Vision Transformer [ViT]

An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale

(A game changer, CNN, MLP-Mixer)

2023.03.03.

수학과 2020160032 이승은

Table of contents

➡ **Attention**

Attention as a feature selection

- **Attention can be viewed as a soft & dynamic feature selection**

: adaptive feature selection // depending on the data
: curse of dimensionality

- **Spatial Attention**

Figure 1. Our model learns a words/image alignment. The visualized attentional maps (3) are explained in section 3.1 & 5.4

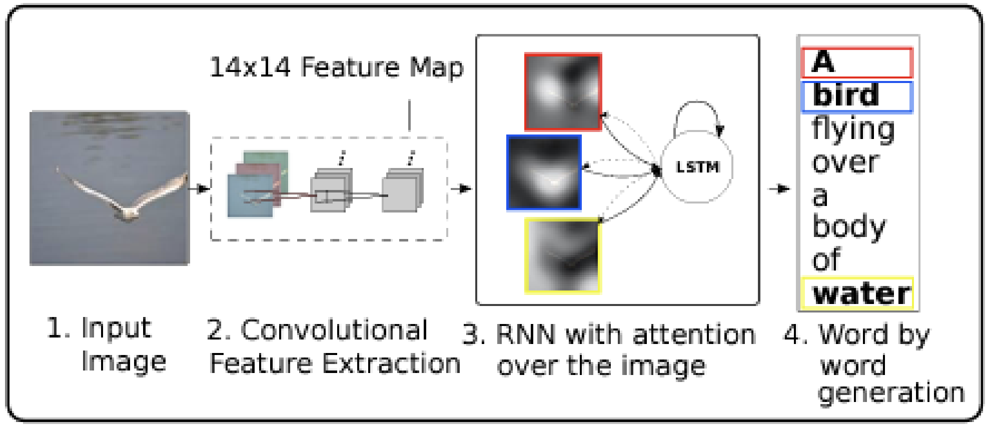


Figure 2. Attention over time. As the model generates each word, its attention changes to reflect the relevant parts of the image. "soft" (top row) vs "hard" (bottom row) attention. (Note that both models generated the same captions in this example.)

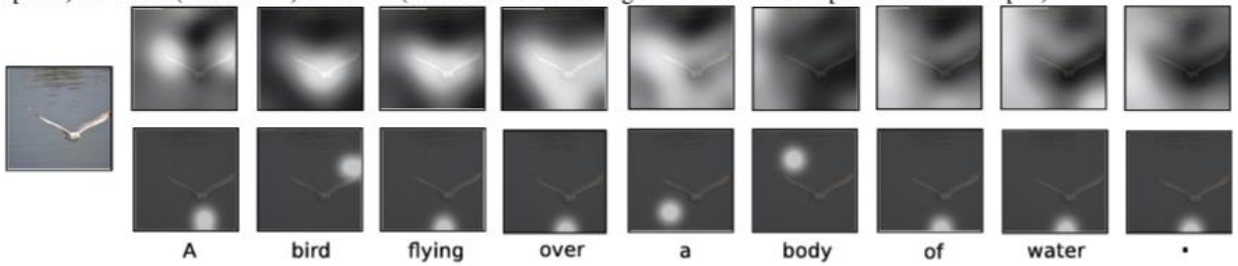


Figure 3. Examples of attending to the correct object (*white* indicates the attended regions, *underlines* indicated the corresponding word)

[ICML15] Xu, Kelvin, et al. "Show, attend and tell: Neural image caption generation with visual attention."

Attention as a feature selection

- **Channel Attention**

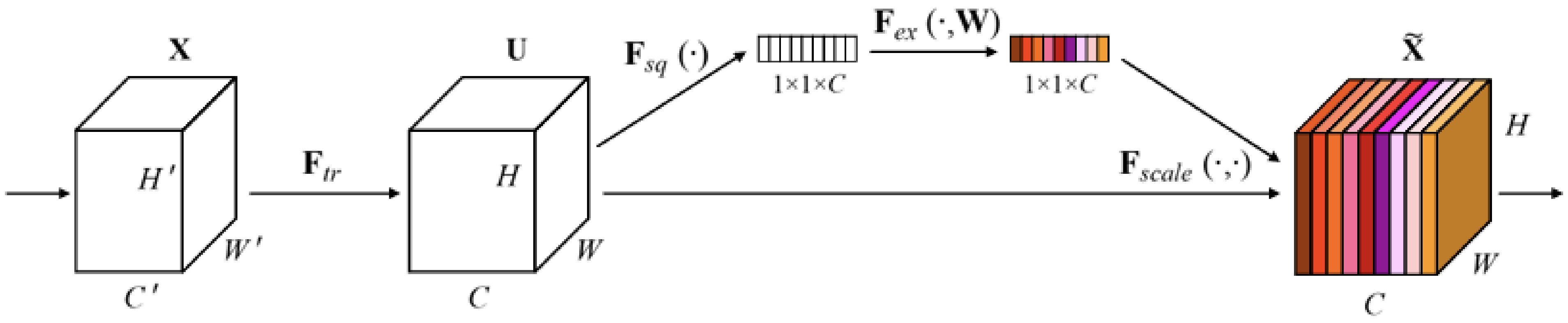


Fig. 1. A Squeeze-and-Excitation block.

[CVPR18] Hu, Jie, Li Shen, and Gang Sun. "Squeeze-and-excitation networks."

Attention as a feature aggregation

- **Scaled dot-product Attention (Transformer)**

$$\mathrm{Attention}(Q, K, V) = \mathrm{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

$X \in \mathbb{R}^{\cancel{d \times N}\ N \times d}$

$Q \in \mathbb{R}^{\cancel{d' \times N}\ N \times d'}$ $\quad Q = \cancel{W_q X}\ XW_q$ where $W_q \in \mathbb{R}^{\cancel{d' \times d}\ d \times d'}$

$K \in \mathbb{R}^{\cancel{d' \times N}\ N \times d'}$ $\quad K = \cancel{W_k X}\ XW_k$ where $W_k \in \mathbb{R}^{\cancel{d' \times d}\ d \times d'}$ } Similarly.

$V \in \mathbb{R}^{\cancel{d \times N}\ N \times d'}$ $\quad V = \cancel{W_v X}\ XW_v$ where $W_v \in \mathbb{R}^{\cancel{d \times d}\ d \times d'}$

$Q = [\ -\ |\ -\ ]$ $\quad$ i^{th} column vector of $Q =: q_i$

$K = [\ -\ |\ -\ ]$ $\quad$ j^{th} col $=: k_j$

$\Rightarrow Q = \begin{bmatrix} \vdots \\ - \\ \vdots \end{bmatrix} q_i \ //\ K = \begin{bmatrix} \vdots \\ - \\ \vdots \end{bmatrix} k_j \ //\ sim(q_i, k_j) = \cancel{q_i^T k_j} = q_i k_j^T$

col vec

row vec

$\Rightarrow \underline{QK^T} = [\, sim(i,j)\,]$

$(N \times d') \times (d' \times N) \Rightarrow N \times N$

Similarity btw q_i and k_j $\Rightarrow$ $sim(q_i, k_j) = q_i^T k_j$; want to do this for every single pair.

$\Rightarrow Q^T K = \begin{bmatrix} \text{Similarity btw} \\ \text{all pairs} \end{bmatrix}$

$sim(i,j)$

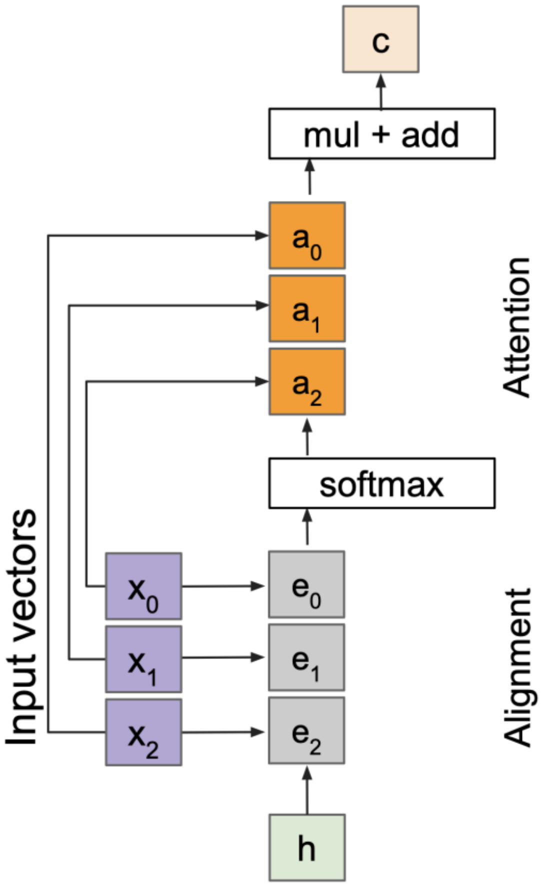
e.g.) $Q = \begin{bmatrix} q_{11} & q_{12} & q_{13} \\ q_{21} & q_{22} & q_{23} \end{bmatrix} \in \mathbb{R}^{2\times3}$, $\quad K = \begin{bmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \end{bmatrix} \in \mathbb{R}^{2\times3}$ $\rightarrow 3\times2$.

$\cancel{Q^T K}$ $QK^T = \cancel{[\ q}\ \begin{bmatrix} q_{11} & q_{12} & q_{13} \\ q_{21} & q_{22} & q_{23} \end{bmatrix} \begin{bmatrix} k_{11} & k_{21} \\ k_{12} & k_{22} \\ k_{13} & k_{23} \end{bmatrix} = \begin{bmatrix} \bigcirc & \square \\ \triangle & \heartsuit \end{bmatrix}$

$2\times3 \quad 3\times2 \quad \Rightarrow \quad 2\times2$

Attention as a feature aggregation

- **General Attention layer**



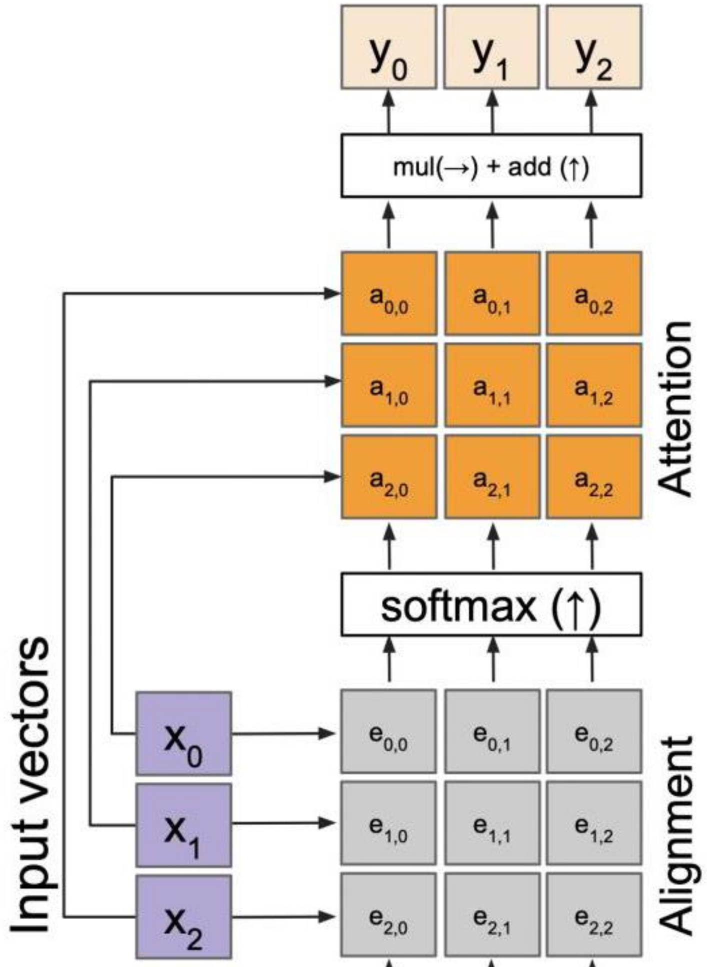
Outputs:
context vector: **c** (shape: D)

Operations:
$\boxed{\text{Alignment: } e_i = h \cdot x_i \,/\, \sqrt{D}}$
Attention: $\mathbf{a} = \text{softmax}(\mathbf{e})$
Output: $\mathbf{c} = \sum_i a_i x_i$

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

Change $f_{att}(.)$ to a scaled simple dot product

- Larger dimensions means more terms in the dot product sum.
- So, the variance of the logits is higher. Large magnitude vectors will produce much higher logits.
- So, the post-softmax distribution has lower-entropy, assuming logits are IID.
- Ultimately, these large magnitude vectors will cause softmax to peak and assign very little weight to all others
- Divide by $\sqrt{D}$ to reduce effect of large magnitude vectors



Outputs:

context vectors: **y** (shape: D)

Multiple query vectors

- each query creates a new output context vector

Operations:

Alignment: $e_{i,j} = q_j \cdot x_i / \sqrt{D}$

Attention: $\mathbf{a} = \text{softmax}(\mathbf{e})$

Output: $y_j = \sum_i a_{i,j} x_i$

Inputs:

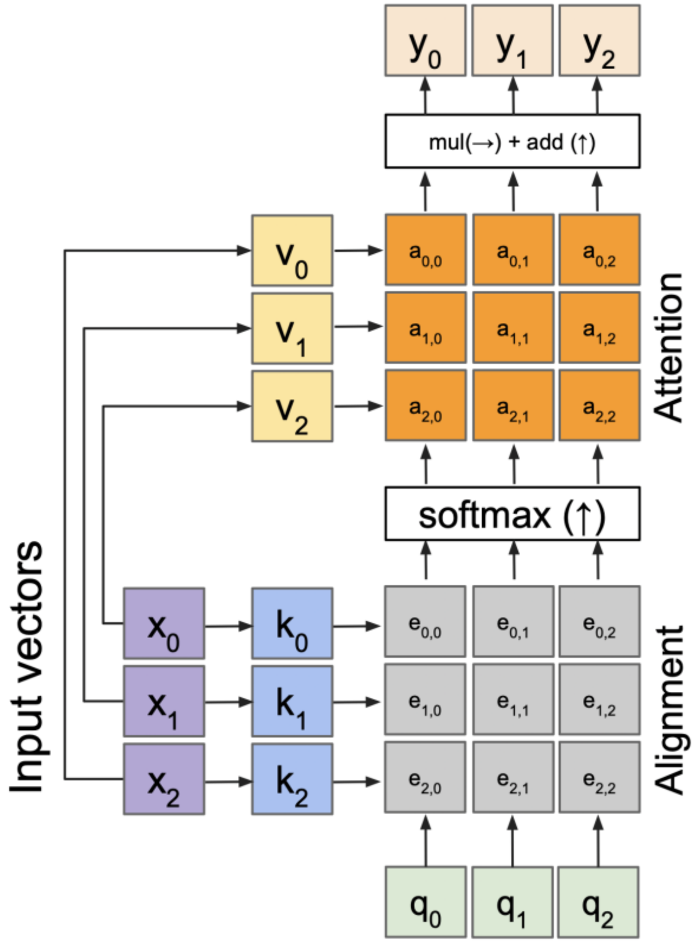
Input vectors: **x** (shape: N x D)

Queries: **q** (shape: M x D)

Multiple query vectors

Attention as a feature aggregation

- **Cross-Attention**



Outputs:
context vectors: **y** (shape: $\boxed{D_v}$)

Operations:
Key vectors: $\mathbf{k} = \mathbf{x}\mathbf{W}_k$
Value vectors: $\mathbf{v} = \mathbf{x}\mathbf{W}_v$
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)
Queries: **q** (shape: M x $\boxed{D_k}$)

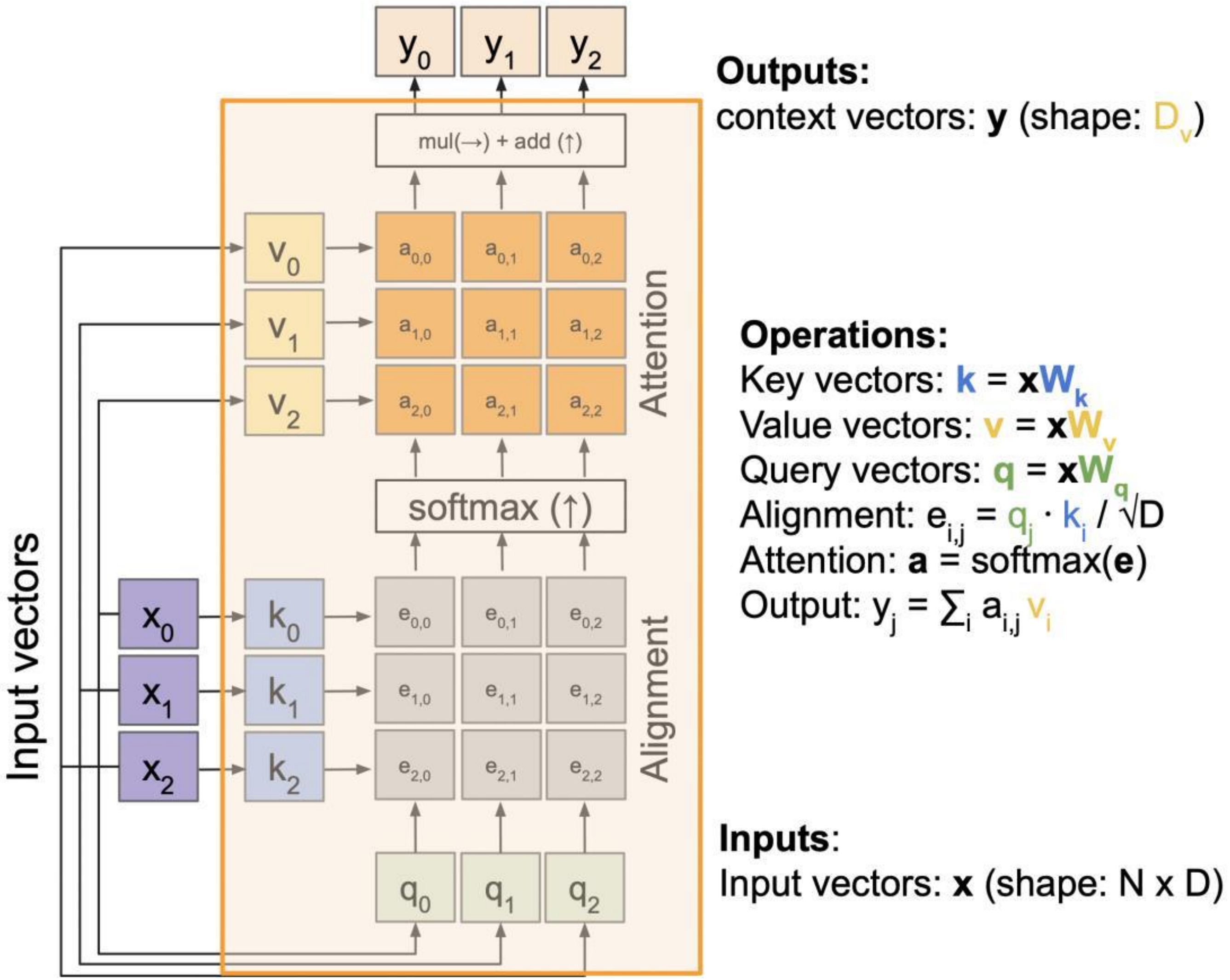
The input and output dimensions can now change depending on the key and value FC layers

Notice that the input vectors are used for both the alignment as well as the attention calculations.

- We can add more expressivity to the layer by adding a different FC layer before each of the two steps.

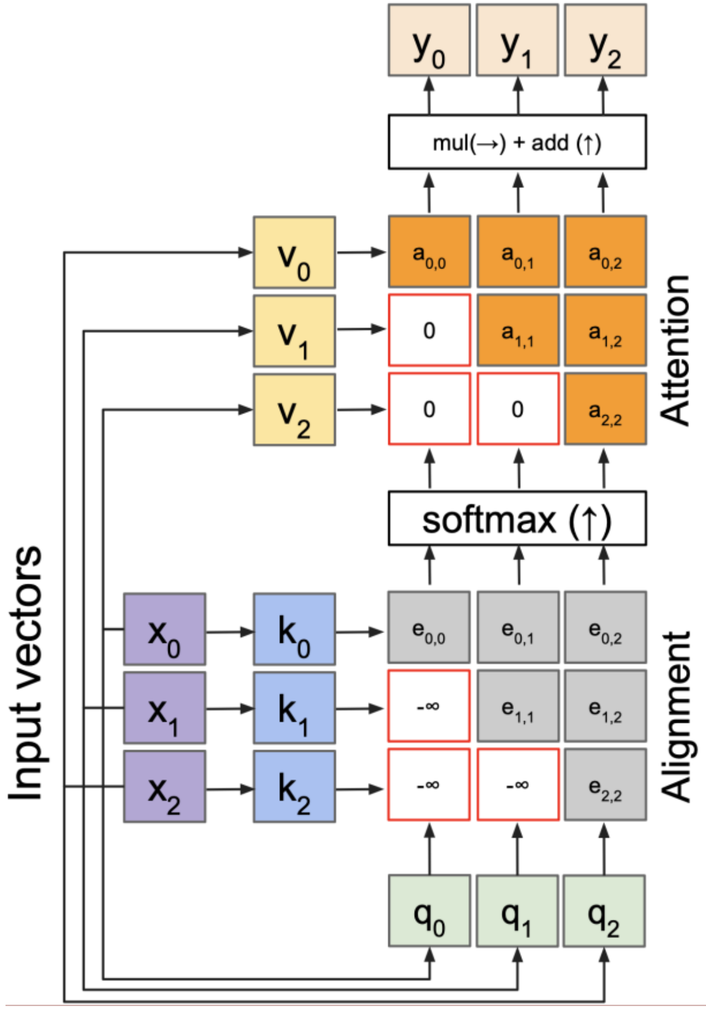
Attention as a feature aggregation

- **Self-Attention**



Attention as a feature aggregation

- **Masked Self-Attention**



Outputs:
context vectors: $\mathbf{y}$ (shape: D_v)

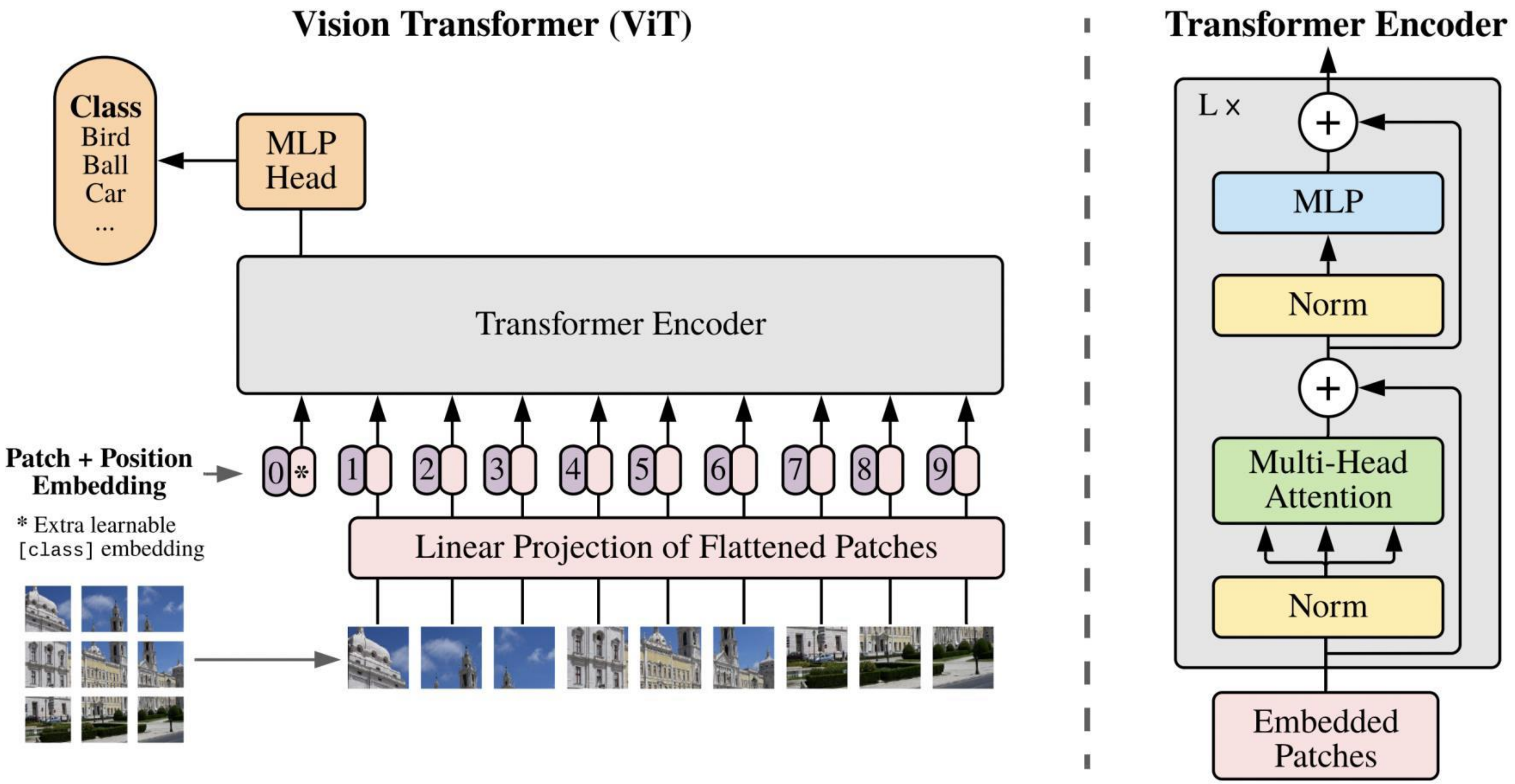
Operations:
Key vectors: $\mathbf{k} = \mathbf{x}\mathbf{W}_k$
Value vectors: $\mathbf{v} = \mathbf{x}\mathbf{W}_v$
Query vectors: $\mathbf{q} = \mathbf{x}\mathbf{W}_q$
Alignment: $e_{i,j} = q_j \cdot k_i / \sqrt{D}$
Attention: $\mathbf{a} = \text{softmax}(\mathbf{e})$
Output: $y_j = \sum_i a_{i,j} v_i$

Inputs:
Input vectors: $\mathbf{x}$ (shape: N x D)

- Prevent vectors from looking at future vectors.
- Manually set alignment scores to -infinity

Vision Transformer

- **Architecture**



The MLP contains two layers with a GELU non-linearity.

$$\mathbf{z}_0 = [\mathbf{x}_{\text{class}};\, \mathbf{x}_p^1\mathbf{E};\, \mathbf{x}_p^2\mathbf{E}; \cdots;\, \mathbf{x}_p^N\mathbf{E}] + \mathbf{E}_{pos}, \qquad \mathbf{E} \in \mathbb{R}^{(P^2\cdot C)\times D},\, \mathbf{E}_{pos} \in \mathbb{R}^{(N+1)\times D} \tag{1}$$

$$\mathbf{z'}_\ell = \mathrm{MSA}(\mathrm{LN}(\mathbf{z}_{\ell-1})) + \mathbf{z}_{\ell-1}, \qquad \ell = 1 \ldots L \tag{2}$$

$$\mathbf{z}_\ell = \mathrm{MLP}(\mathrm{LN}(\mathbf{z'}_\ell)) + \mathbf{z'}_\ell, \qquad \ell = 1 \ldots L \tag{3}$$

$$\mathbf{y} = \mathrm{LN}(\mathbf{z}_L^0) \tag{4}$$

https://github.com/lucidrains/vit-pytorch/blob/main/images/vit.gif

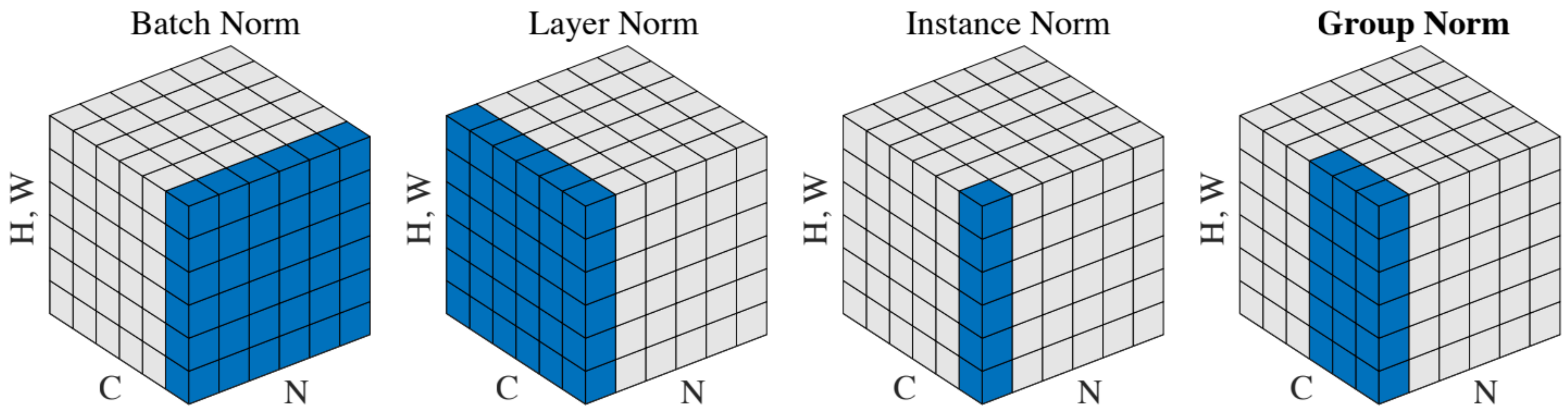


Figure 2. **Normalization methods**. Each subplot shows a feature map tensor, with N as the batch axis, C as the channel axis, and (H, W) as the spatial axes. The pixels in blue are normalized by the same mean and variance, computed by aggregating the values of these pixels.

Vision Transformer

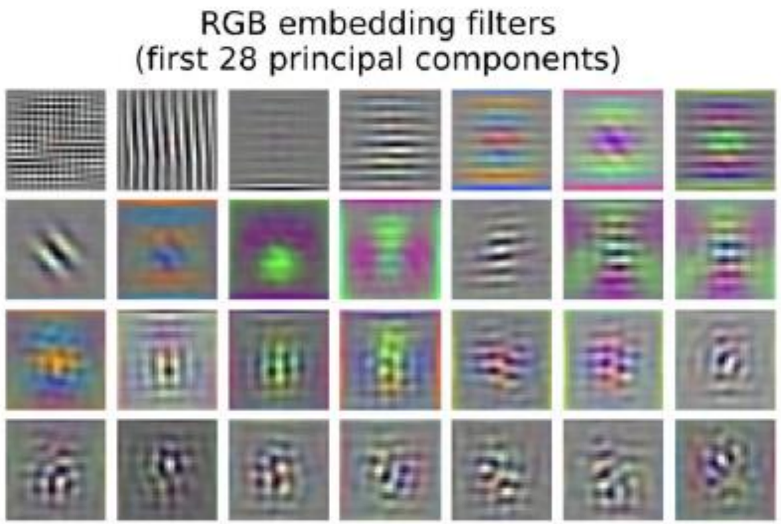
- **Details and Interpretation**

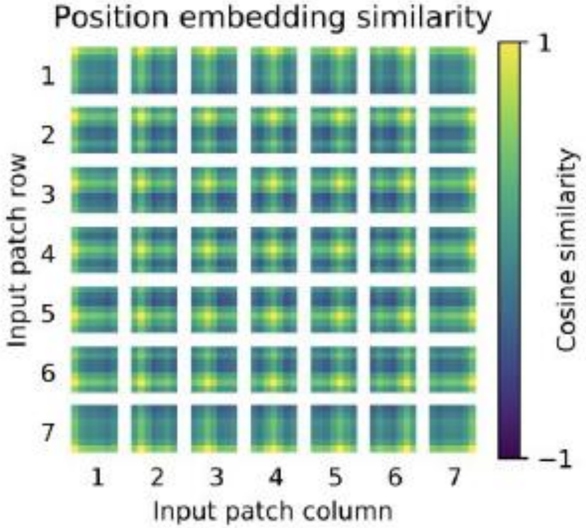
- ViT vs CNN
  - Locality of features:
    - CNN localized vs ViT global
  - Translation equivariance
    - CNN O vs ViT (X?)
  - Fine-tuning? Higher Resolution?
    - Change the prediction head and learn it

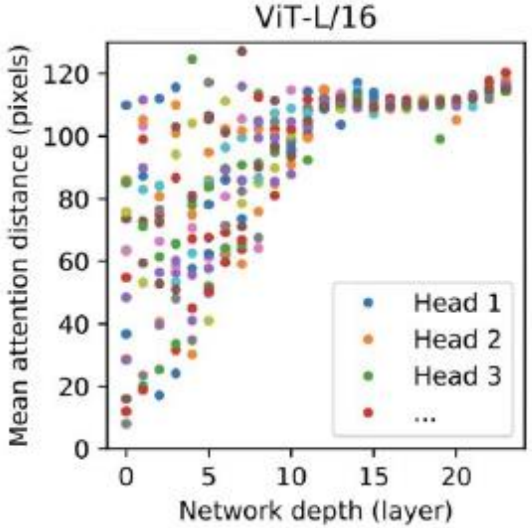
- Invariance F(T(X)) = F(X)
- equivariance F(T(X)) = T(F(X))

	Ours-JFT (ViT-H/14)	Ours-JFT (ViT-L/16)	Ours-I21k (ViT-L/16)	BiT-L (ResNet152x4)	Noisy Student (EfficientNet-L2)
ImageNet	$\mathbf{88.55} \pm 0.04$	87.76 ± 0.03	85.30 ± 0.02	87.54 ± 0.02	$88.4/88.5^{*}$
ImageNet ReaL	$\mathbf{90.72} \pm 0.05$	90.54 ± 0.03	88.62 ± 0.05	90.54	90.55
CIFAR-10	$\mathbf{99.50} \pm 0.06$	99.42 ± 0.03	99.15 ± 0.03	99.37 ± 0.06	–
CIFAR-100	$\mathbf{94.55} \pm 0.04$	93.90 ± 0.05	93.25 ± 0.05	93.51 ± 0.08	–
Oxford-IIIT Pets	$\mathbf{97.56} \pm 0.03$	97.32 ± 0.11	94.67 ± 0.15	96.62 ± 0.23	–
Oxford Flowers-102	99.68 ± 0.02	$\mathbf{99.74} \pm 0.00$	99.61 ± 0.02	99.63 ± 0.03	–
VTAB (19 tasks)	$\mathbf{77.63} \pm 0.23$	76.28 ± 0.46	72.72 ± 0.21	76.29 ± 1.70	–
TPUv3-core-days	2.5k	0.68k	0.23k	9.9k	12.3k

Table 2: Comparison with state of the art on popular image classification benchmarks. We report mean and standard deviation of the accuracies, averaged over three fine-tuning runs. Vision Transformer models pre-trained on the JFT-300M dataset outperform ResNet-based baselines on all datasets, while taking substantially less computational resources to pre-train. ViT pre-trained on the smaller public ImageNet-21k dataset performs well too. *Slightly improved 88.5% result reported in Touvron et al. (2020).







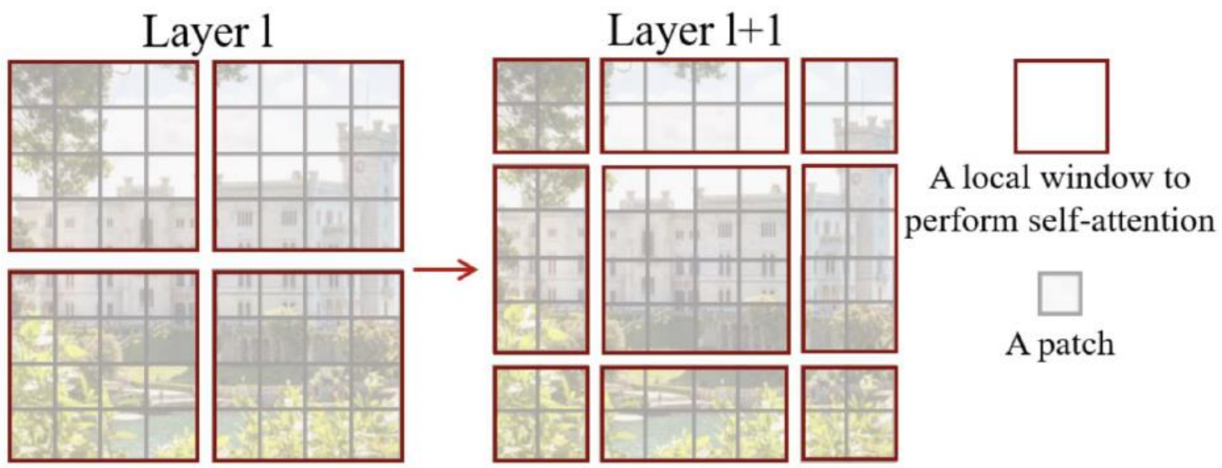
~ Conv Kernels

~ Translation Equivariant?

~ receptive field

Swin Transformer

- Patch Merging
- Shifted Windows ➡ Adds Locality (cf. LocalViT)
- Hierarchic
- Segmentation, Detection
- Converges faster than Vanilla ViT (but depends on the task in practice)



- shifted window
- self-attention computation in the new windows crosses the boundaries of the previous windows (tackling non-overlapped windows)

Most Asked Questions

- **Attention vs. Transformer**

  - Transformer: Attention is All you Need
  RNN, LSTM의 한계 (순차적 입력 -> 병렬처리 어려움)
  Transformer: 순차적 x
  -> sequence를 한번에 병렬처리
  \+ Attention으로 중요 부분 전달 및 위치 정보 활용

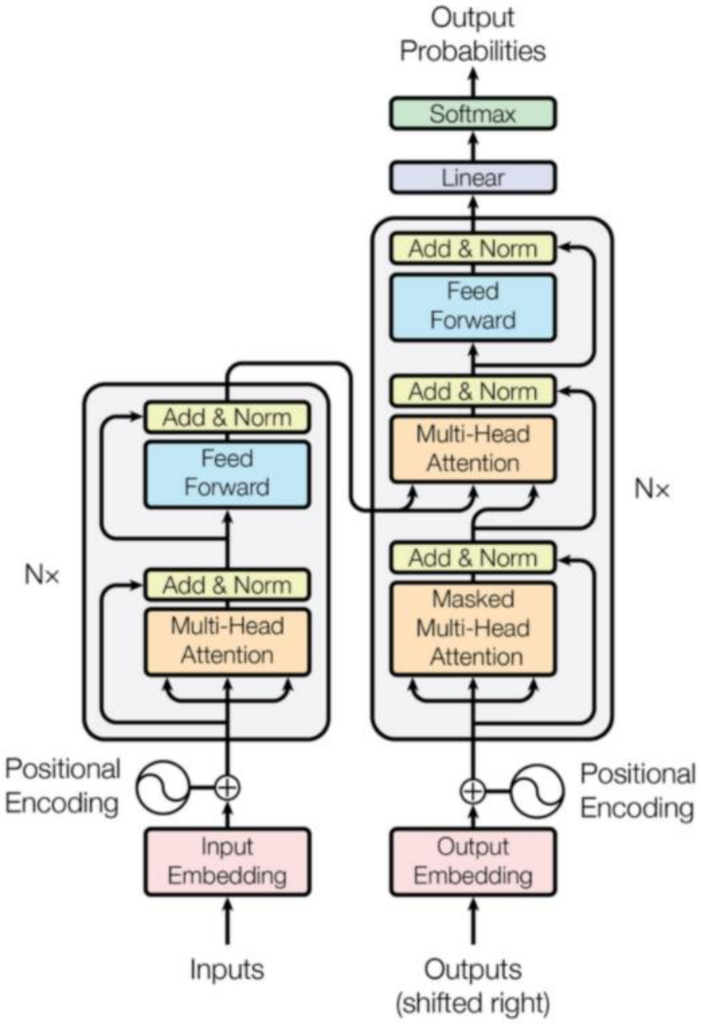


Figure 1: The Transformer - model architecture.

Most Asked Questions

- **Why did ViT succeed? (In my opinion…)**

  - Transformer (convergence, stable)
  - Pre-trained
  - Global (detailed feature extraction)

Most Asked Questions

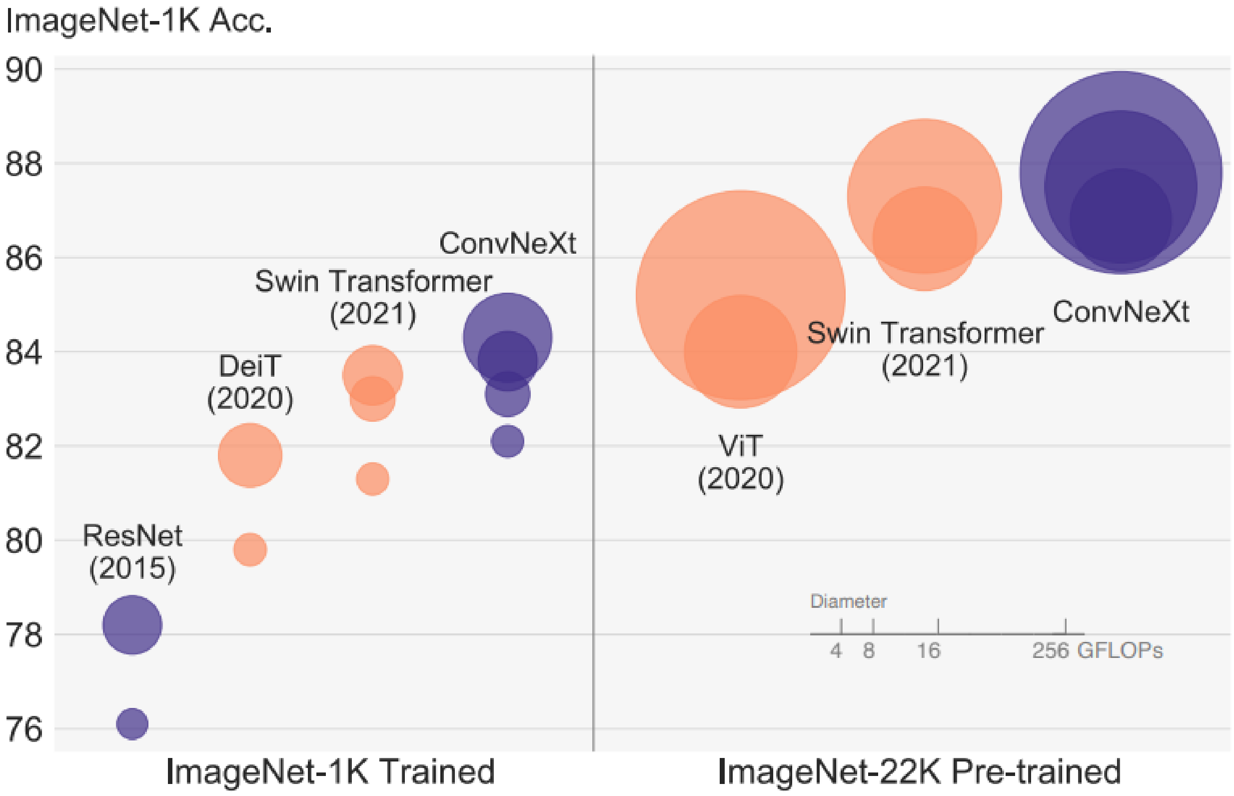
- **Limitations of ViT**

  - OOM [Out of Memory] Error
  - Small dataset
  - Training time

  -> BERT, BART, Transformer -> ViT
  : SOTA in practice

Most Asked Questions

- **CNN vs. ViT in practice**
  - depends on the type of data (Audio, Image, Integrated data)
  - Make CONV Great Again (CNN with Transformer)



Most Asked Questions

- **ViT, SOTA, and the future of Computer Vision**

  - CNN
  - Classical image processing, ML
  - Representation
  - New tasks

Trend	Dataset	Best Model	Paper	Code	Compare
	ImageNet	BASIC-L (Lion, fine-tuned)			See all
	CIFAR-10	ViT-H/14			See all
	CIFAR-100	EffNet-L2 (SAM)			See all
	STL-10	µ2Net+ (ViT-L/16)			See all
	ObjectNet	CoCa			See all
	MNIST	Heterogeneous ensemble with simple CNN			See all
	SVHN	WRN28-10 (SAM)			See all
	ImageNet ReaL	Baseline (ViT-G/14)			See all
	iNaturalist 2018	MetaFormer (MetaFormer-2,384,extra_info)			See all
	Flowers-102	CCT-14/7x2			See all

Pytorch / Keras

Pytorch: https://github.com/rwightman/pytorch-image-models/blob/main/timm/models/vision_transformer.py
Keras: https://github.com/faustomorales/vit-keras

https://drive.google.com/file/d/1S6iKMPWK7GNLiDWOm6Md_sB7WqmD9SOe/view?usp=sharing

Reference

- Lecture 11: Attention and Transformers - CS231n
- An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale (https://arxiv.org/abs/2010.11929)
- Swin Transformer: Hierarchical Vision Transformer using Shifted Windows (https://arxiv.org/abs/2103.14030)