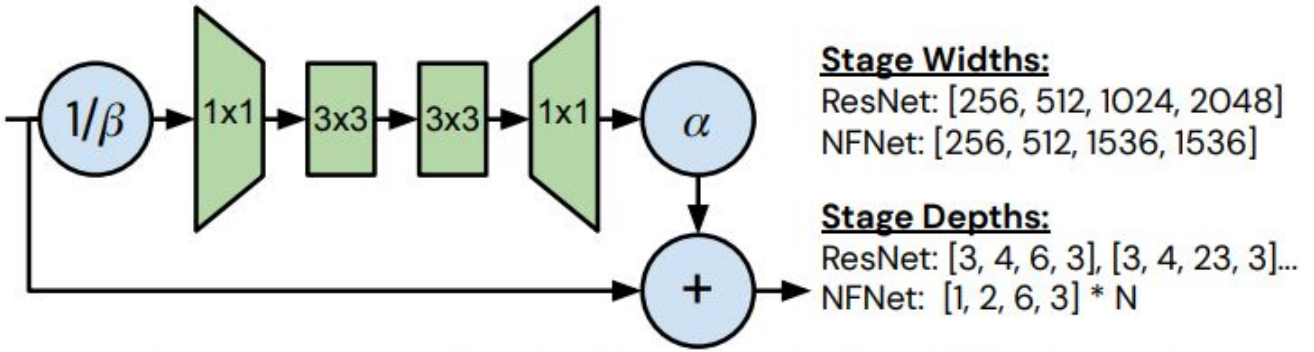


Figure 3. Summary of NFNet bottleneck block design and architectural differences. See Figure 5 in Appendix C for more details.

TResNet

- TResNet: High Performance GPU-Dedicated Architecture, WACV 2021

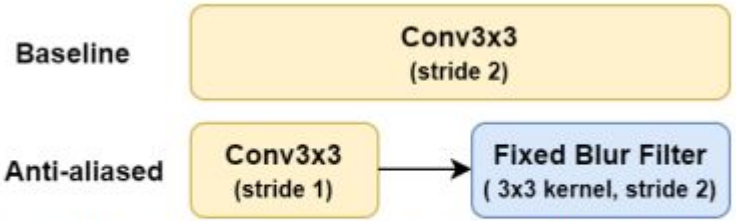
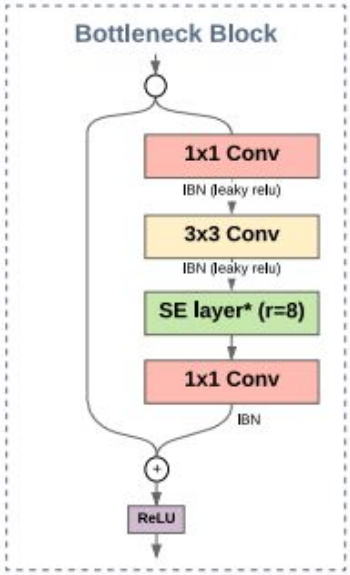
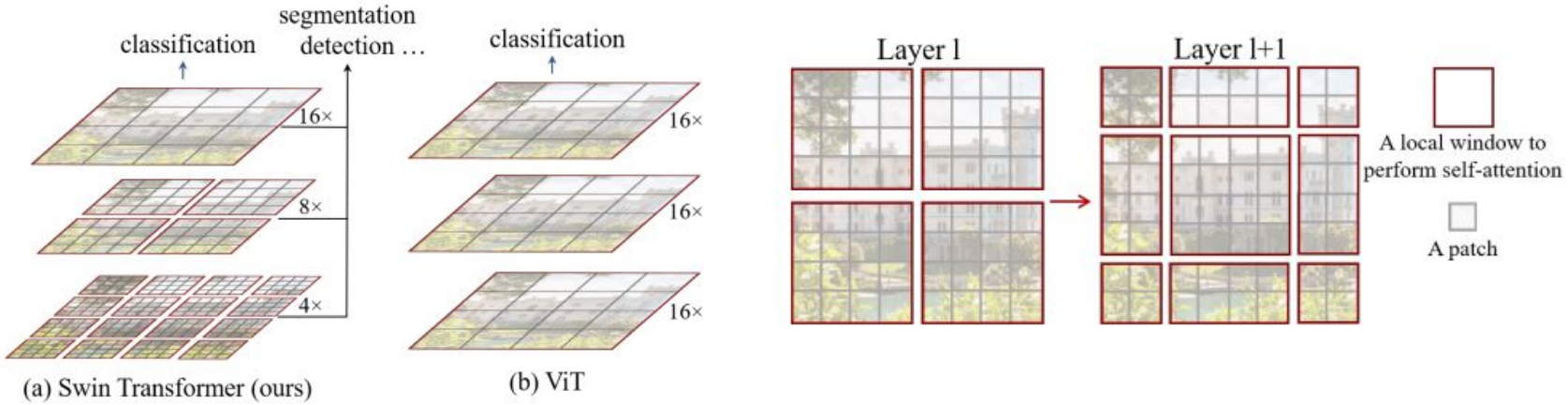


Figure 2. **The AA downsampling scheme of TResNet architecture**. All stride-2 convolutions are replaced by stride-1 convolutions, followed by a fixed downsampling blur filter [46].



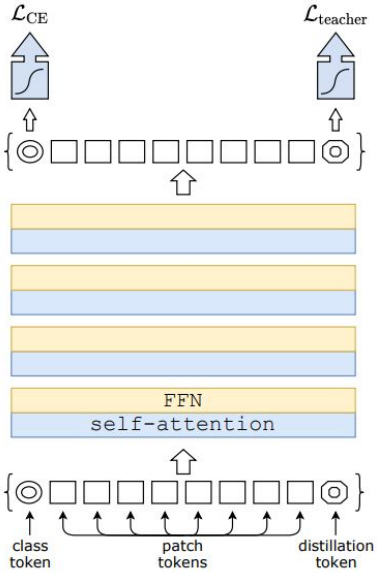
Swin Transformer

- Swin Transformer: Hierarchical Vision Transformer using Shifted Windows, ICCV 2021



DeiT

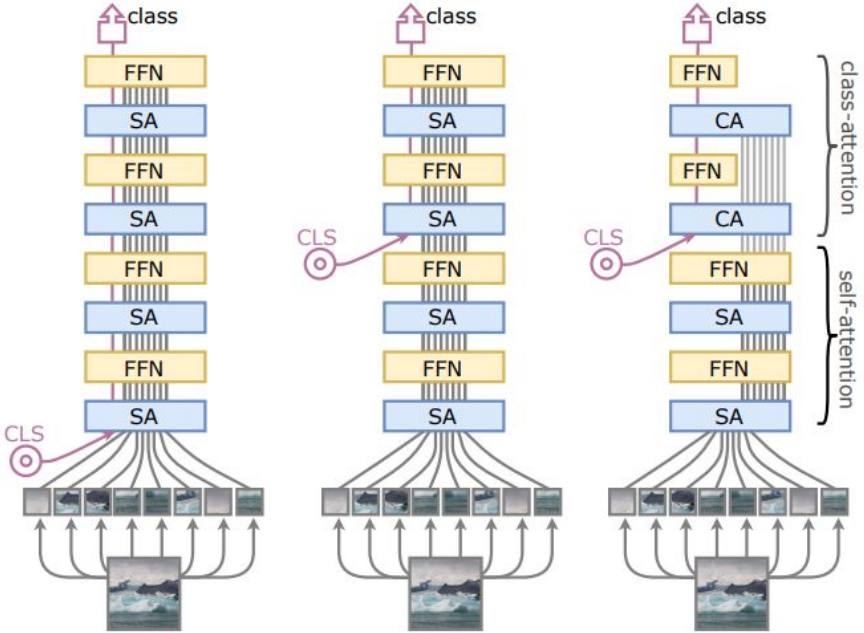
- Training data-efficient image transformers & distillation through attention, ICML 2021



DeiT-Ti	5M	224^2	2536.5	72.2	80.1	60.4
DeiT-S	22M	224^2	940.4	79.8	85.7	68.5
DeiT-B	86M	224^2	292.3	81.8	86.7	71.5
DeiT-B↑384	86M	384^2	85.9	83.1	87.7	72.4
DeiT-Ti⚗	6M	224^2	2529.5	74.5	82.1	62.9
DeiT-S⚗	22M	224^2	936.2	81.2	86.8	70.0
DeiT-B⚗	87M	224^2	290.9	83.4	88.3	73.2
DeiT-Ti⚗ / 1000 epochs	6M	224^2	2529.5	76.6	83.9	65.4
DeiT-S⚗ / 1000 epochs	22M	224^2	936.2	82.6	87.8	71.7
DeiT-B⚗ / 1000 epochs	87M	224^2	290.9	84.2	88.7	73.9

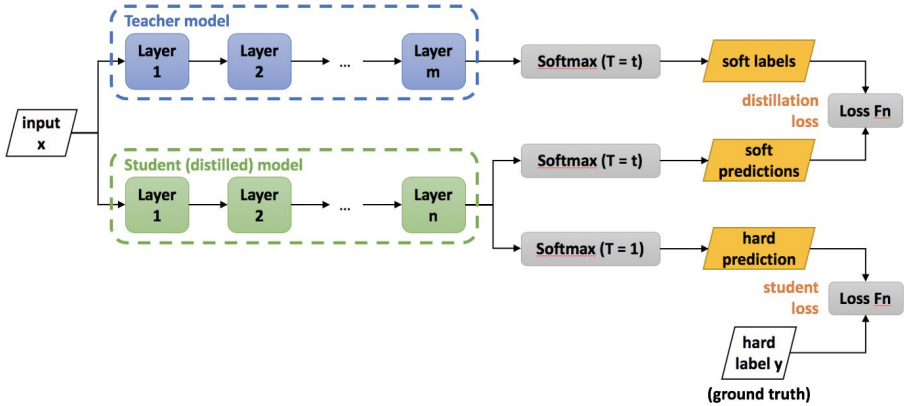
CaiT

- Going deeper with Image Transformers, ICCV 2021



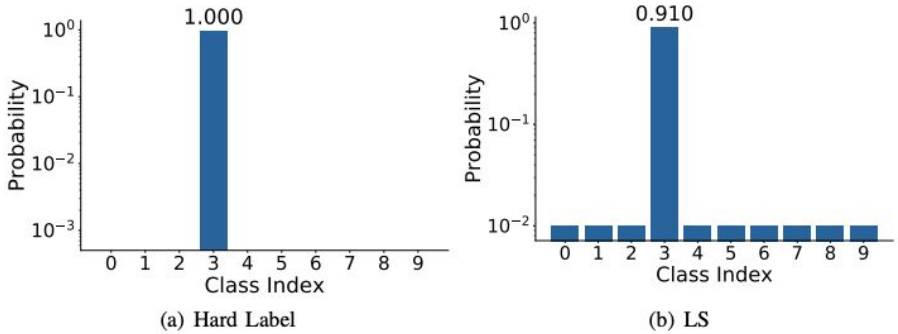
Knowledge distillation and label smoothing

Knowledge distillation (KD)



https://nervanasystems.github.io/distiller/knowledge_distillation.html

Label smoothing (LS)



https://arxiv.org/pdf/2011.12562.pdf

Knowledge distillation and label smoothing

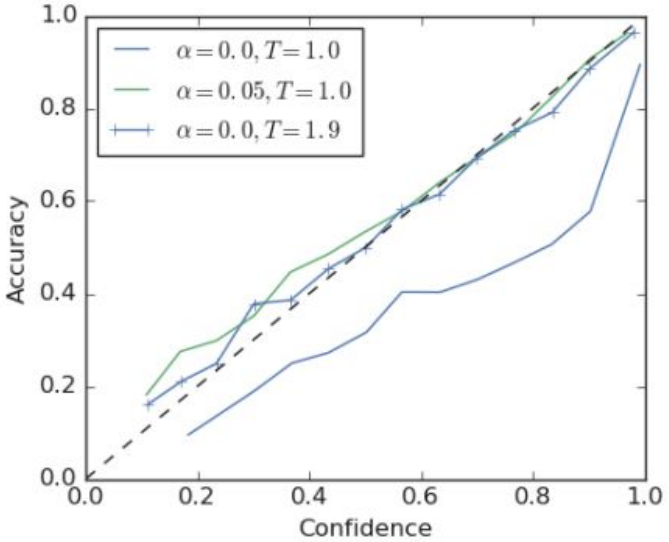
- Overconfidence

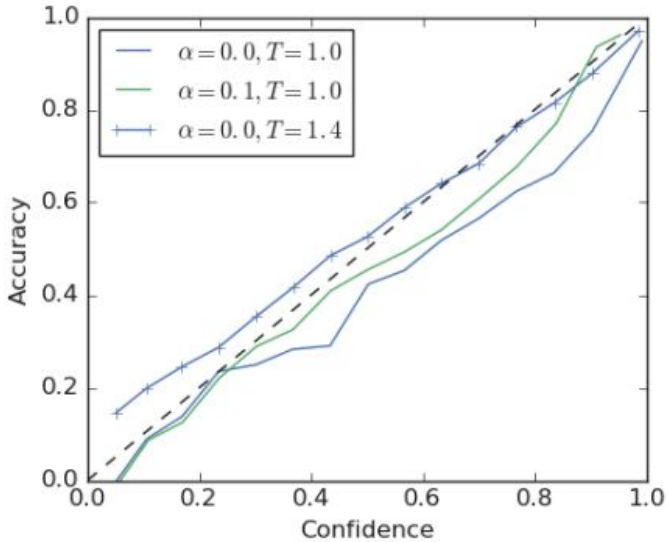
Parsed an image of myself through the animal network and it's 98% confident I'm a dog.

https://jramkiss.github.io/2020/07/29/overconfident-nn/

Knowledge distillation and label smoothing

- To overcome overconfidence

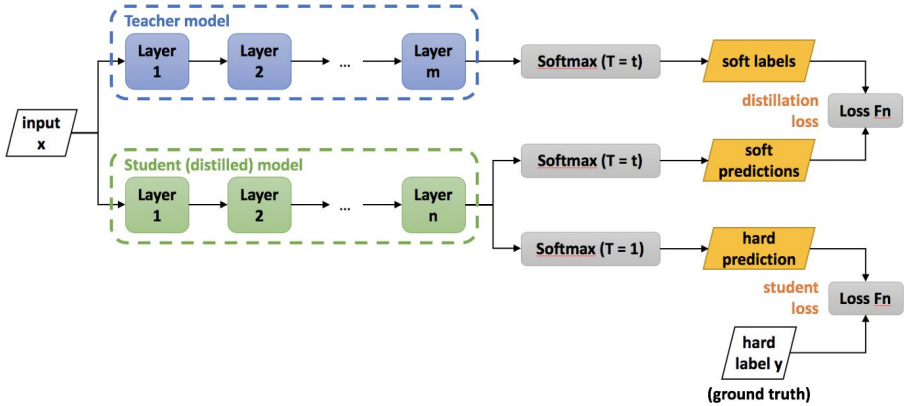


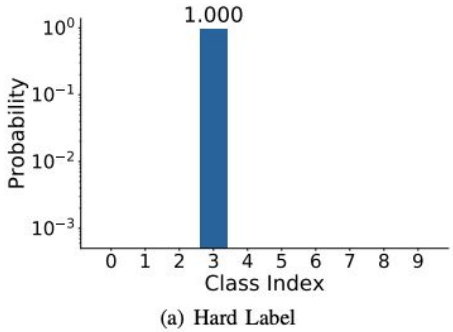


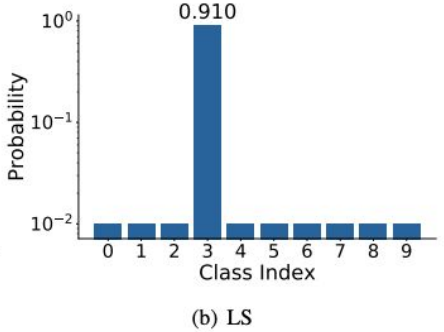
https://proceedings.neurips.cc/paper/2019/file/f1748d6b0fd9d439f71450117eba2725-Paper.pdf

Knowledge distillation and label smoothing

- CNN learns the texture more than shape







https://nervanasystems.github.io/distiller/knowledge_distillation.html

https://arxiv.org/pdf/2011.12562.pdf

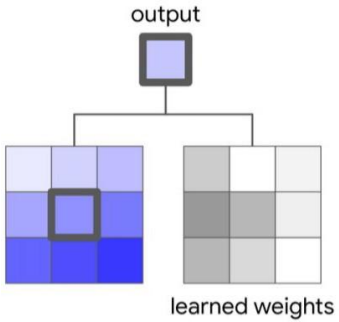
Transformers for vision?

- “LSTM → Transformer” ~ “ConvNet → ??? ”
- Issue with self-attention for vision: computation is quadratic in the input sequence length, quickly gets very expensive (with > few thousand tokens)
  - For ImageNet: 224x224 pixels → ~50,000 sequence length
  - Even worse for higher resolution and video

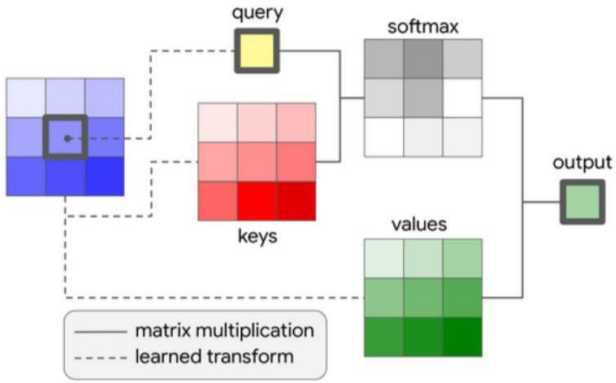
How can we deal with this quadratic complexity?

Local Self-Attention

Convolution



Local self-attention



Idea: Make self-attention local, use it instead of convs in a ResNet

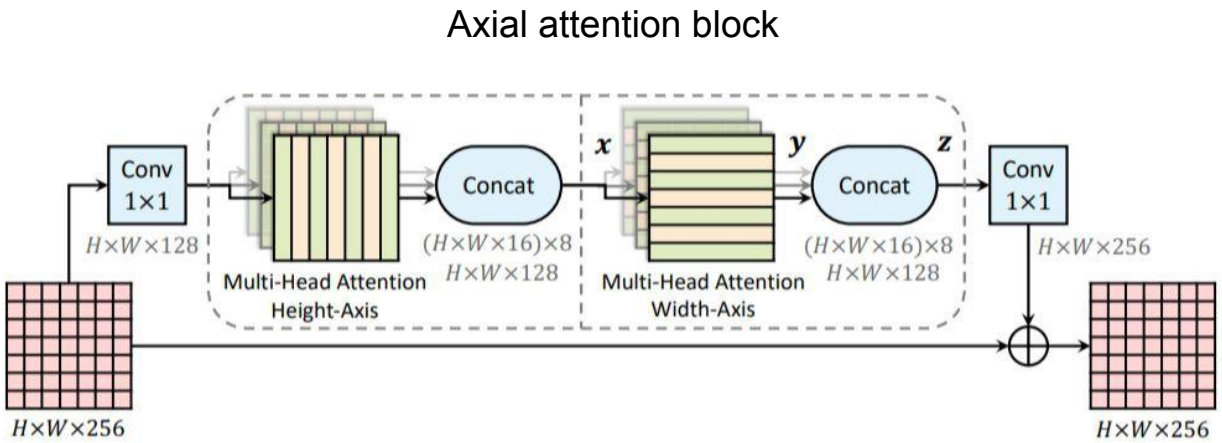
Hu et al., Local Relation Networks for Image Recognition, ICCV 2019

Ramachandran et al., Stand-Alone Self-Attention in Vision Models, NeurIPS 2019

Zhao et al., Exploring Self-attention for Image Recognition, CVPR 2020

Figures from Ramachandran et al.

Axial Self-Attention



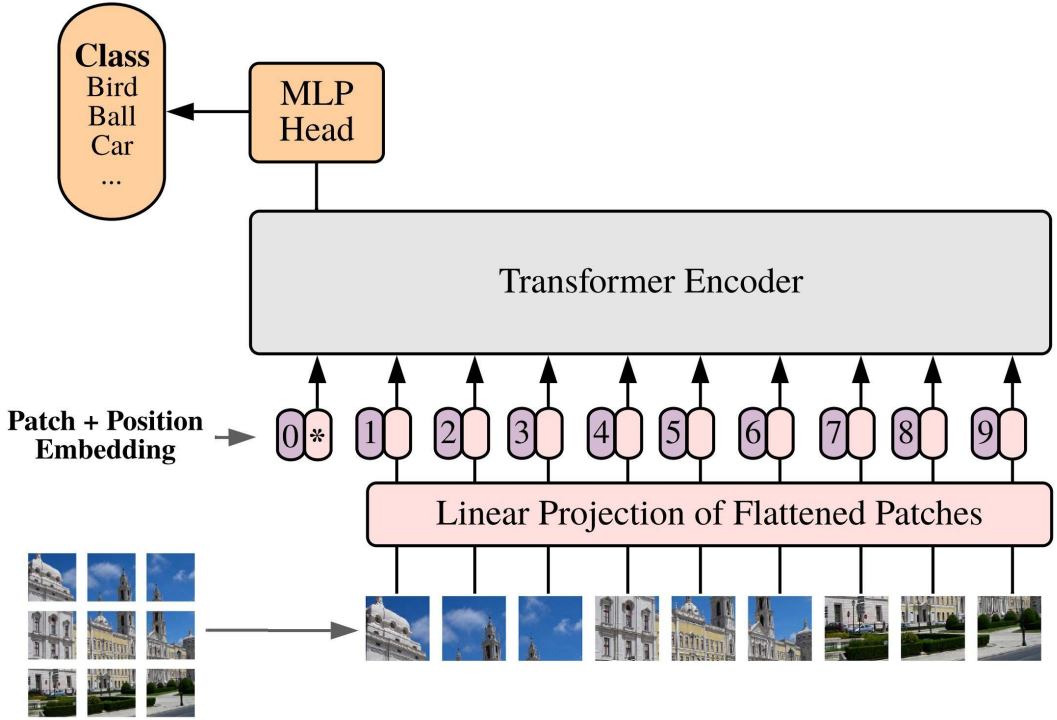
Idea: Make self-attention 1D (a.k.a. axial), use it instead of convs

Figure from Wang et al.

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

Vision Transformer (ViT)

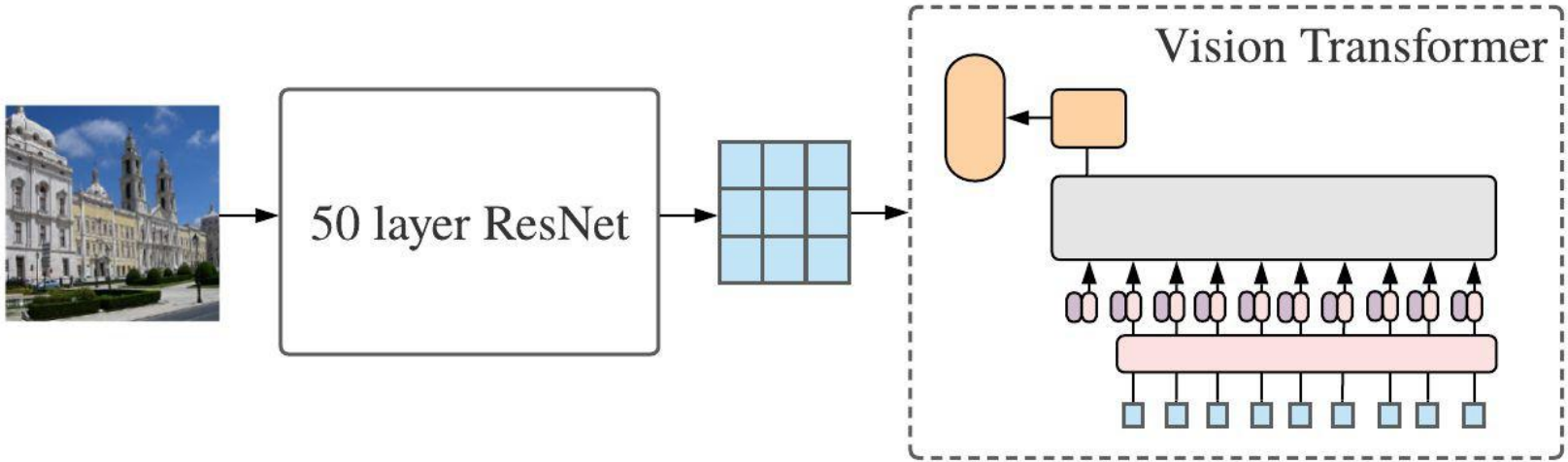
Idea: Take a transformer and apply it directly to image patches



Cordonnier et al., *On the Relationship between Self-Attention and Convolutional Layers*, ICLR 2020

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

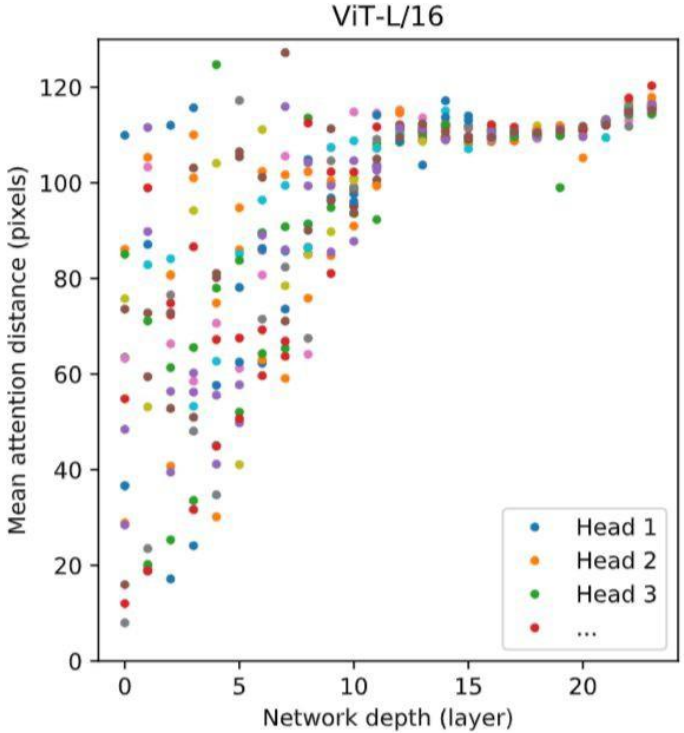
ResNet-ViT Hybrid



Bichen Wu et al. *Visual Transformers: Token-based Image Representation and Processing for Computer Vision*, arXiv 2020

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

Analysis: “Receptive Field Size”



Conclusion: Initial layers are partially local, deeper layers are global

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

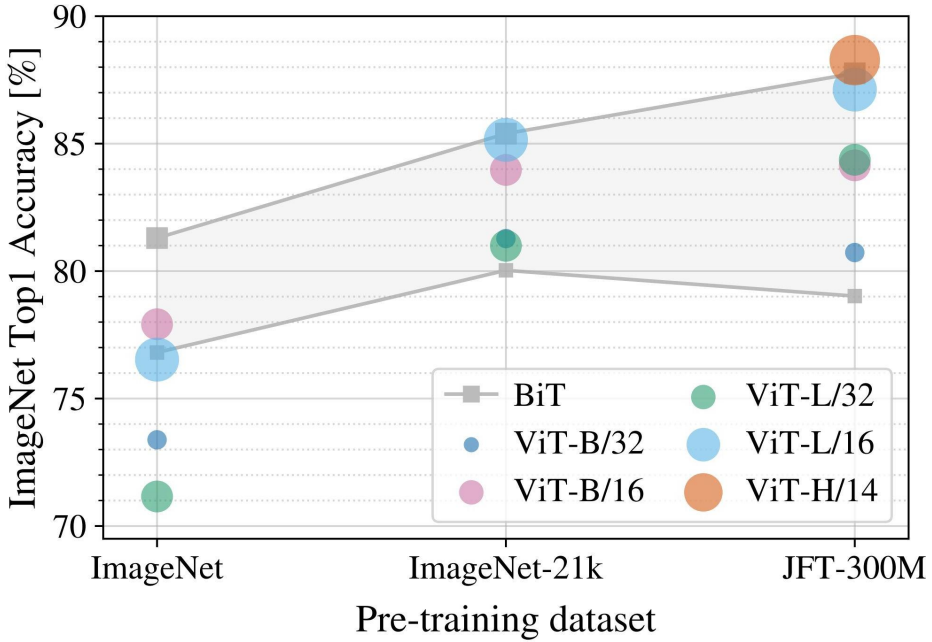
Scaling with Data

Key
ViT = Vision Transformer
BiT = Big Transfer (~ResNet)

ViT overfits on ImageNet, but shines on larger datasets

* with heavy regularization ViT has been shown to also work on ImageNet (Touvron et al.)

** training ViT on ImageNet with the sharpness-aware minimizer (SAM) also works very well (Chen et al.)

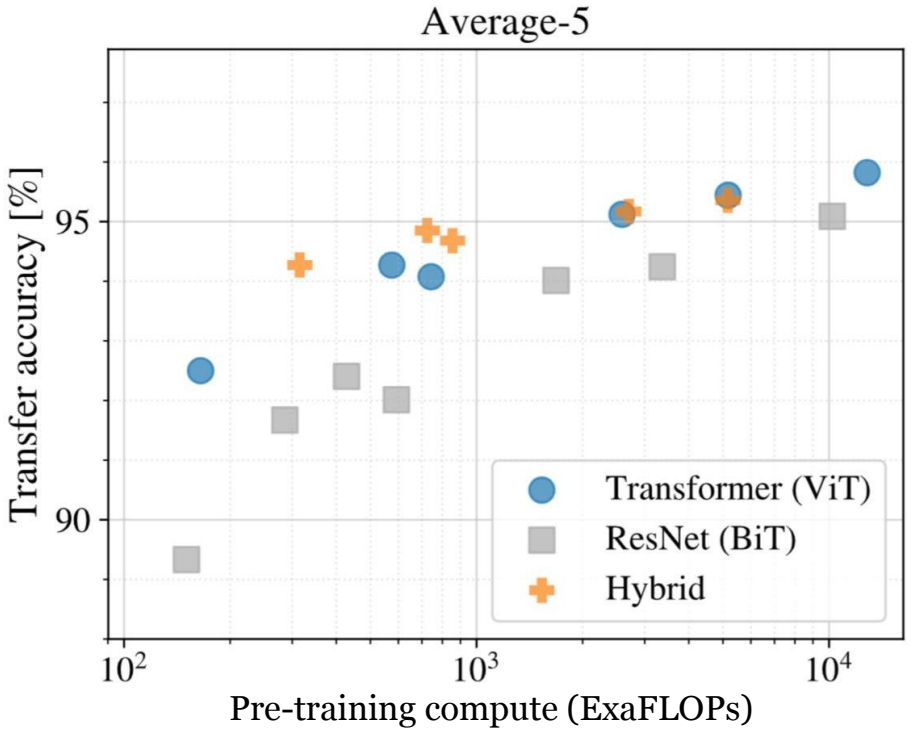


Dosovitskiy et al., *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*, ICLR 2021
Xiangning Chen et al., *When Vision Transformers Outperform ResNets without Pretraining or Strong Data Augmentations*, arXiv 2021
Touvron et al., *Training data-efficient image transformers & distillation through attention*, arXiv 2020

Scaling with Compute

Given sufficient data, ViT gives good performance/FLOP

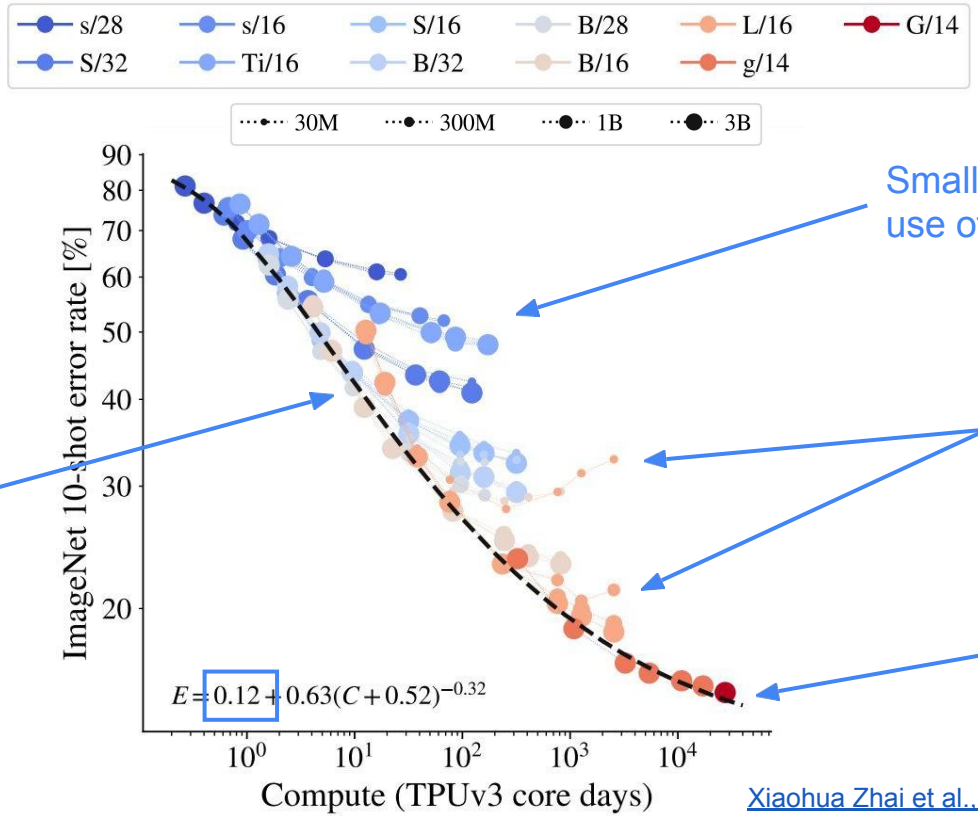
Hybrids yield benefits only for smaller models



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

Scaling Laws

How many images do you need for a big model & vice-versa?



Small models can't make use of >30M examples

Even 300M examples are insufficient for large models

Power-law behaviour $E=aC^{-b}$

Saturates before y=0

Xiaohua Zhai et al., *Scaling Vision Transformers*, arXiv 2021

Shape / Texture bias

- CNN learns the texture more than shape

(a) Texture image

81.4%	**Indian elephant**
10.3%	indri
8.2%	black swan

(b) Content image

71.1%	**tabby cat**
17.3%	grey fox
3.3%	Siamese cat

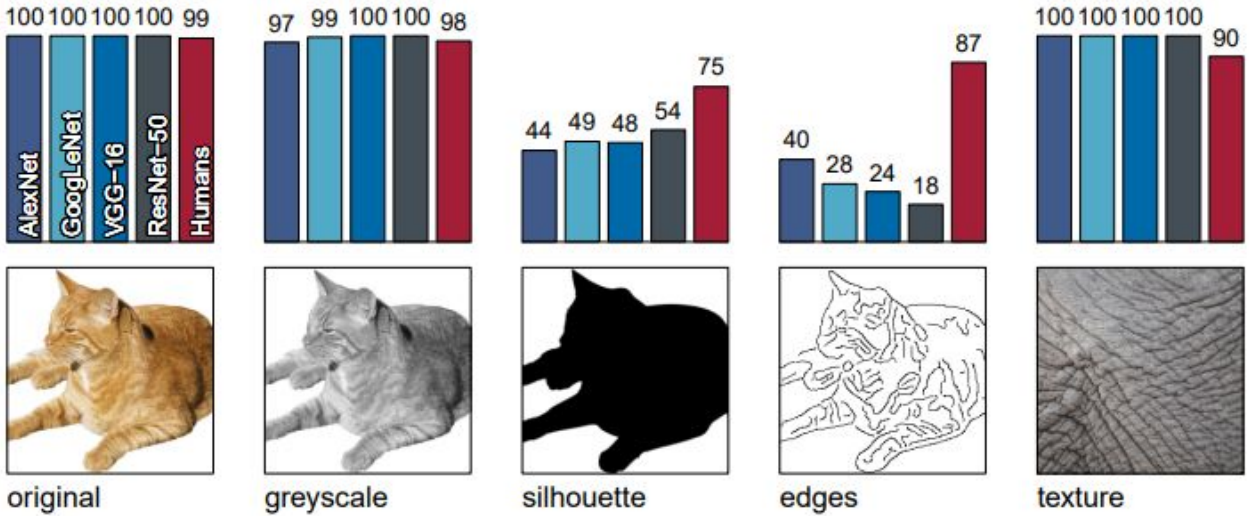
(c) Texture-shape cue conflict

63.9%	**Indian elephant**
26.4%	indri
9.6%	black swan

Geirhos et al., IMAGENET-TRAINED CNNS ARE BIASED TOWARDS TEXTURE; INCREASING SHAPE BIAS IMPROVES ACCURACY AND ROBUSTNESS (ICLR 2019)

Shape / Texture bias

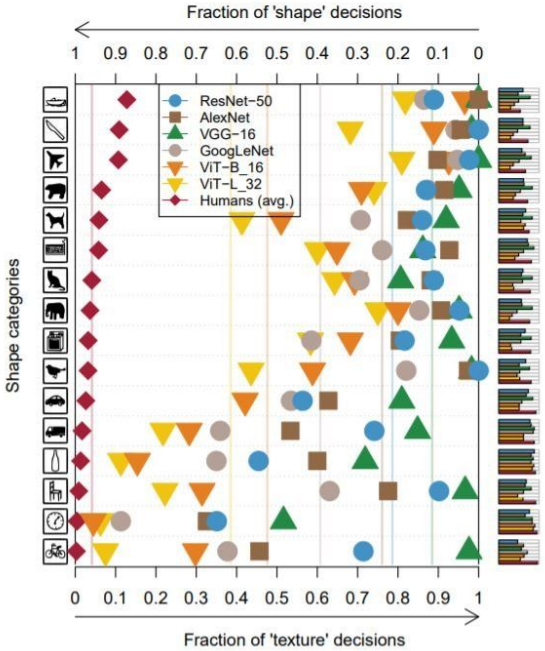
- CNN learns the texture more than shape

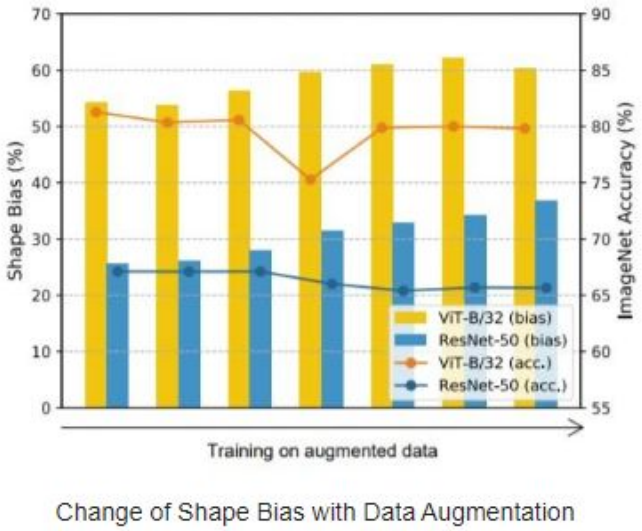


Geirhos et al., IMAGENET-TRAINED CNNS ARE BIASED TOWARDS TEXTURE; INCREASING SHAPE BIAS IMPROVES ACCURACY AND ROBUSTNESS (ICLR 2019)

Shape / Texture bias

- ViT learns a little more about shape information.



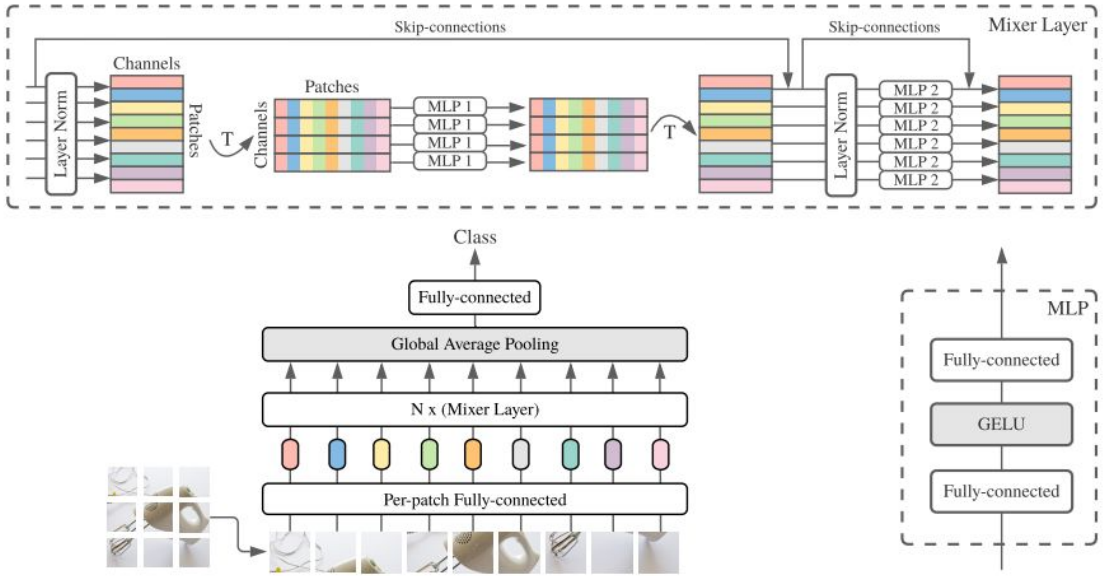


Change of Shape Bias with Data Augmentation

https://ai-scholar.tech/en/articles/transformer/human_vision

MLP-Mixer

- MLP-Mixer: An all-MLP Architecture for Vision, ICLR 2021



https://openreview.net/pdf?id=EI2KOXKdnP

MLP-Mixer

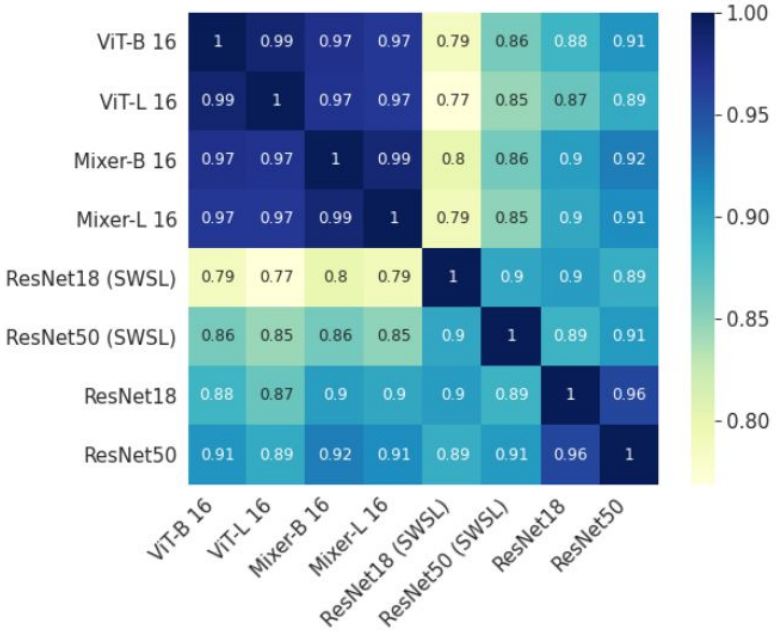
- MLP-Mixer: An all-MLP Architecture for Vision, ICLR 2021

	ImNet top-1	ReaL top-1	Avg 5 top-1	VTAB-1k 19 tasks	Throughput img/sec/core	TPUv3 core-days
Pre-trained on ImageNet-21k (public)						
HaloNet [51]	85.8	—	—	—	120	0.10k
Mixer-L/16	84.15	87.86	93.91	74.95	105	0.41k
ViT-L/16 [14]	85.30	88.62	94.39	72.72	32	0.18k
BiT-R152x4 [22]	85.39	—	94.04	70.64	26	0.94k
Pre-trained on JFT-300M (proprietary)						
NFNet-F4+ [7]	89.2	—	—	—	46	1.86k
Mixer-H/14	87.94	90.18	95.71	75.33	40	1.01k
BiT-R152x4 [22]	87.54	90.54	95.33	76.29	26	9.90k
ViT-H/14 [14]	88.55	90.72	95.97	77.63	15	2.30k
Pre-trained on unlabelled or weakly labelled data (proprietary)						
MPL [35]	90.0	91.12	—	—	—	20.48k
ALIGN [21]	88.64	—	—	79.99	15	14.82k

https://openreview.net/pdf?id=EI2KOXKdnP

CNN, ViT, and MLP-Mixer??

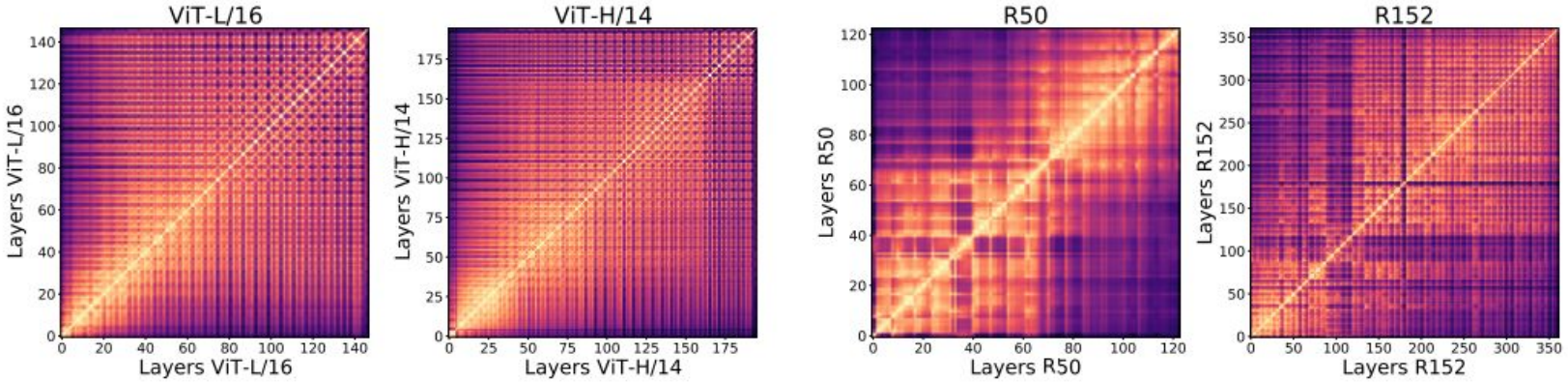
- Similarity matrix



https://arxiv.org/pdf/2110.02797.pdf

CNN, ViT, and MLP-Mixer??

- Do Vision Transformers See Like Convolutional Neural Networks?, Neurips 2021



CNN, ViT, and MLP-Mixer??

- Do Vision Transformers See Like Convolutional Neural Networks?, Neurips 2021