Deep learning

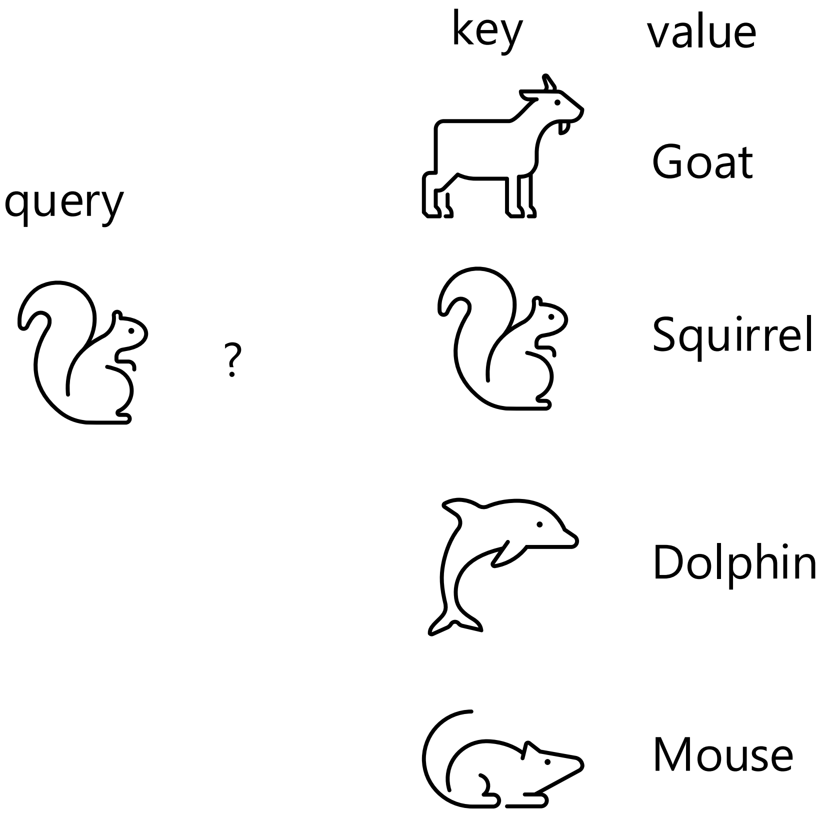
Transformers

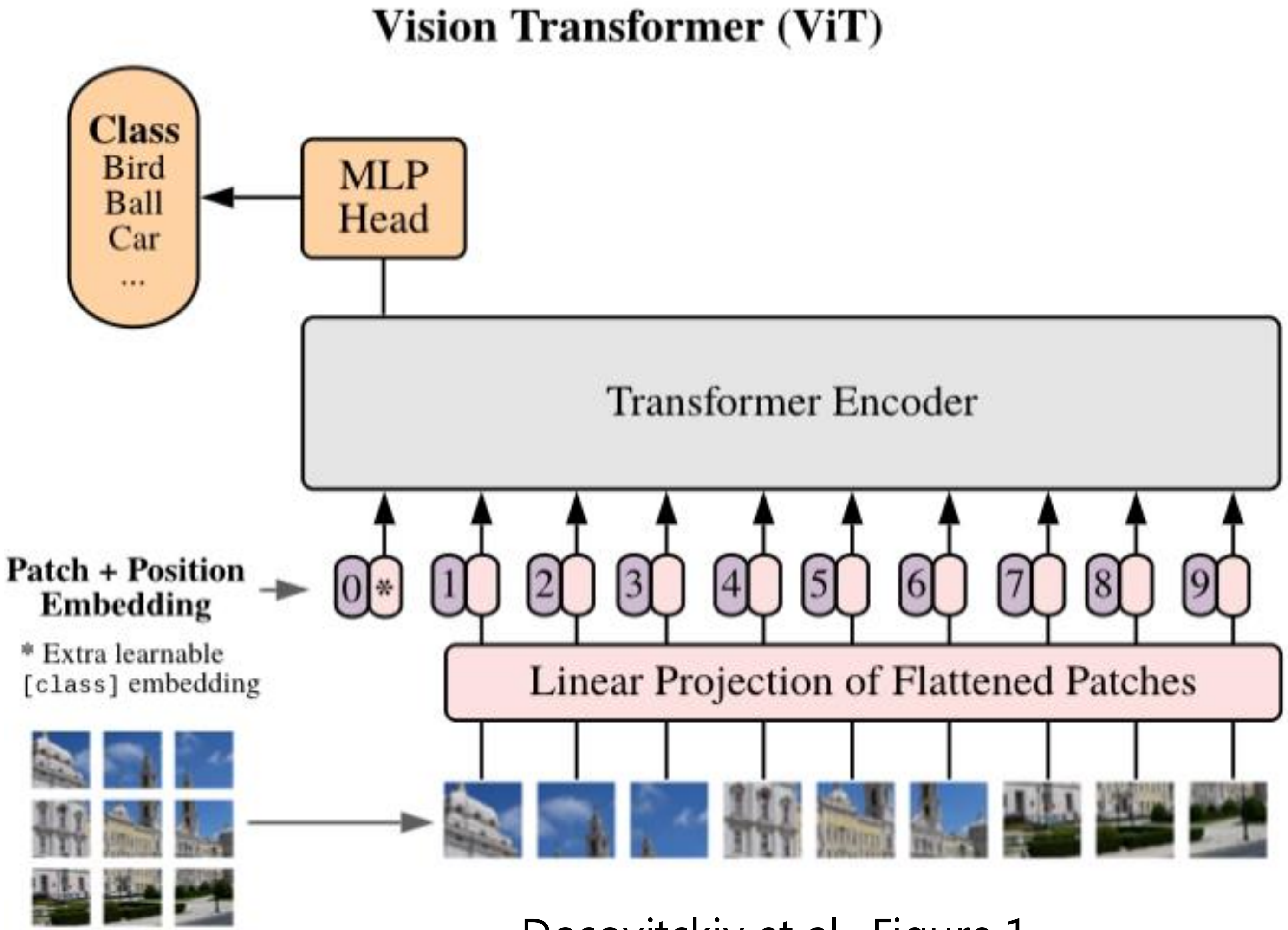
2024/12/18

Jon Sporring,
Department of Computer Science

UNIVERSITY OF COPENHAGEN

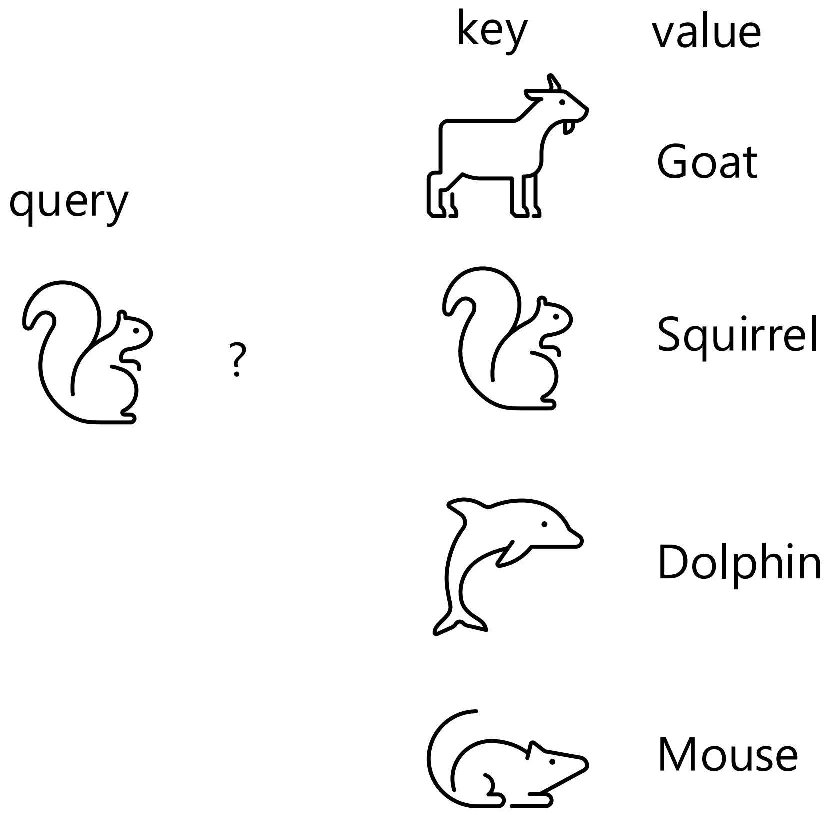
Transformer networks – Vision Transformer ViT: Vaswani ea (2017) & Dosovitskiy ea (2020)

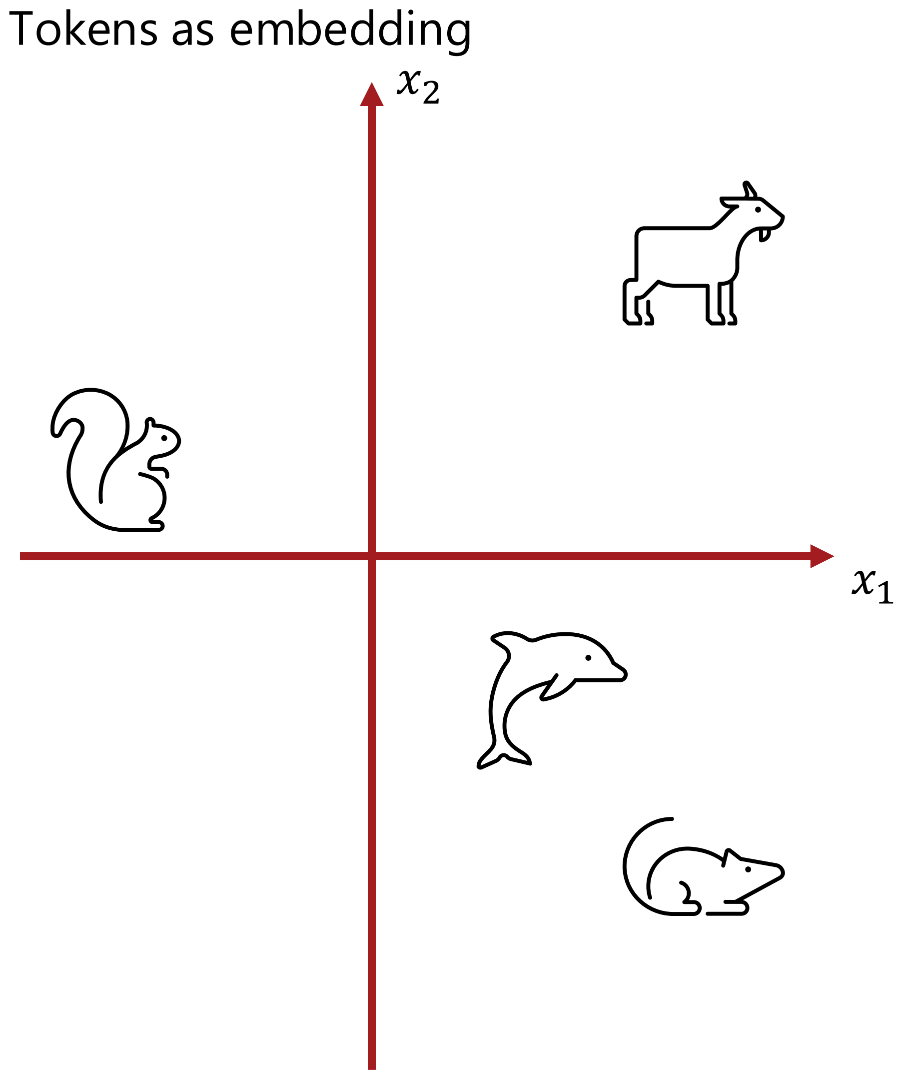




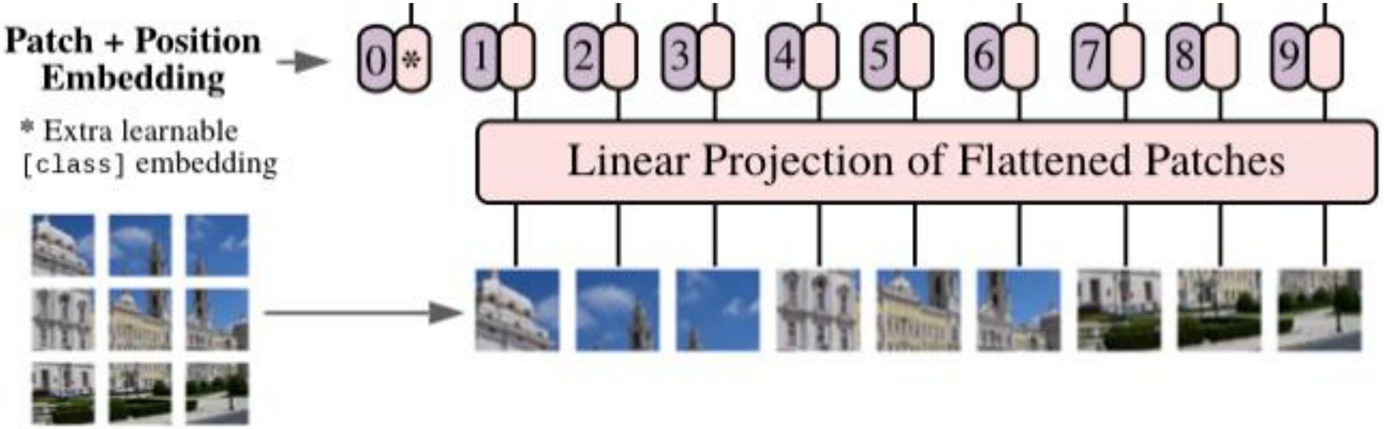
Dosovitskiy et al., Figure 1

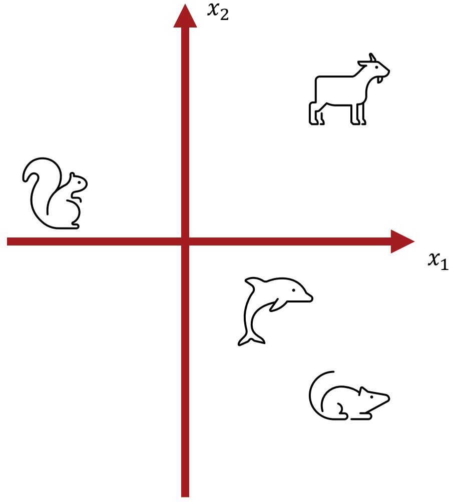
Tokens: Cut into patches, flatten and linearly embed in lower dimensional space





Embeddings++





$I \in \mathbb{R}^{H\times W\times C}, I_p \in \mathbb{R}^{M\times M\times C}$, Flatten: $\mathbb{R}^{M\times M\times C} \to \mathbb{R}^{M^2C}$, Token: $\mathbb{R}^{M^2C} \to \mathbb{R}^D$

Convention: row vectors

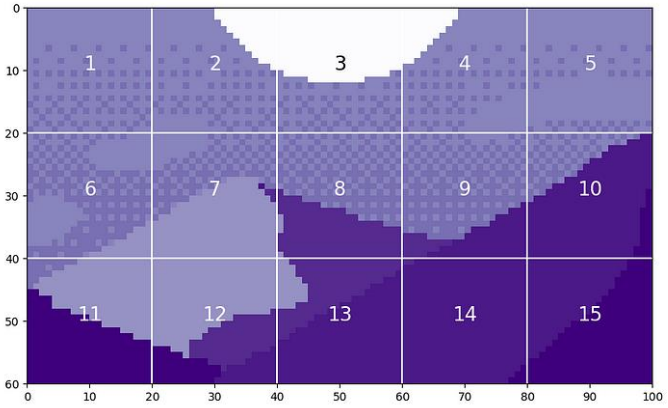
$$\boldsymbol{x}_j = \text{Flatten}\left(I_j\right) \in \mathbb{R}^{M^2C}$$

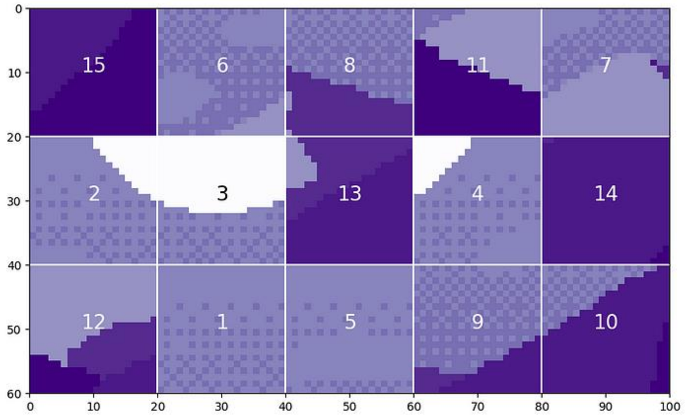
$$\boldsymbol{t}_j = \text{Token}\left(\boldsymbol{x}_j\right) = \boldsymbol{x}_j\mathbf{E} \in \mathbb{R}^D$$

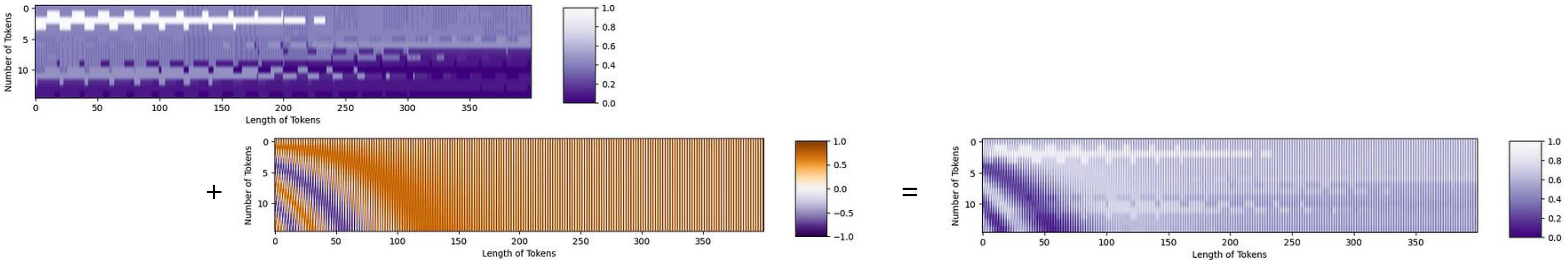
$$\boldsymbol{z}^0 = [\boldsymbol{x}_{\text{class}}; \boldsymbol{t}_1; \boldsymbol{t}_2; \dots; \boldsymbol{t}_N] + \text{PositionEmbedding}\,() \in \mathbb{R}^{(N+1)\times D}$$

Positional Embedding: Vision depends on position

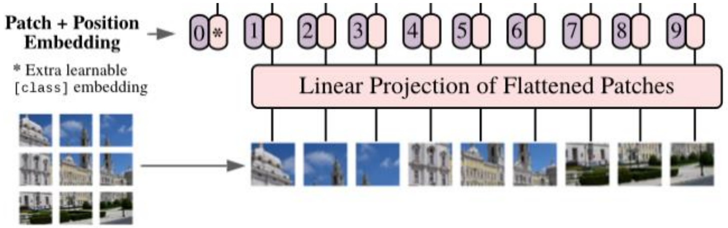
Yuan et al (2021). *Tokens-to-Token ViT: Training Vision Transformers from Scratch on ImageNet.*
Figures from
https://towardsdatascience.com/position-embeddings-for-vision-transformers-explained-a6f9add341d5

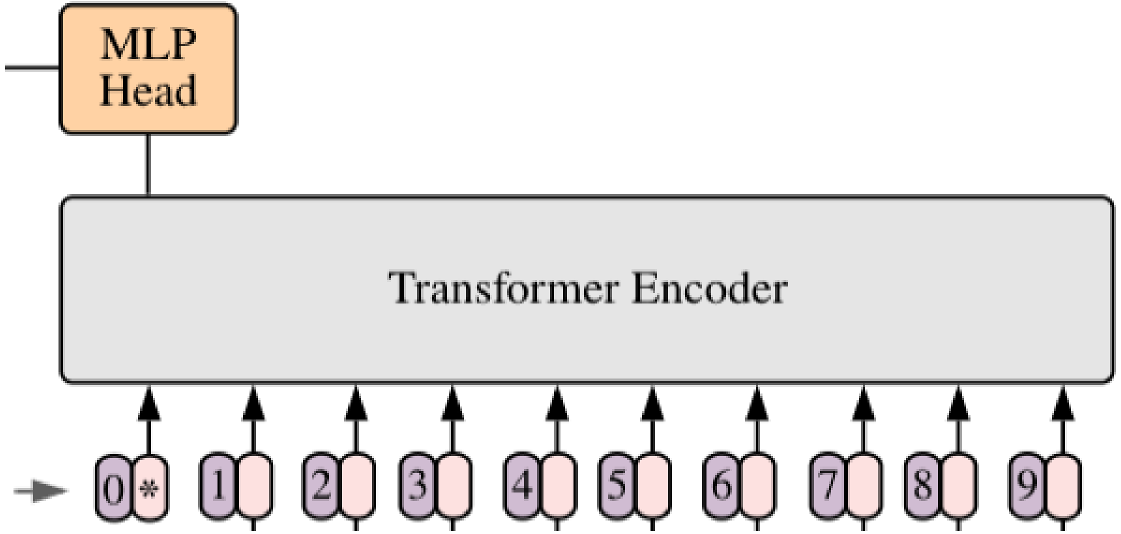






Encoding





A sequence of attention layers: $l = 1 \ldots L$

$$\boldsymbol{z}_0 = [\boldsymbol{x}_{\text{class}}; \boldsymbol{t}_1; \boldsymbol{t}_2; \ldots; \boldsymbol{t}_N] \in \mathbb{R}^{(N+1)\times D}$$

$$\boldsymbol{z}'_l = \text{MSA}\big(\text{LN}(\boldsymbol{z}'_{l-1})\big) + \boldsymbol{z}'_{l-1} \in \mathbb{R}^{(N+1)\times D}$$

$$\boldsymbol{z}_l = \text{MLP}_2\big(\text{LN}(\boldsymbol{z}_l)\big) + \boldsymbol{z}'_l \in \mathbb{R}^{(N+1)\times D}$$

MSA: Multiheaded self-attention

MLP_n: n-layer Multilayer perceptron

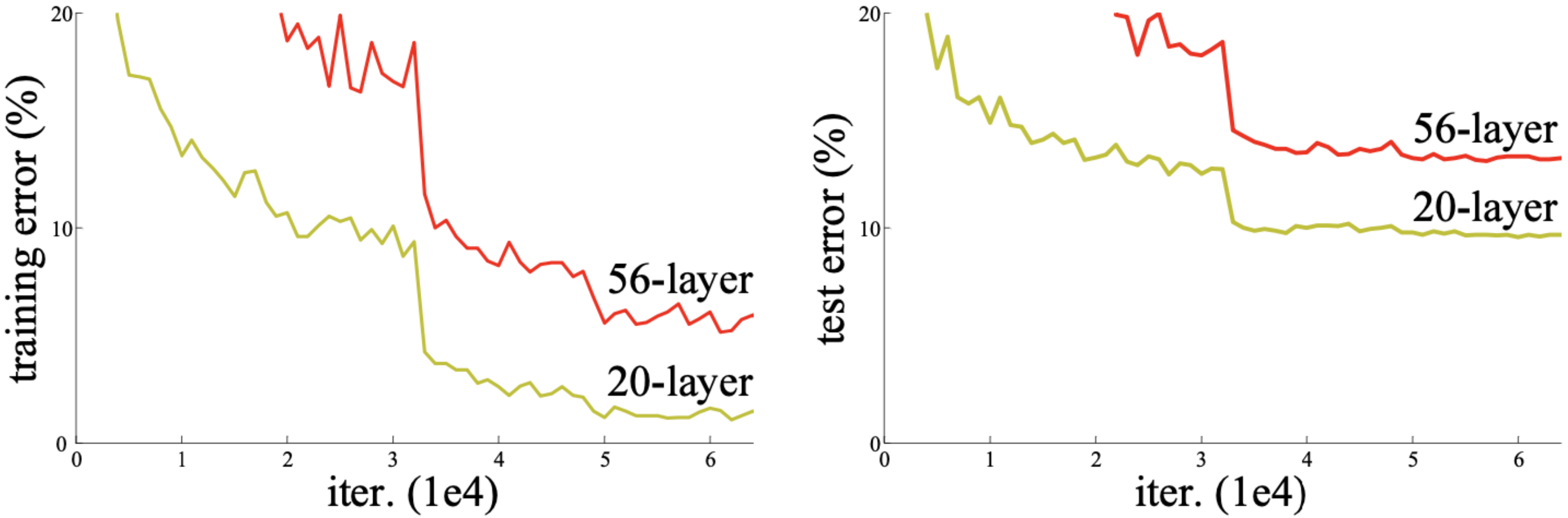
LN: Layer normalization

Residual learning

Residual learning for optimizing deeper networks

He et al, “Deep Residual Learning for Image Recognition, CVPR 2016

Observation: Deeper networks = poorer convergence



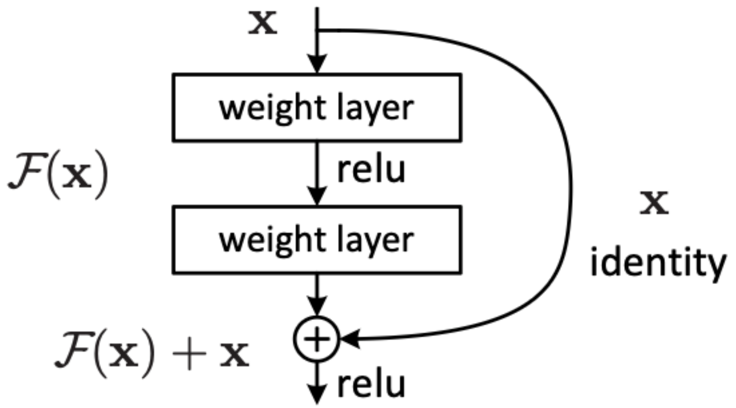
He et al. Figure 1

Residual learning for optimizing deeper networks

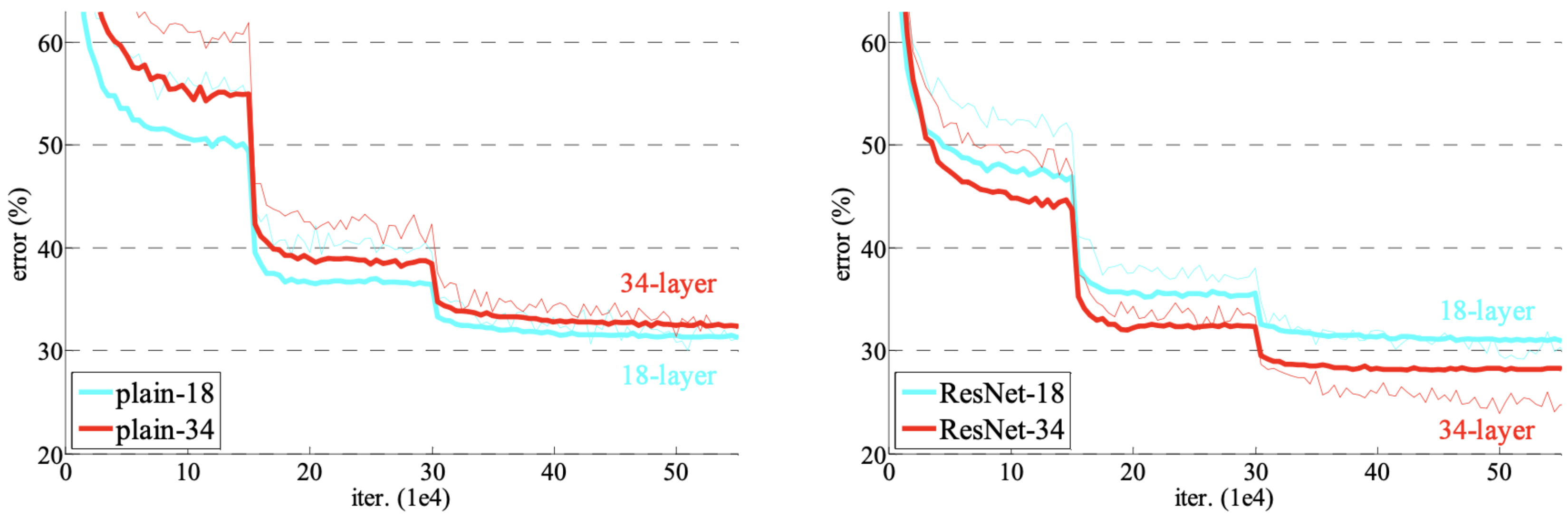
He et al, “Deep Residual Learning for Image Recognition, CVPR 2016

Observation: Deeper networks = poorer convergence
Solution: Train on residuals

$$\mathcal{F}(\mathbf{x}) := \mathcal{H}(\mathbf{x}) - \mathbf{x}.$$



He et al. Figure 2



He et al. Figure 4

Multilayer perceptron (MLP) and Layer Normalization (LN)

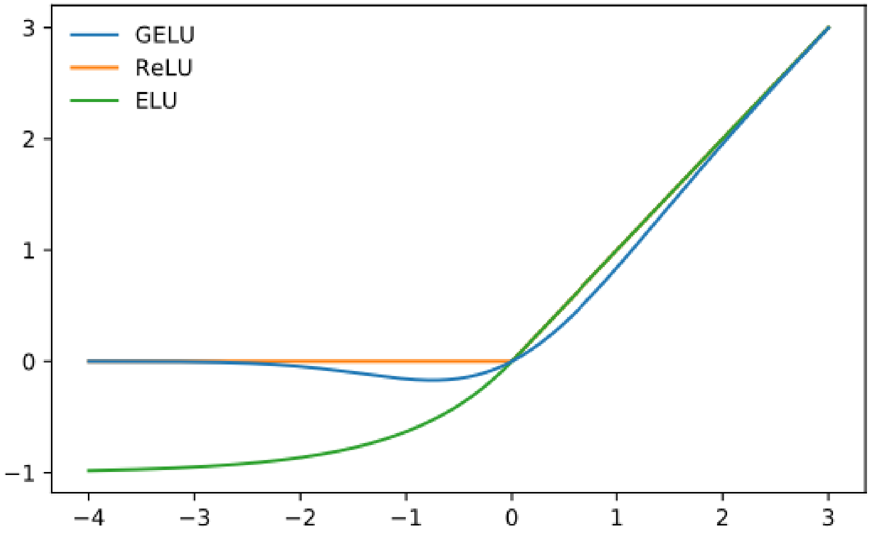
Ba, Kiros, and Hinton, Layer normalization, 2016

$$\mathrm{LN}(\boldsymbol{z}) = \frac{\boldsymbol{z} - \bar{\boldsymbol{z}}}{\sqrt{\mathrm{Var}(\boldsymbol{z}) + \epsilon}}$$

A sequence of attention layers: $l = 1 \ldots L$

$$\boldsymbol{z}_0 = [\boldsymbol{x}_{\text{class}};\ \boldsymbol{t}_1;\ \boldsymbol{t}_2; \ldots, ; \boldsymbol{t}_N] \in \mathbb{R}^{(N+1)\times D}$$
$$\boldsymbol{z}'_l = \mathrm{MSA}\big(\mathrm{LN}(\boldsymbol{z}'_{l-1})\big) + \boldsymbol{z}'_{l-1} \in \mathbb{R}^{(N+1)\times D}$$
$$\boldsymbol{z}_l = \mathrm{MLP}\big(\mathrm{LN}(\boldsymbol{z}'_l)\big) + \boldsymbol{z}'_l \in \mathbb{R}^{(N+1)\times D}$$

MLP: Single layer with two GELU
GELU: Hendryks & Gimpel, Gaussian Error Linear Units (GELU)

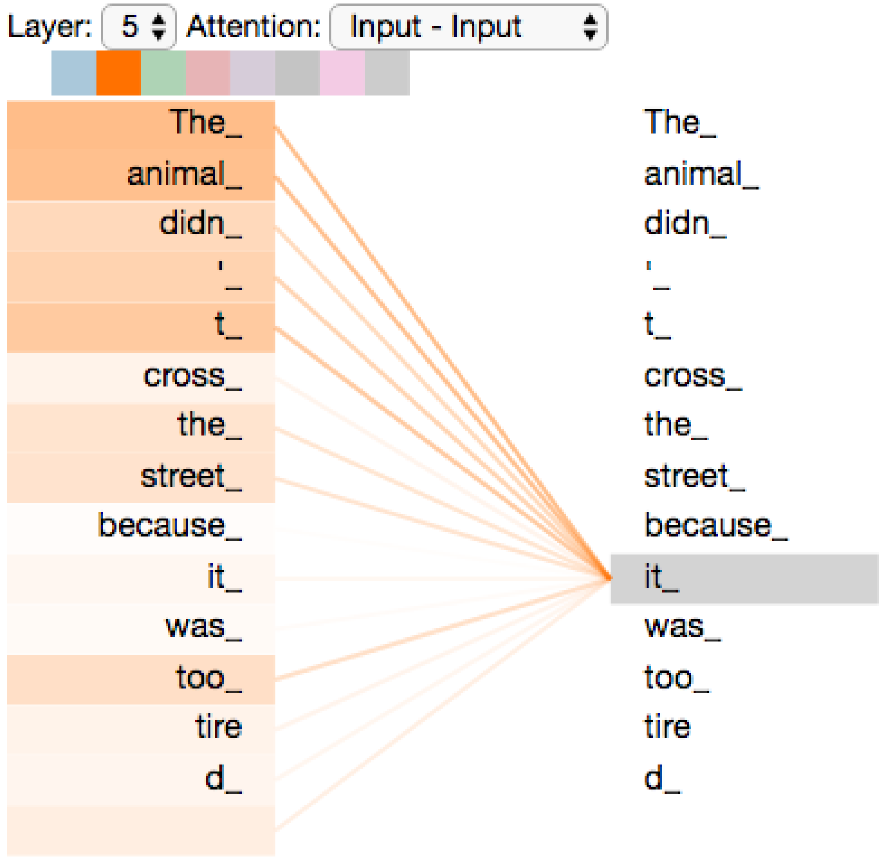


Hendryks & Gimpel, Figure 1

$$\mathrm{GELU}(x) = \frac{x}{2}\left[1 + \mathrm{erf}\left(\frac{x}{\sqrt{2}}\right)\right]$$

$$\simeq x/2(1 + \tanh\left((x + 0.044715x^3)\sqrt{\frac{2}{\pi}}\right)$$

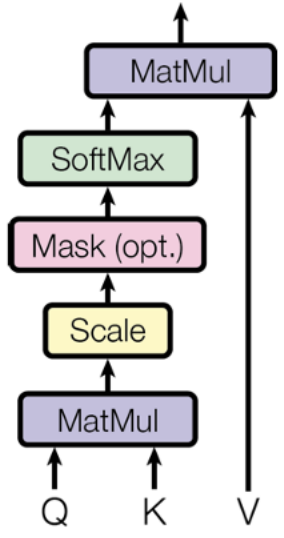
Attention:
The animal didn’t cross the street because it was too tired



Self-attention head

$I \in \mathbb{R}^{H\times W\times C}, I_p \in \mathbb{R}^{M\times M\times C}$, Flatten: $\mathbb{R}^{M\times M\times C} \to \mathbb{R}^{M^2C}$, Token: $\mathbb{R}^{M^2C} \to \mathbb{R}^{D}$

$\boldsymbol{z} \in \mathbb{R}^{(N+1)\times D}$



Vaswani, Fig 2

Query $\boldsymbol{q}_j = \boldsymbol{z}_{j*}U_q \in \mathbb{R}^{D_h}$

Key $\boldsymbol{k}_j = \boldsymbol{z}_{j*}U_k \in \mathbb{R}^{D_h}$

Value $\boldsymbol{v}_j = \boldsymbol{z}_{j*}U_v \in \mathbb{R}^{D_h}$

$U_* \in \mathbb{R}^{D\times D_h}$ learnable

$$\mathbf{a}_{ij} = \mathrm{softmax}\left(\frac{\boldsymbol{q}_i \cdot \boldsymbol{k}_j}{\sqrt{D_h}}\right)$$

$$\mathrm{softmax}(\{s_j\}) = \left\{\frac{e^{s_j}}{\sum_{j=1}^{n} e^{s_j}}\right\}$$

$$\mathrm{SA}(\boldsymbol{z}_{i*}) = \sum_j \mathbf{a}_{ij}\boldsymbol{v}_j \in \mathbb{R}^{D_h}$$

Query how one embedding ($\boldsymbol{q}$) matches keys ($\boldsymbol{k}$), which are all other embeddings.

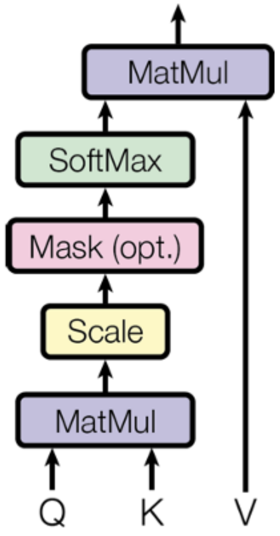
Softmax select the best maching querys to their cosine difference

Output is now computed as a weigted sum of values ($\boldsymbol{v}$).

Multihead self-attention

$I \in \mathbb{R}^{H\times W\times C}, I_p \in \mathbb{R}^{M\times M\times C}$, Flatten: $\mathbb{R}^{M\times M\times C} \to \mathbb{R}^{M^2C}$, Token: $\mathbb{R}^{M^2C} \to \mathbb{R}^{D}$

$\boldsymbol{z} \in \mathbb{R}^{(N+1)\times D}$



Vaswani, Fig 2

Query	$\boldsymbol{Q} = \{\boldsymbol{q}_j\} \in \mathbb{R}^{(N+1)\times D_h}$	
Key	$\boldsymbol{K} = \{\boldsymbol{k}_j\} \in \mathbb{R}^{(N+1)\times D_h}$	$U_* \in \mathbb{R}^{D\times D_h}$ learnable
Value	$\boldsymbol{V} = \{\boldsymbol{v}_j\} \in \mathbb{R}^{(N+1)\times D_h}$	

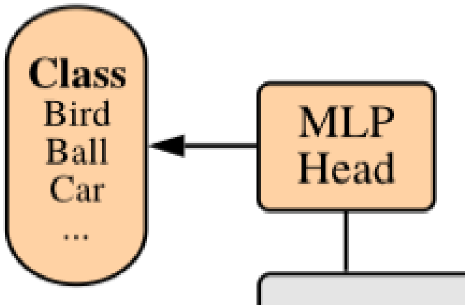
$$\boldsymbol{A} = \mathrm{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{D_h}}\right) \in \mathbb{R}^{(N+1)\times(N+1)}$$

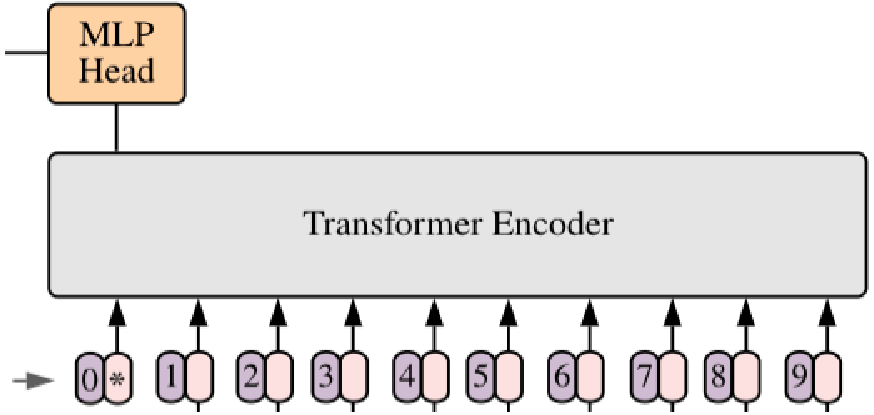
$$\mathrm{SA}(\boldsymbol{z}) = \boldsymbol{A}\boldsymbol{V} \in \mathbb{R}^{(N+1)\times D_h}$$

$$\boldsymbol{U}_{msa} \in \mathbb{R}^{kD_h\times D}$$

$$\mathrm{MSA}(\boldsymbol{z}) = [\mathrm{SA}_1(\boldsymbol{z}), \mathrm{SA}_2(\boldsymbol{z}), \dots, \mathrm{SA}_k(\boldsymbol{z})]\boldsymbol{U}_{msa} \in \mathbb{R}^{(N+1)\times D}$$

Encoding





A sequence of attention layers: $l = 1 \dots L$

$$\boldsymbol{z}_0 = [\boldsymbol{x}_{\text{class}};\ \boldsymbol{t}_1; \boldsymbol{t}_2; \dots; \boldsymbol{t}_N] \in \mathbb{R}^{2(N+1)\times D}$$

$$\boldsymbol{z}'_l = \text{MSA}\big(\text{LN}(\boldsymbol{z}'_{l-1})\big) + \boldsymbol{z}'_{l-1} \in \mathbb{R}^{2(N+1)\times D}$$

$$\boldsymbol{z}_l = \text{MLP}_2\big(\text{LN}(\boldsymbol{z}'_l)\big) + \boldsymbol{z}'_l \in \mathbb{R}^{2(N+1)\times D}$$

$$\boxed{\text{Class} = \text{MLP}_1(\boldsymbol{z}_L)}$$

Inductive bias [Dosoviskiy]

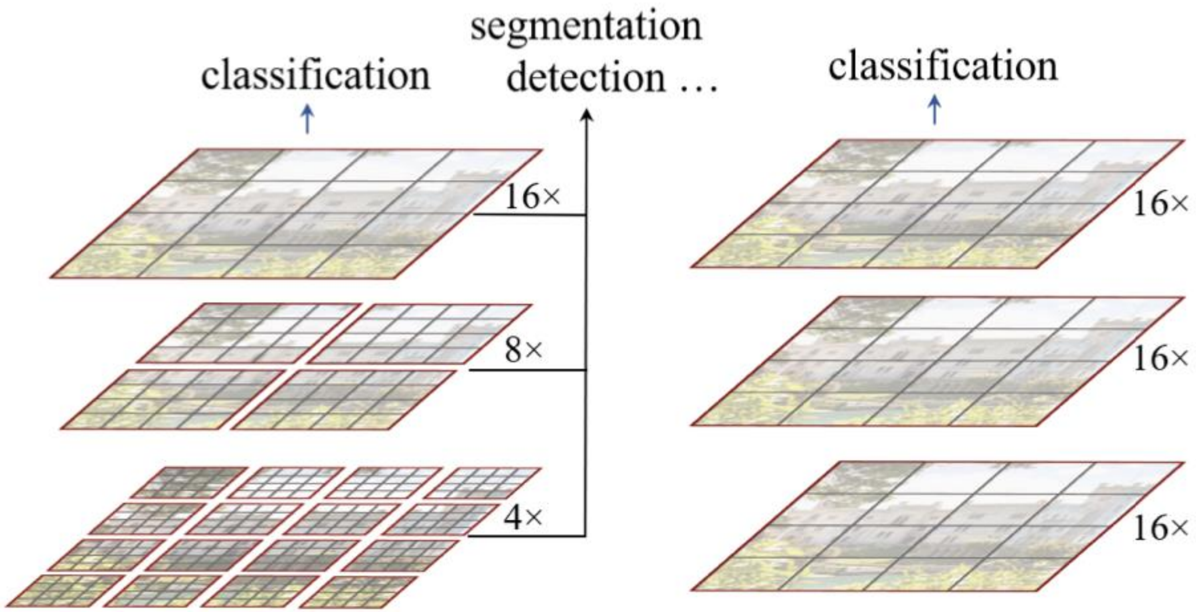
Inductive bias. We note that Vision Transformer has much less image-specific inductive bias than CNNs. In CNNs, locality, two-dimensional neighborhood structure, and translation equivariance are baked into each layer throughout the whole model. In ViT, only MLP layers are local and translationally equivariant, while the self-attention layers are global. The two-dimensional neighborhood

ImageNet: https://image-net.org/
14*10^6 images, 2*10^4 categories, 10^6 images with bounding boxes

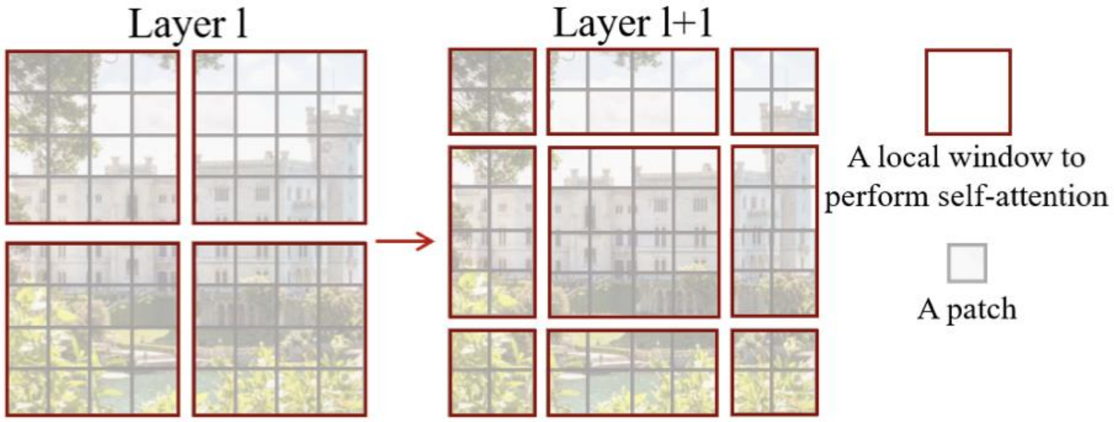
State-of-the-art 2024

Swin Transformer: Hierarchical Vision Transformer using Shifted Windows

xuan Wei†
o



(a) Swin Transformer (ours) (b) ViT

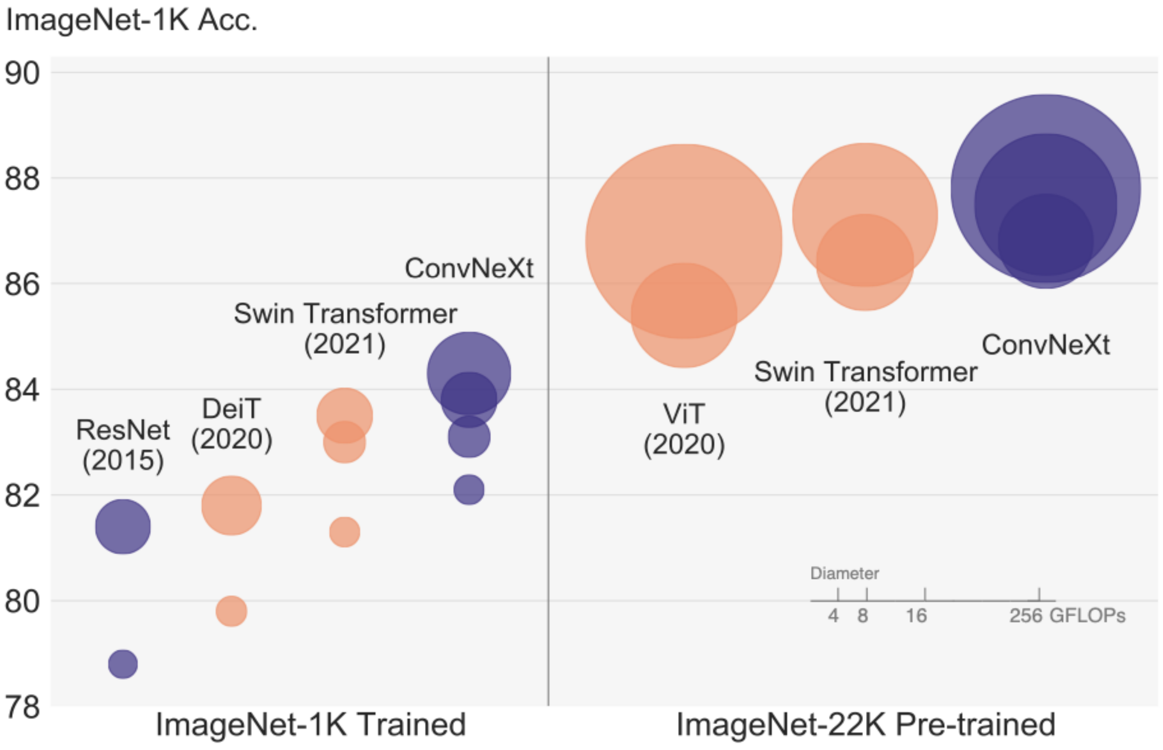


ConvNeXt

A ConvNet for the 2020s

Zhuang Liu[1,2*] Hanzi Mao[1] Chao-Yuan Wu[1] Christoph Feichtenhofer[1] Trevor Darrell[2] Saining Xie[1†]
[1]Facebook AI Research (FAIR) [2]UC Berkeley

Code: https://github.com/facebookresearch/ConvNeXt



Papers with code https://paperswithcode.com/