AN IMAGE IS WORTH 16X16 WORDS: TRANSFORMERS FOR IMAGE RECOGNITION AT SCALE

Prabal Ghosh
Côte d'Azur University

AIQDSC34 - ECUE More on Deep Learning Algorithms

https://arxiv.org/abs/2010.11929

Outline

- Motivation for ViT
- Vision Transformer architecture
- Transformer Encoder
- Computing Self-Attention: Example
- Computing Multi-Head Attention: Example
- Vision Transformer Architecture Explained
- Training
- Self-supervision
- Training Dataset Vision Transformer
- Pre-Training Vision Transformer
- Image Classification Accuracies
- Fine-tuning ViT
- Training ViT
- ViT vs ResNet
- Inspecting ViT representations
- Conclusion

Motivation:

- The encoder-decoder architecture of Transformer achieved state of the art performance on machine translation tasks , by allowing significantly more parallelization (i.e less time to train).
- **RestNet** (CNN) was the **best** solution **for image classification**.

- Can we apply Transformers to images and get state-of-the-art results ?

Idea of the Paper:-

- Use the **Transformer Encoder architecture** with fewest possible modifications and apply on **image classification tasks**.

- Idea is to **train Vision Transformer (ViT) on sufficiently large amounts of data**, with fewer computational resources than state-of-the-art CNN, and **then fine_tune on smaller datasets**.

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

Alexey Dosovitskiy[*,†], **Lucas Beyer**[*], **Alexander Kolesnikov**[*], **Dirk Weissenborn**[*], **Xiaohua Zhai**[*], **Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, Neil Houlsby**[*,†]

[*]equal technical contribution, [†]equal advising

Google Research, Brain Team

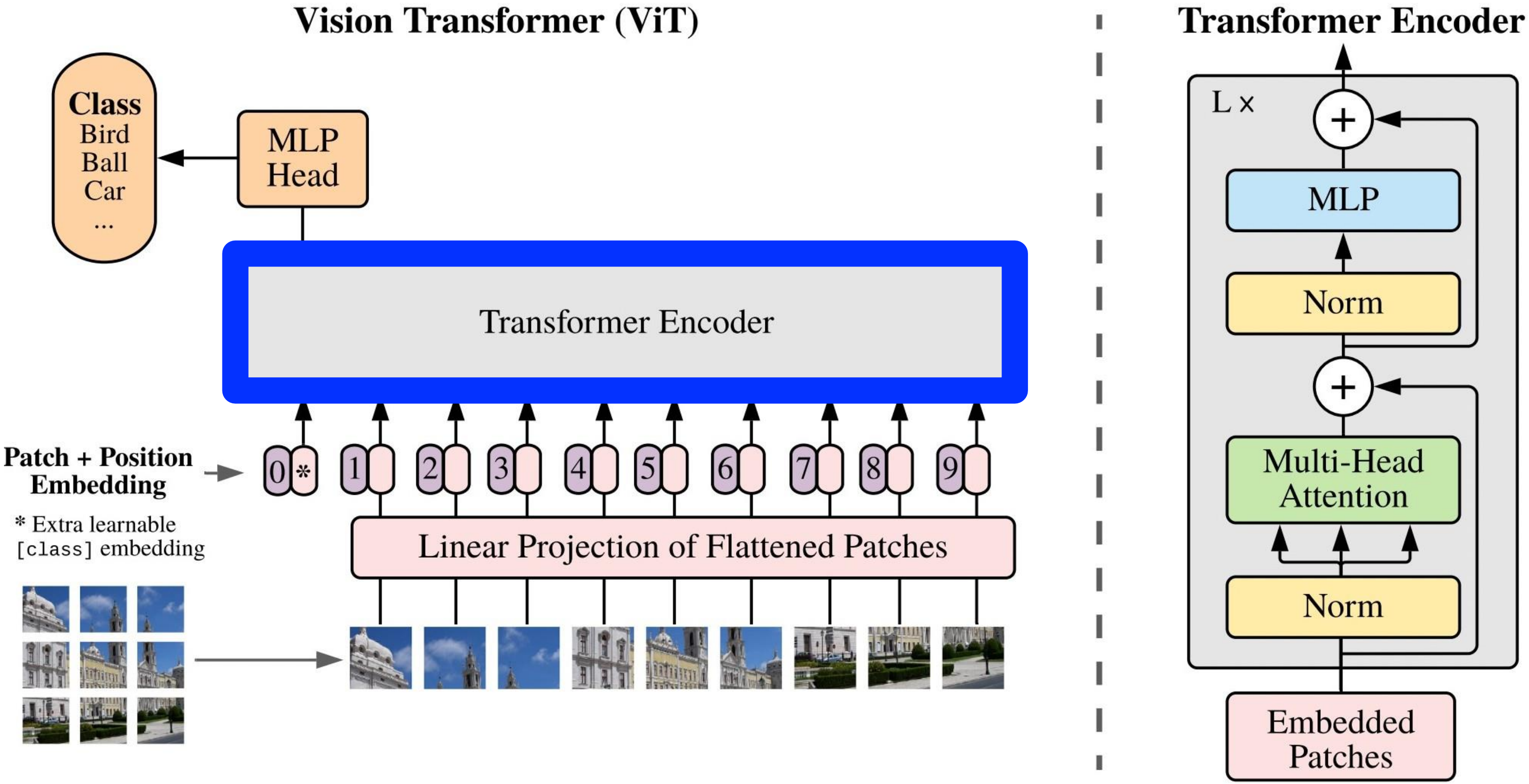
{adosovitskiy, neilhoulsby}@google.com

Abstract

While the Transformer architecture has become the de-facto standard for natural language processing tasks, its applications to computer vision remain limited. In vision, attention is either applied in conjunction with convolutional networks, or used to replace certain components of convolutional networks while keeping their overall structure in place. We show that this reliance on CNNs is not necessary and a pure transformer applied directly to sequences of image patches can perform very well on image classification tasks. When pre-trained on large amounts of data and transferred to multiple mid-sized or small image recognition benchmarks (ImageNet, CIFAR-100, VTAB, etc.), Vision Transformer (ViT) attains excellent results compared to state-of-the-art convolutional networks while requiring substantially fewer computational resources to train.[1]

[[2010.11929] An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale (arxiv.org)]

Vision Transformers Architecture: Uses Popular BERT Architecture



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

Transformer Encoder

Five key ideas

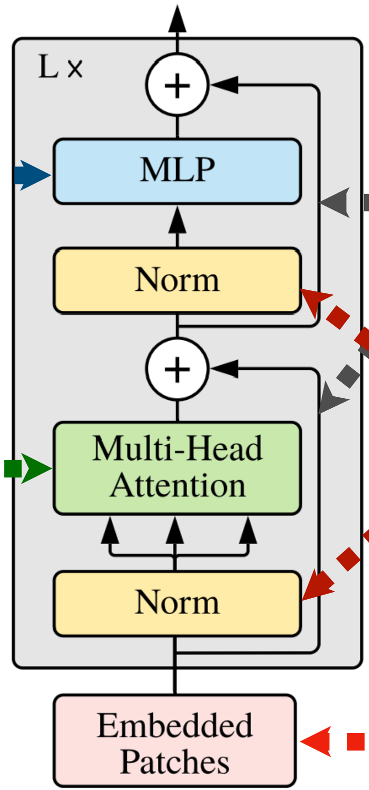
Compute-heavy blocks

Multi-Layer Perceptron

Allows embeddings to "think" independently

Multi-Head Attention

Allows embeddings to communicate



Optimisation blocks

Residual Connections

Help gradients flow

Layer Normalisation

Stabilise learning

Positional Embeddings

Allows transformer to identify embeddings

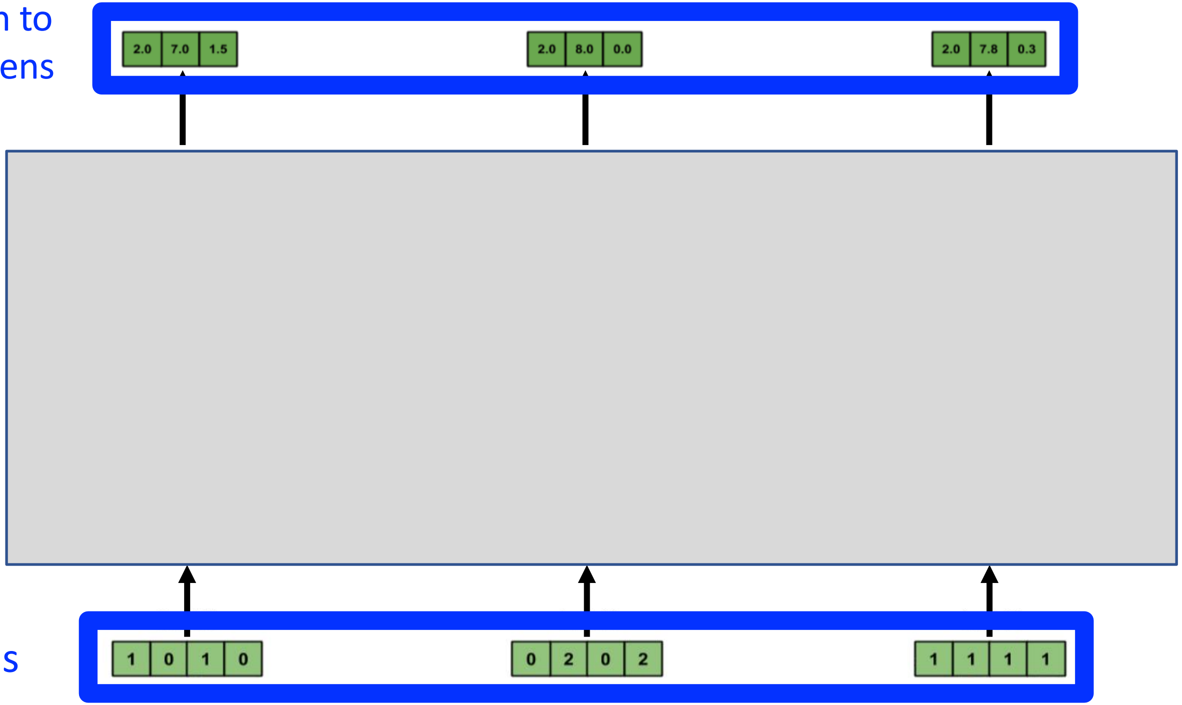
References/Image credits

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

Credit for "communicate/think" metaphor - Andrej Karpathy

Computing Self-Attention: Example

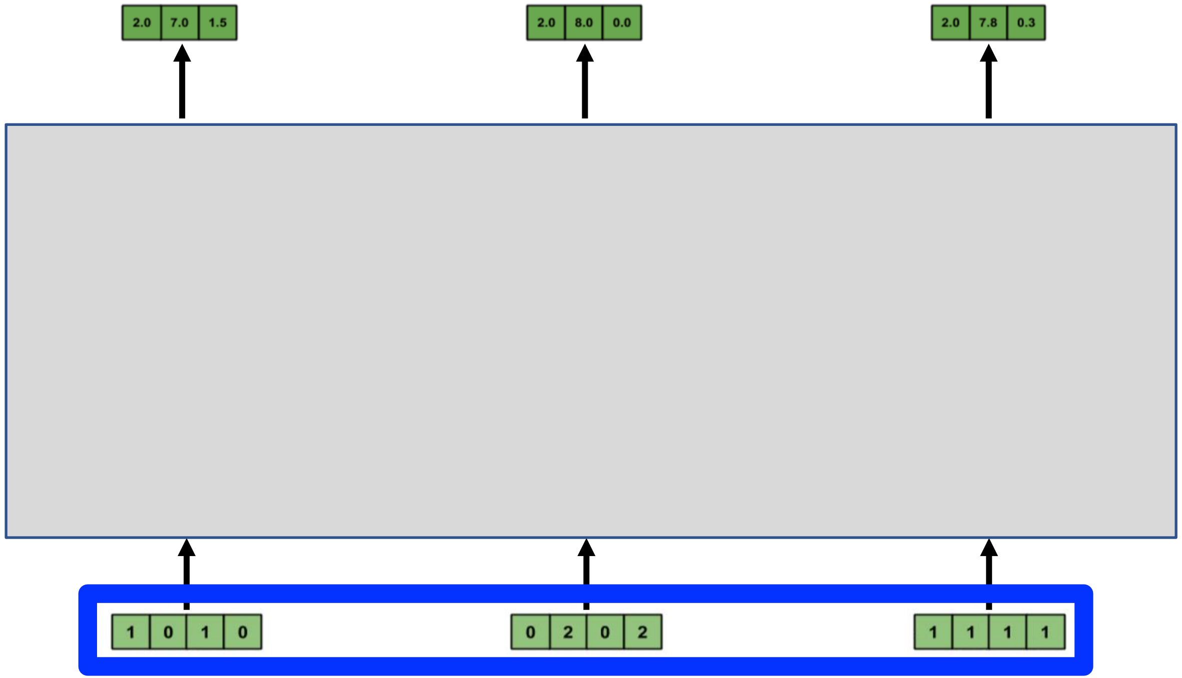
New representation of each input token to reflect each one's relationship to all tokens



Input tokens

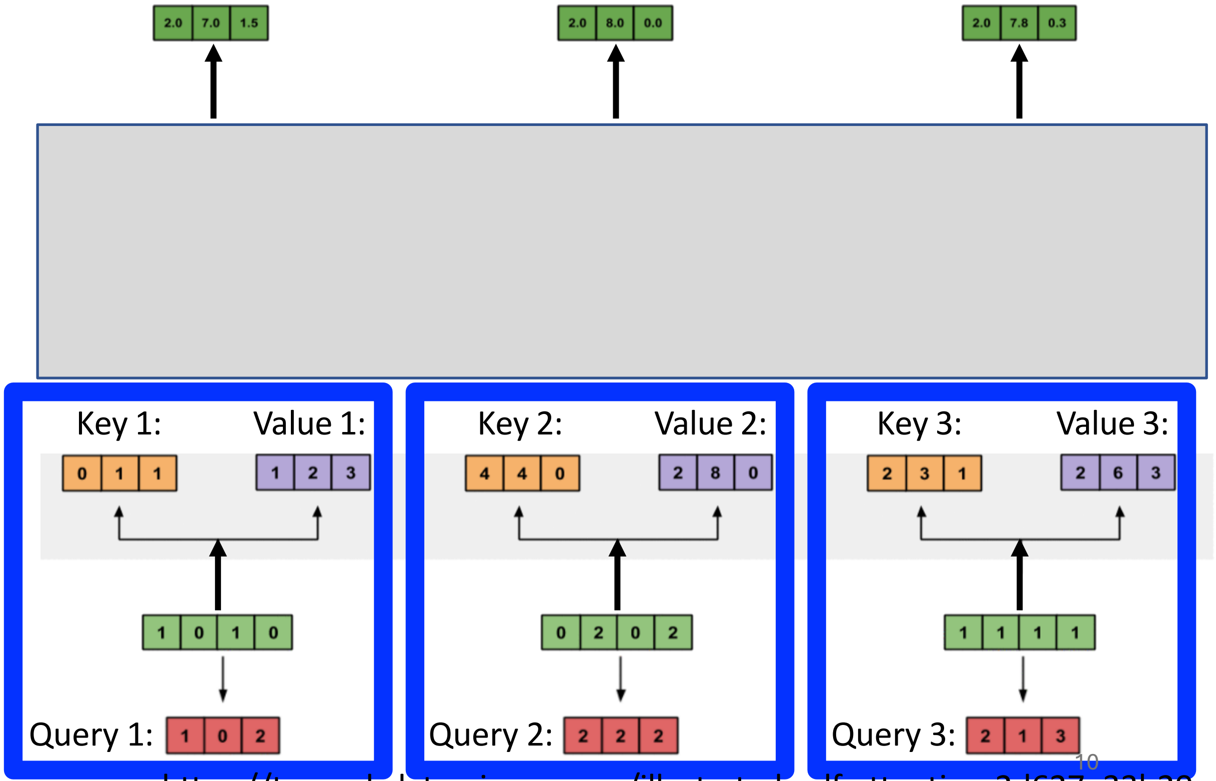
https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example



https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example



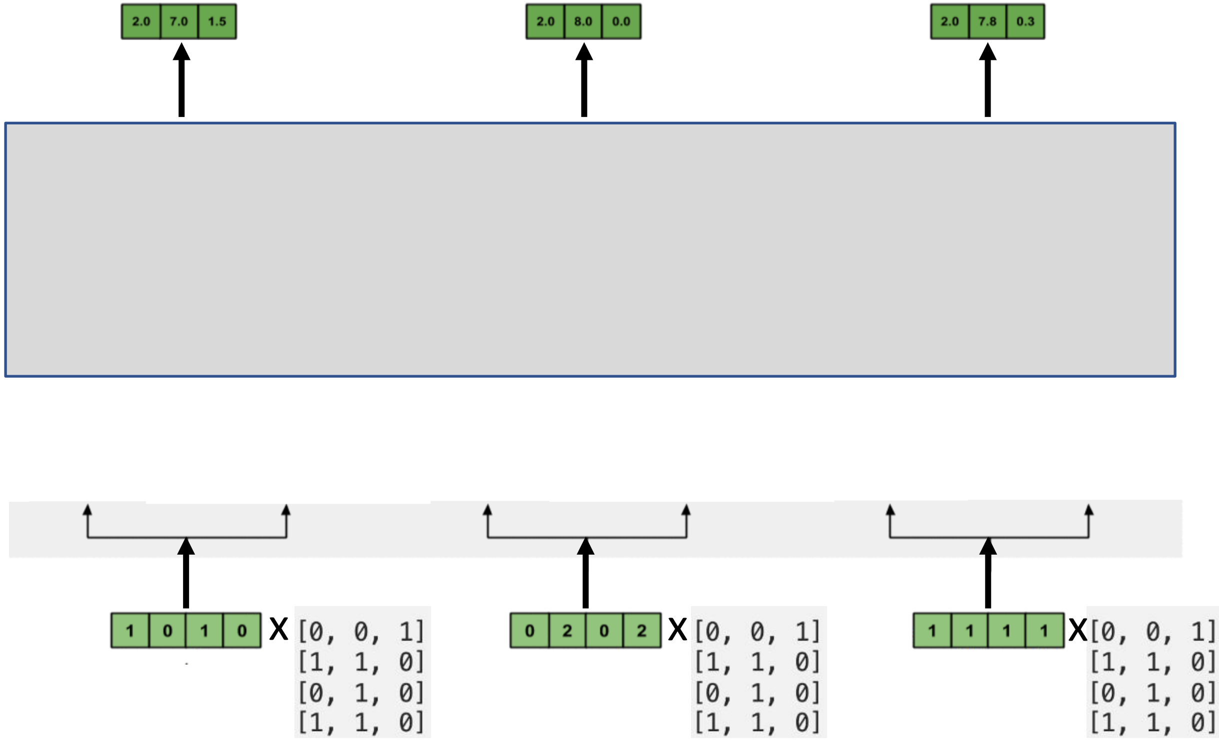
Three vectors are derived for each input by multiplying with three weight matrices (learned during training): query, key, and value

https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example

e.g., key weights

```
[0, 0, 1]
[1, 1, 0]
[0, 1, 0]
[1, 1, 0]
```

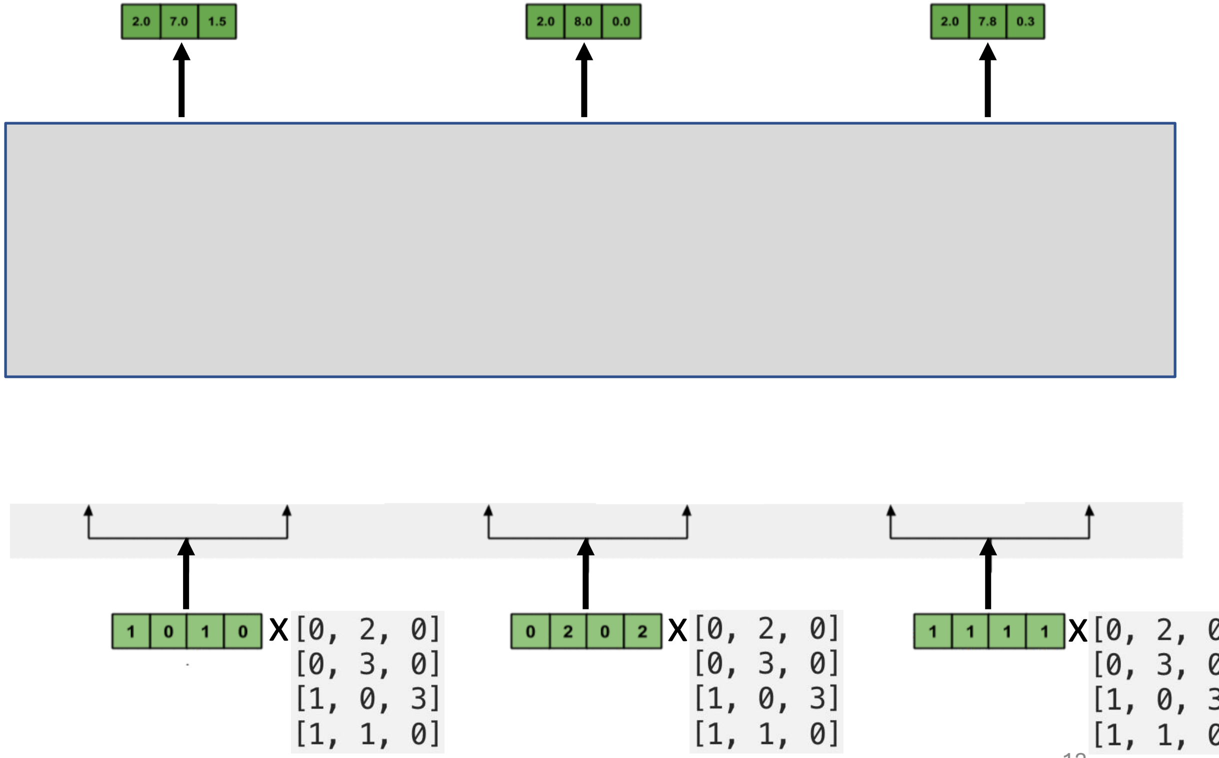


https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example

e.g., value weights

```
[0, 2, 0]
[0, 3, 0]
[1, 0, 3]
[1, 1, 0]
```

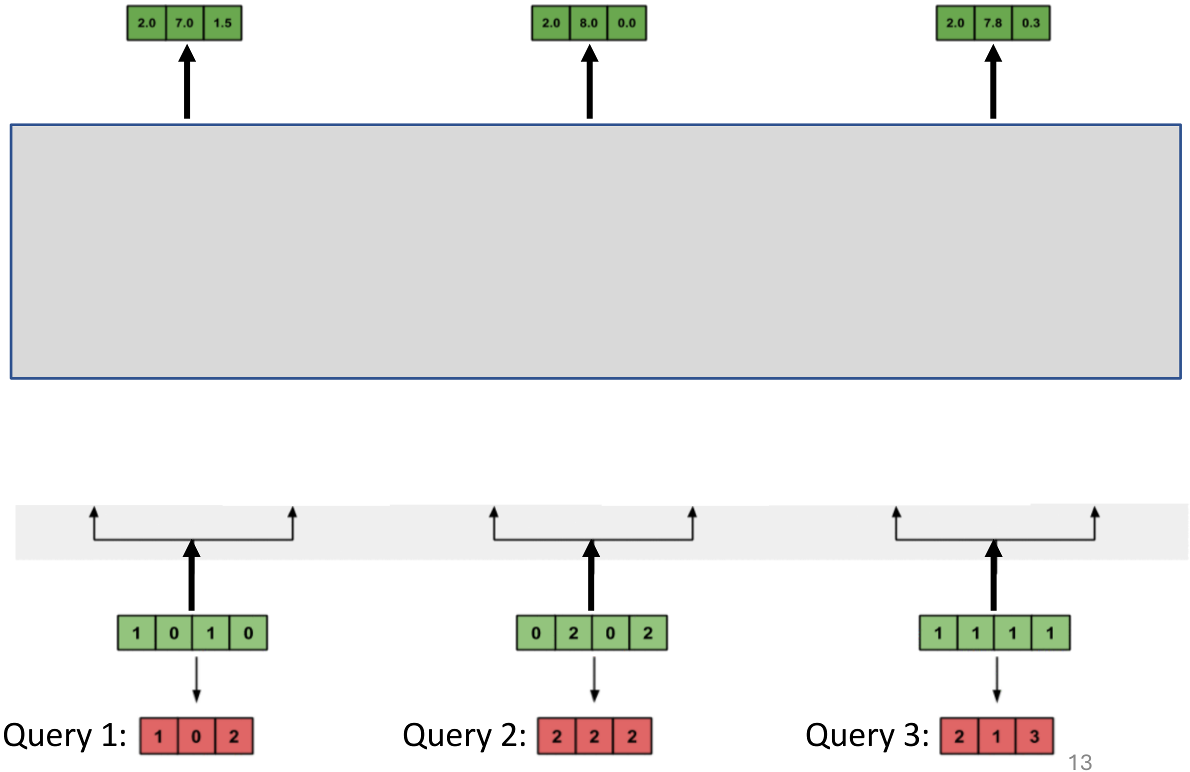


https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example

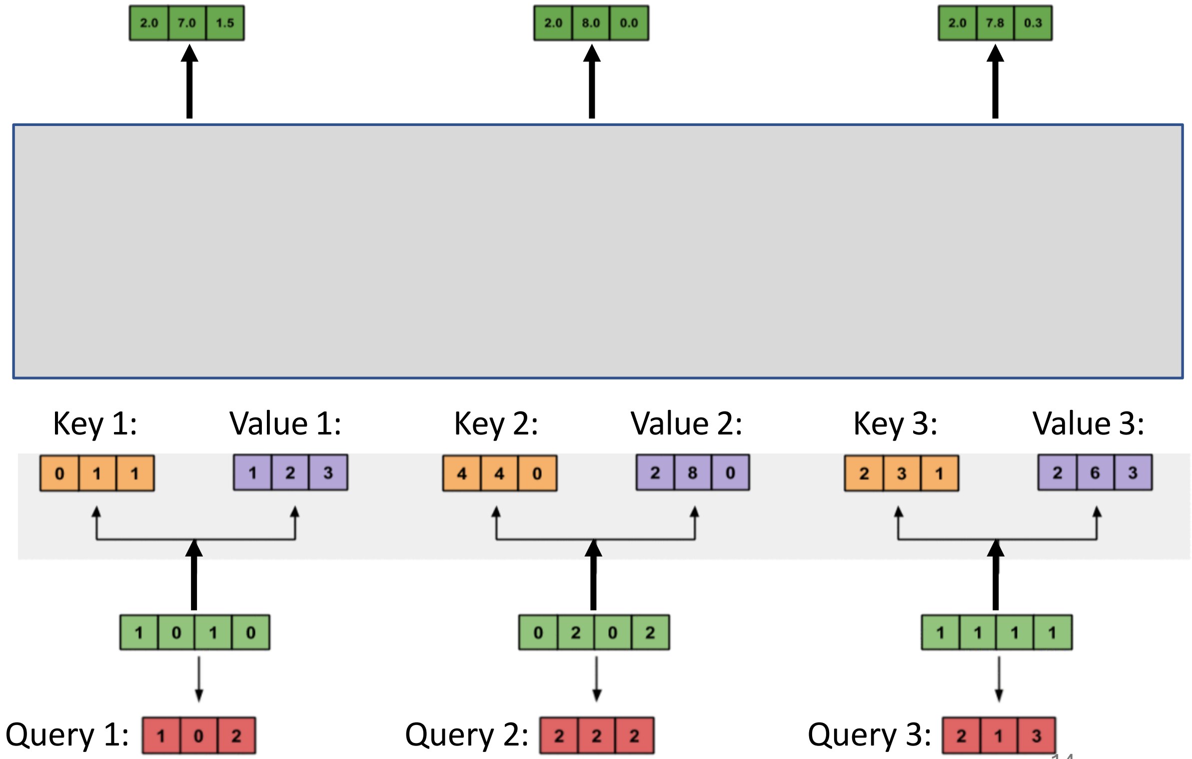
e.g., query weights

```
[1, 0, 1]
[1, 0, 0]
[0, 0, 1]
[0, 1, 1]
```



https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example

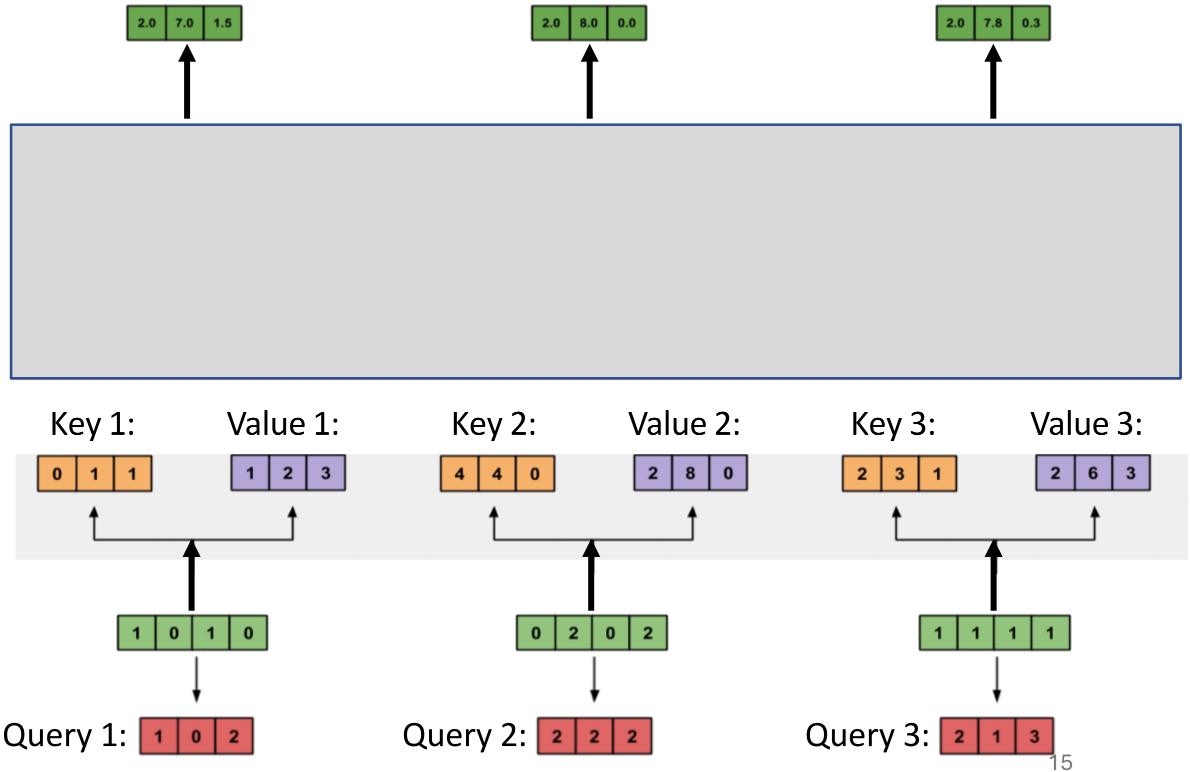


https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example

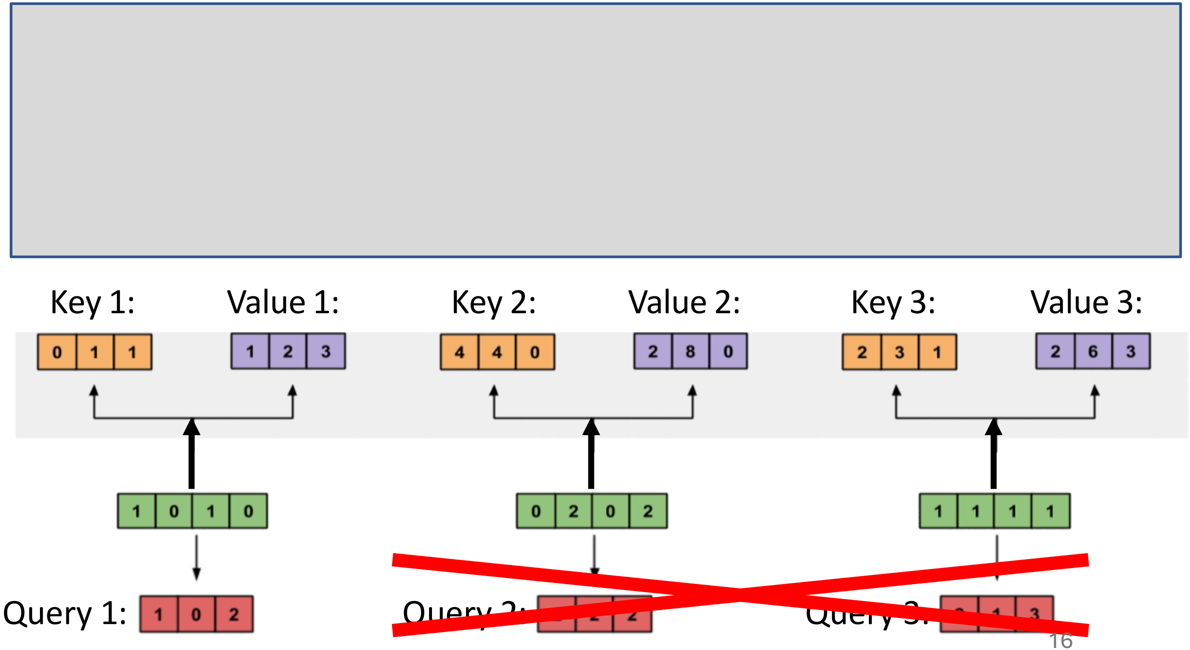
What is the purpose of the three weight matrices?

For each input, 2 of the derived vectors are used to compute **attention weights** (query and key) and the 3rd is **information** passed on for the new representation (value)



https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example



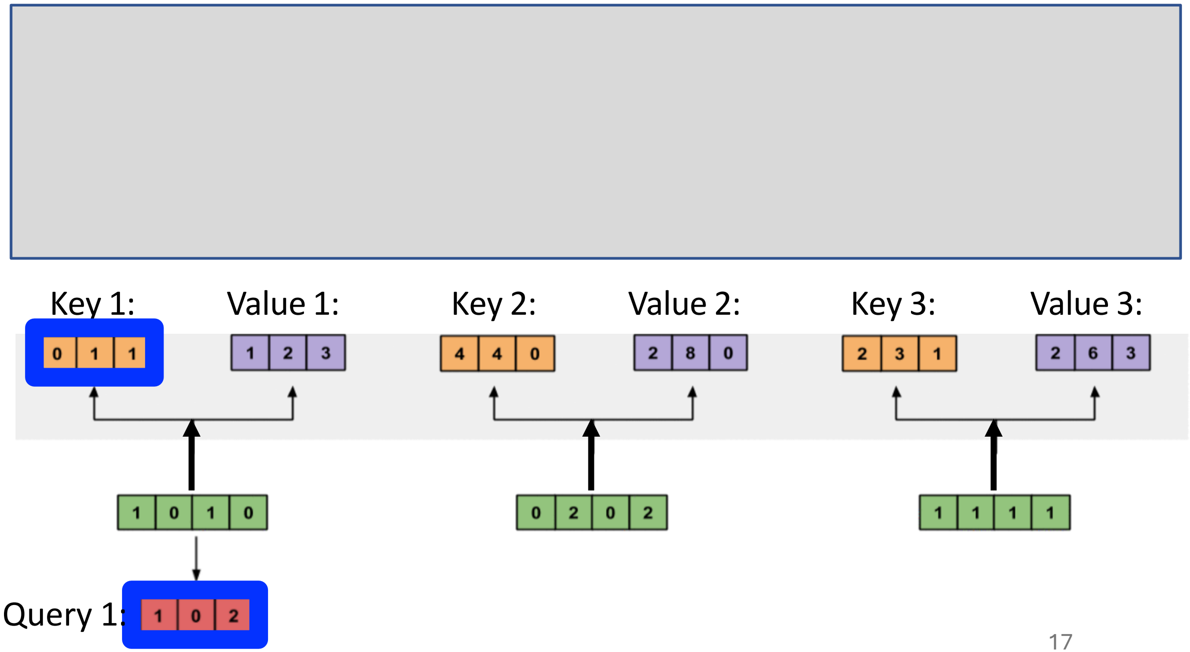
We now will examine how to find the new representation for the first input.

https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example

Attention score: dot product of query with all keys to identify relevant tokens; e.g.,

$$\begin{bmatrix}1 & 0 & 2\end{bmatrix} \times \begin{bmatrix}0\\1\\1\end{bmatrix} = ?$$

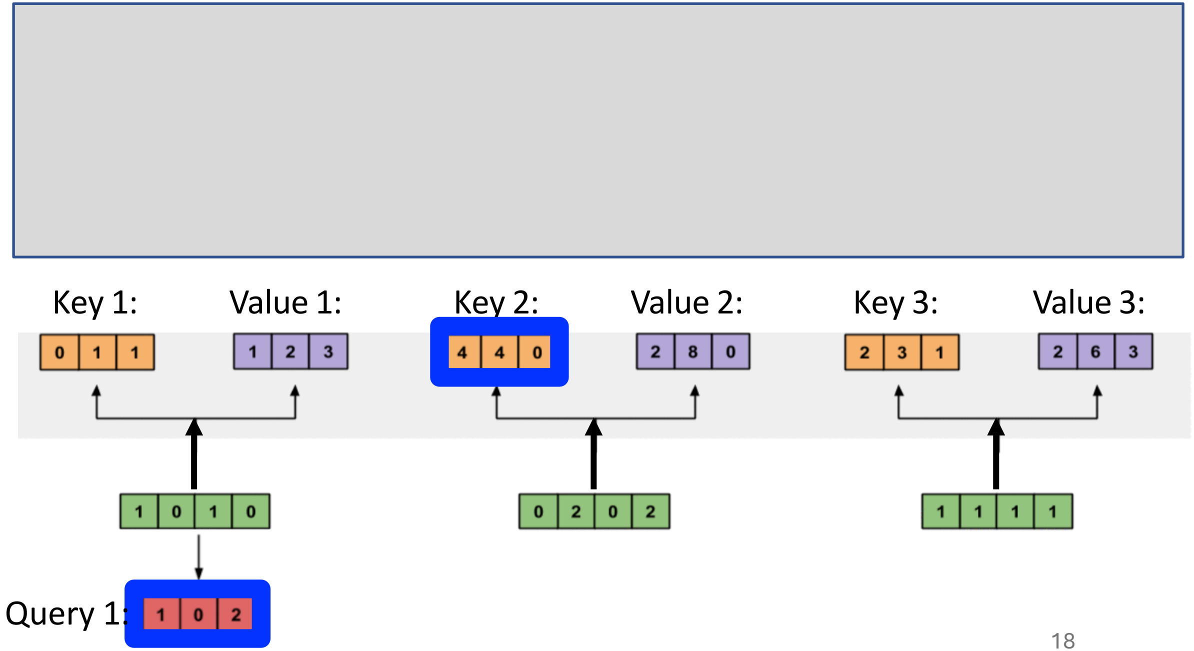


https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example

Attention score: dot product of query with all keys to identify relevant tokens; e.g.,

$$\begin{bmatrix} 1 & 0 & 2 \end{bmatrix} \times \begin{bmatrix} 4 \\ 4 \\ 0 \end{bmatrix} = ?$$

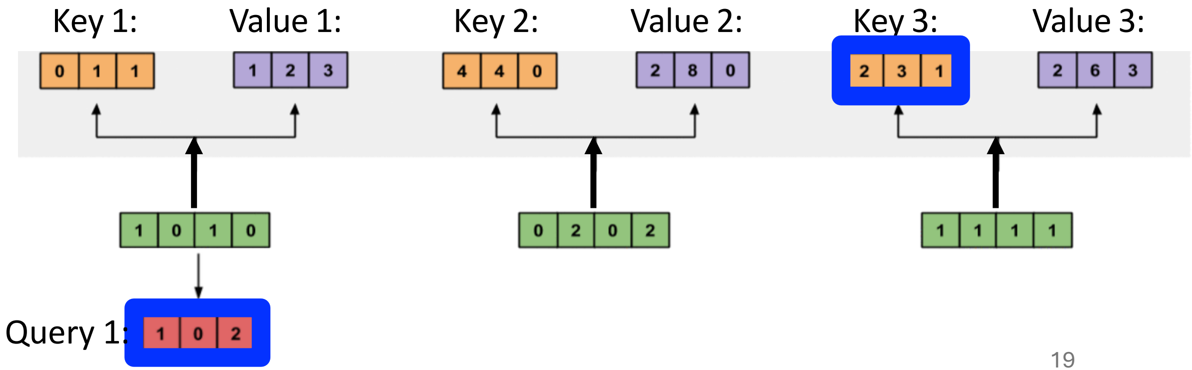


https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example

Attention score: dot product of query with all keys to identify relevant tokens; e.g.,

$$\begin{bmatrix} 1 & 0 & 2 \end{bmatrix} \times \begin{bmatrix} 2 \\ 3 \\ 1 \end{bmatrix} = ?$$



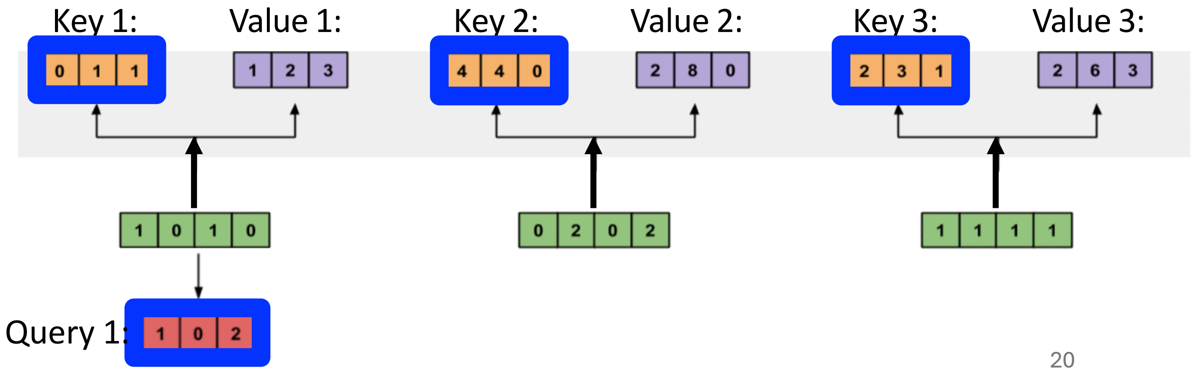
https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example

Why dot product? Indicates similarity of two vectors

- Match = 1 (i.e., cos(0))
- Opposites = -1 (i.e., cos(180))

To which input(s) is input 1 most related?



https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

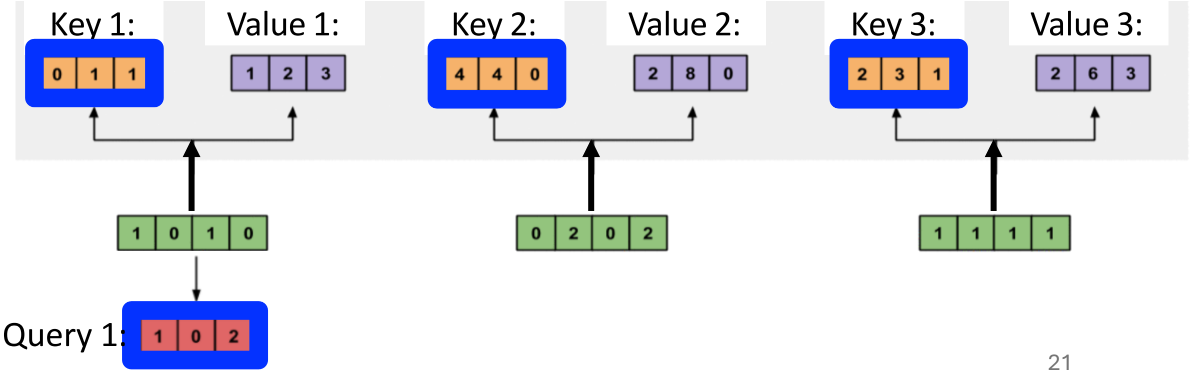
Computing Self-Attention: Example

Attention weights: softmax scores for all inputs to quantify each token's relevance; e.g.,

= softmax([2, 4, 4])

= [0.0, 0.5, 0.5])

Note: softmax doesn't return 0, but can arise from rounding

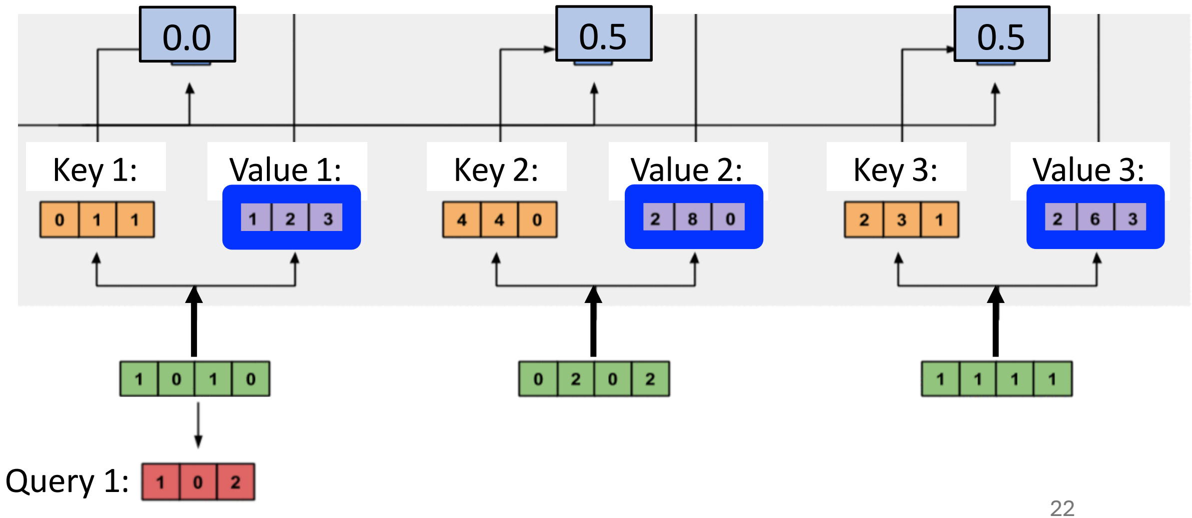


https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example

Compute new representation of input token that reflects entire input:

1. Attention weights x Values



https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

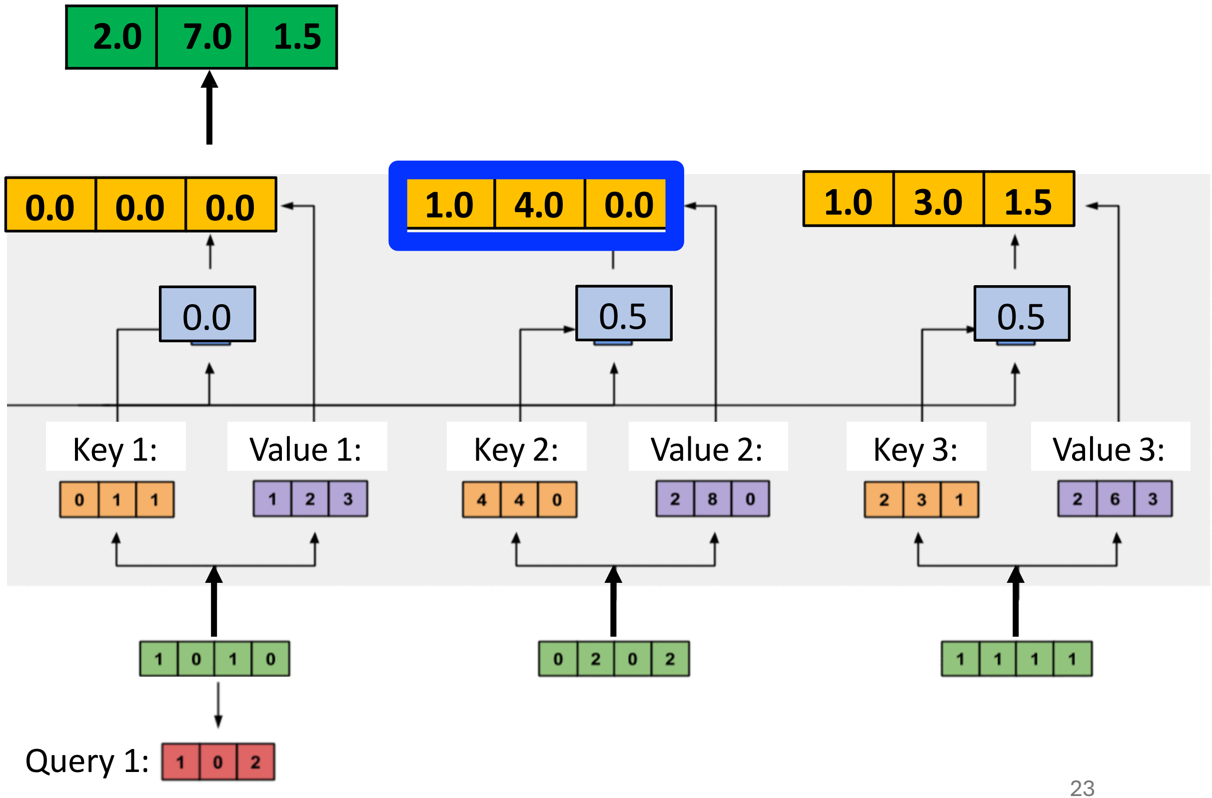
Computing Self-Attention: Example

Compute new representation of input token that reflects entire input:

1. Attention weights x Values

2. Sum all weighted vectors

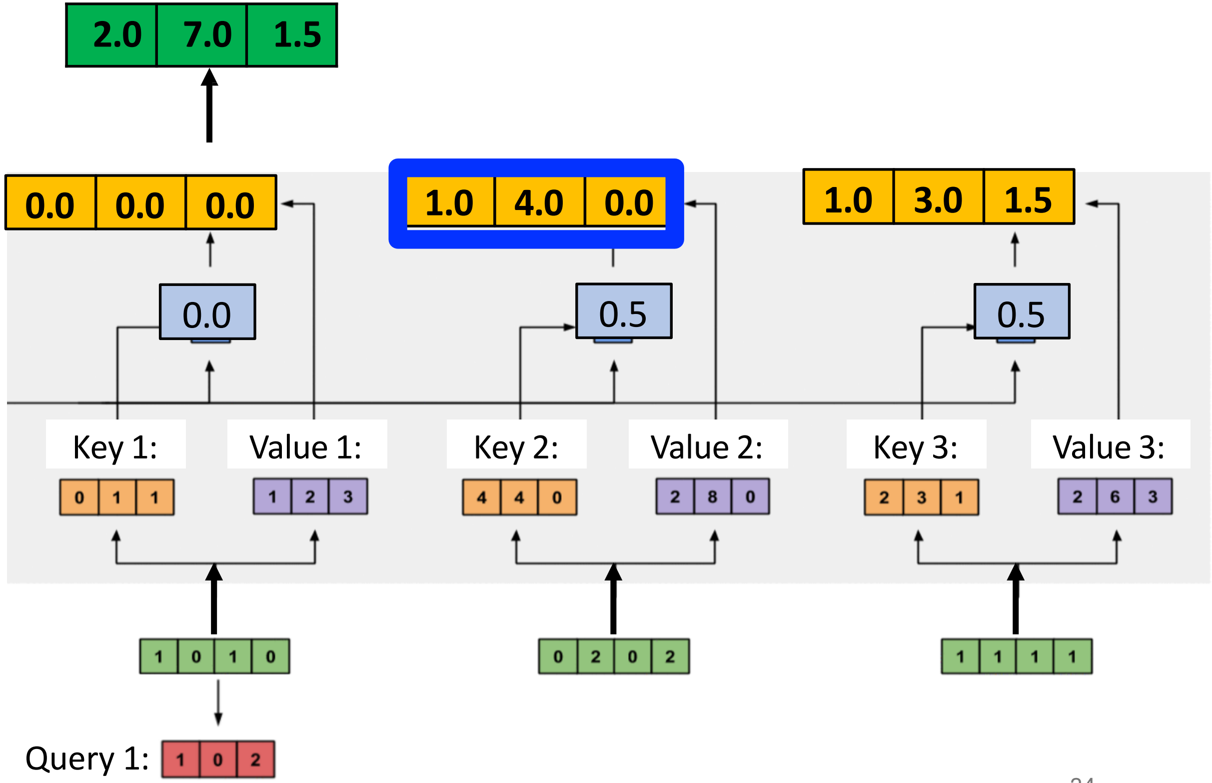
To which input(s) is input 1 most related?



https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example

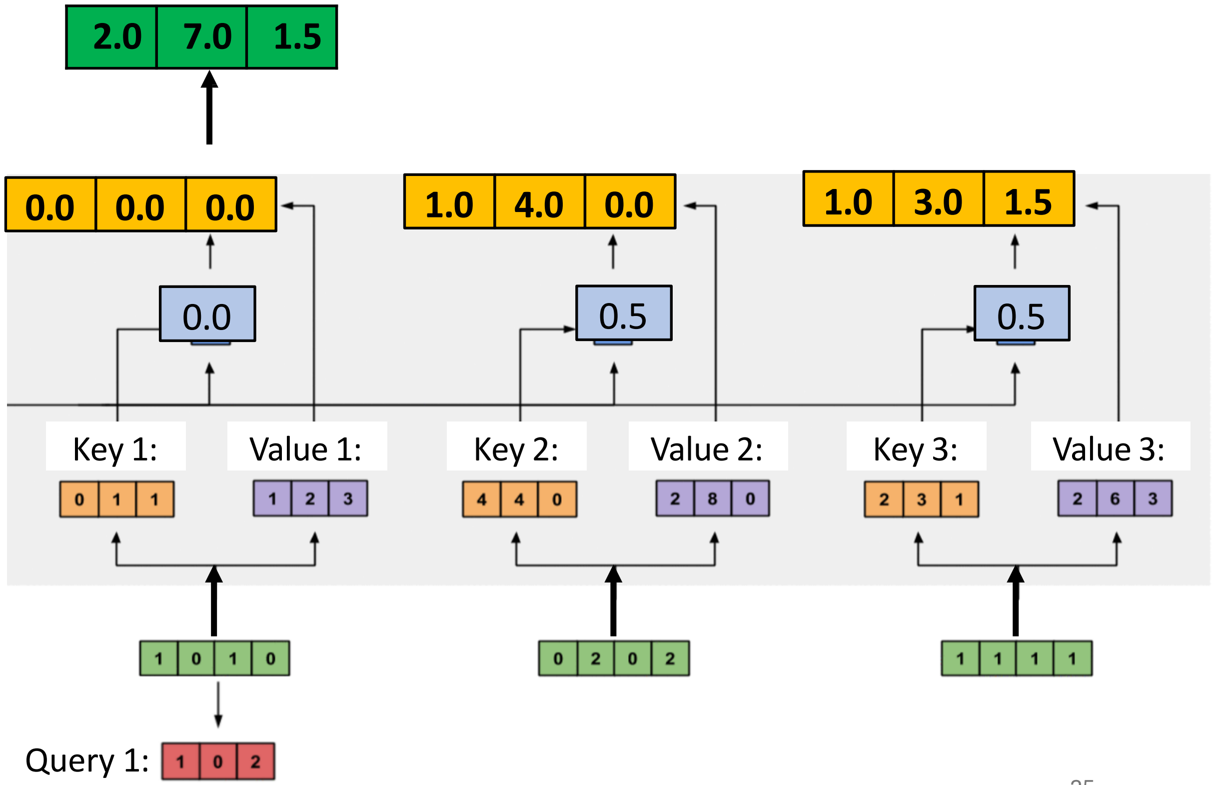
Attention weights amplify input representations (values) that we want to pay attention to and repress the rest



https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example

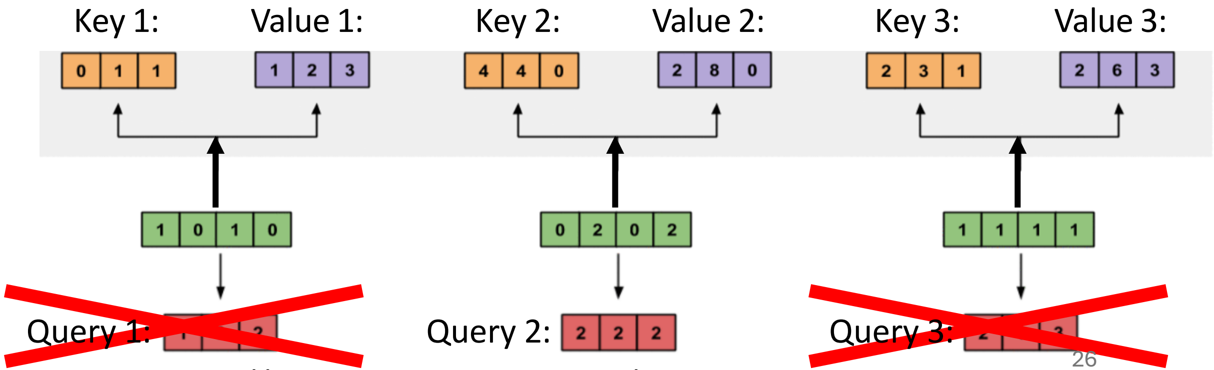
Attention weights amplify input representations (values) that we want to pay attention to and repress the rest



https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example

Repeat the same process for each remaining input token

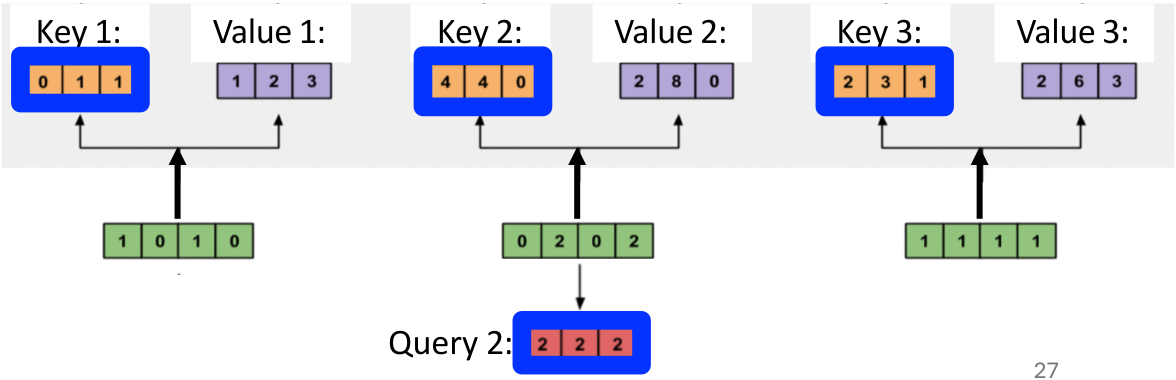


https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example

1. Compute attention weights
- Softmax resulting 3 scores from query x keys

To which input(s) is input 2 most related?

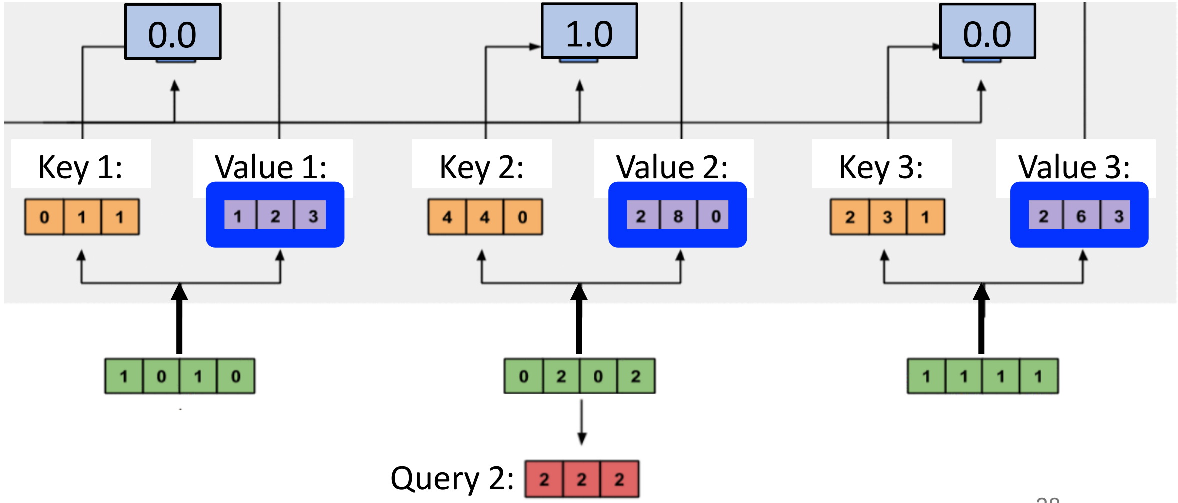


https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example

1. Compute attention weights
- Softmax resulting 3 scores from query x keys

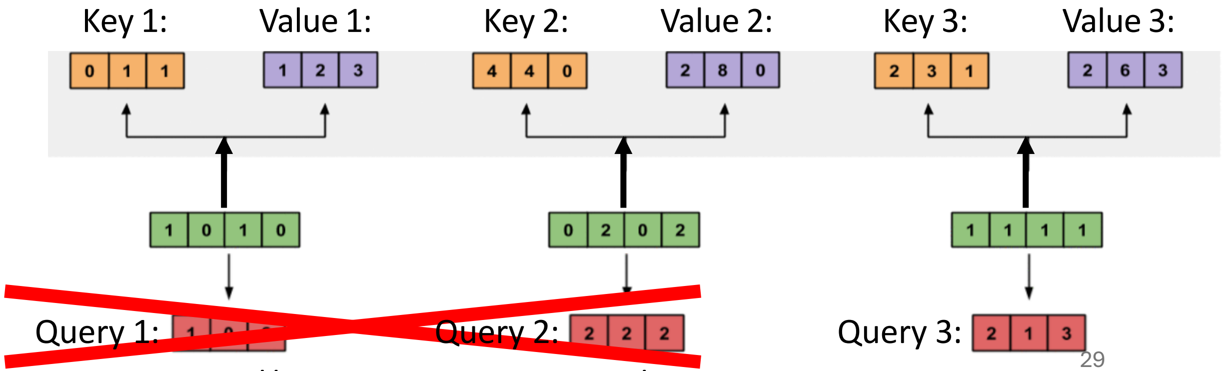
2. Compute weighted sum of values using attention scores



https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example

Repeat the same process for each remaining input token

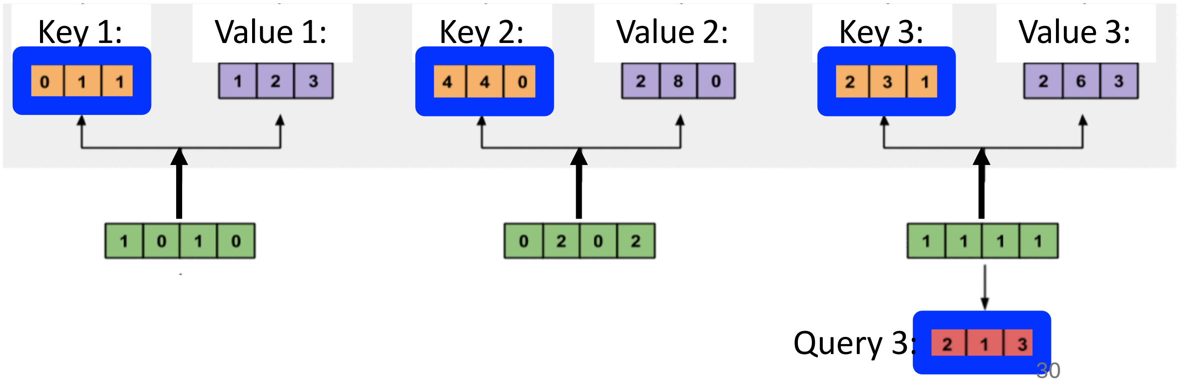


https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example

1. Compute attention weights
- Softmax resulting 3 scores from query x keys

To which input(s) is input 3 most related?

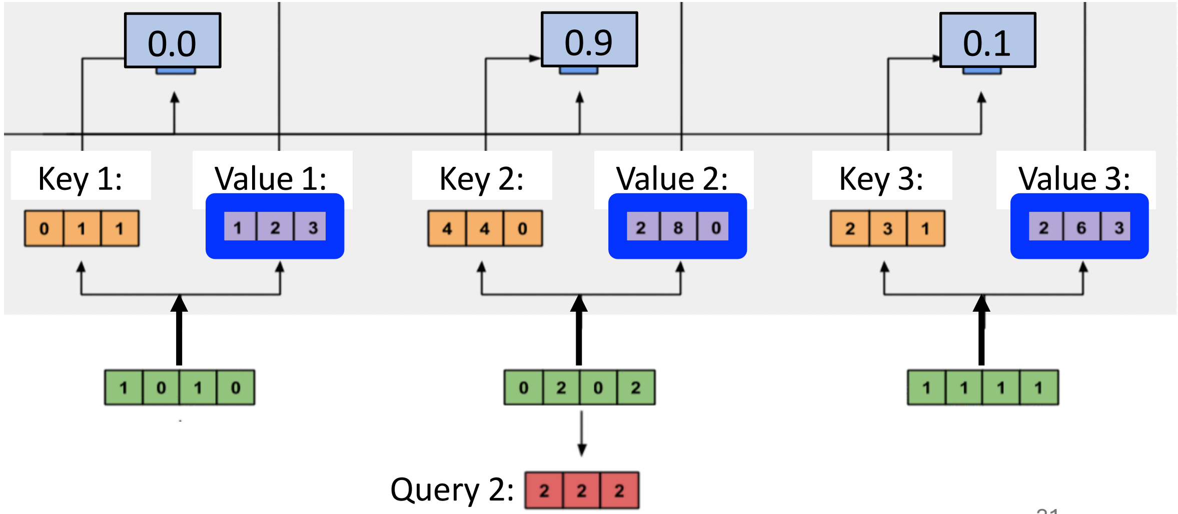


https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example

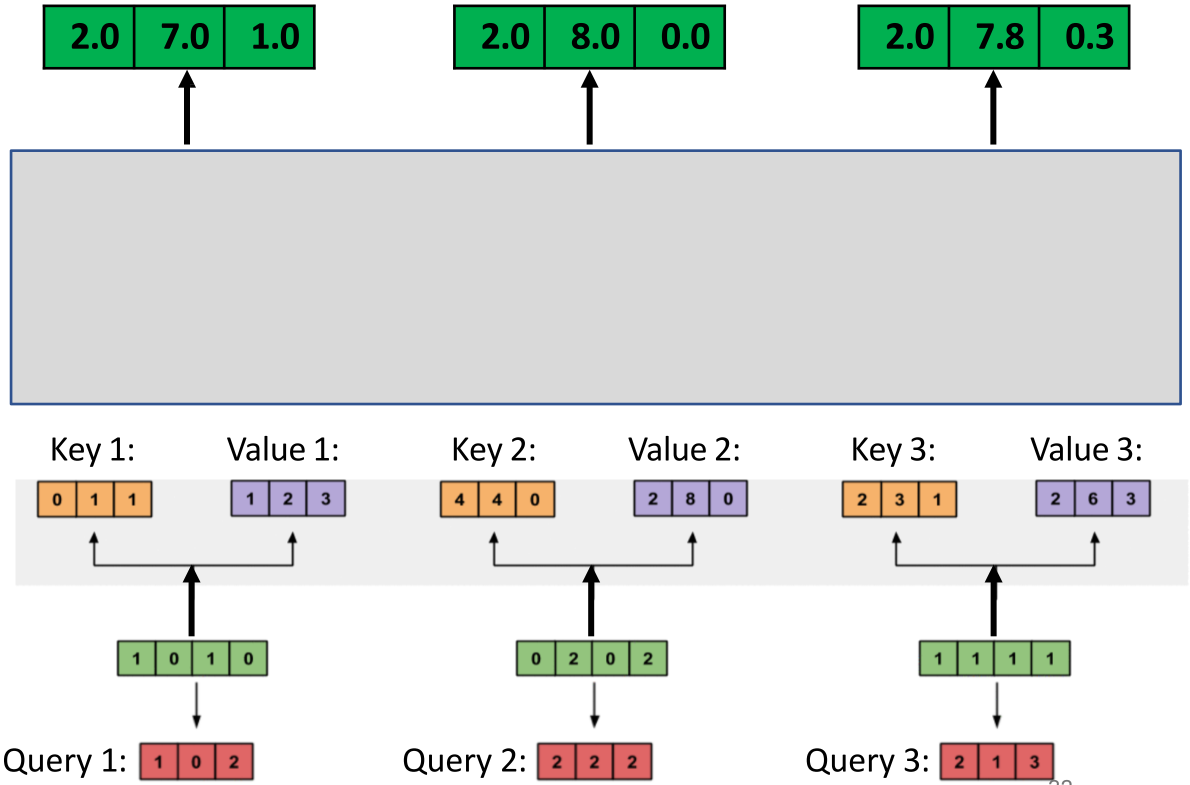
1. Compute attention weights
- Softmax resulting 3 scores from query x keys

2. Compute weighted sum of values using attention scores



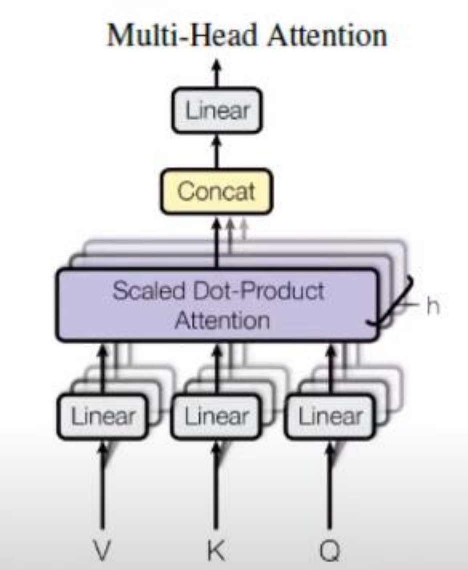
https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

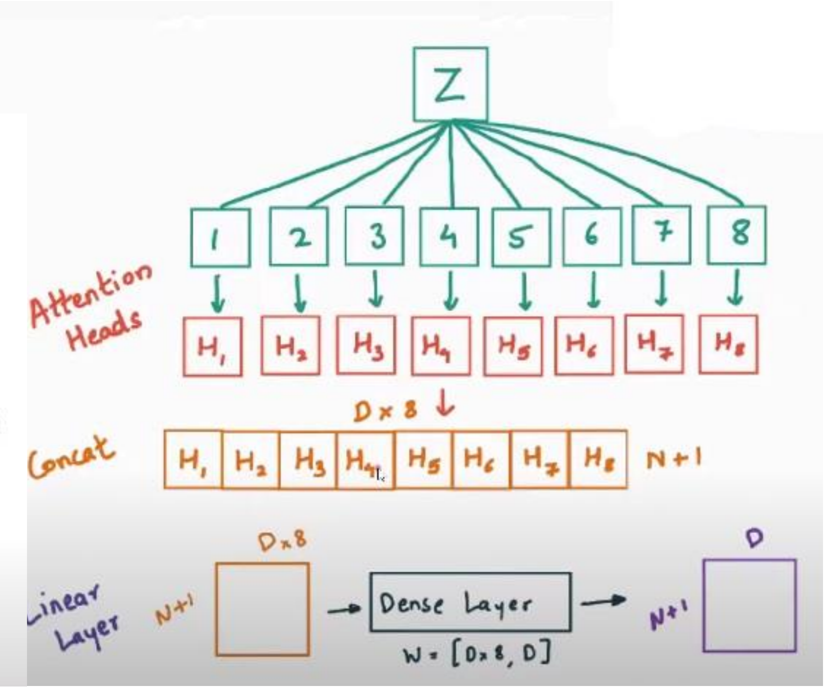
Computing Self-Attention: Example



https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

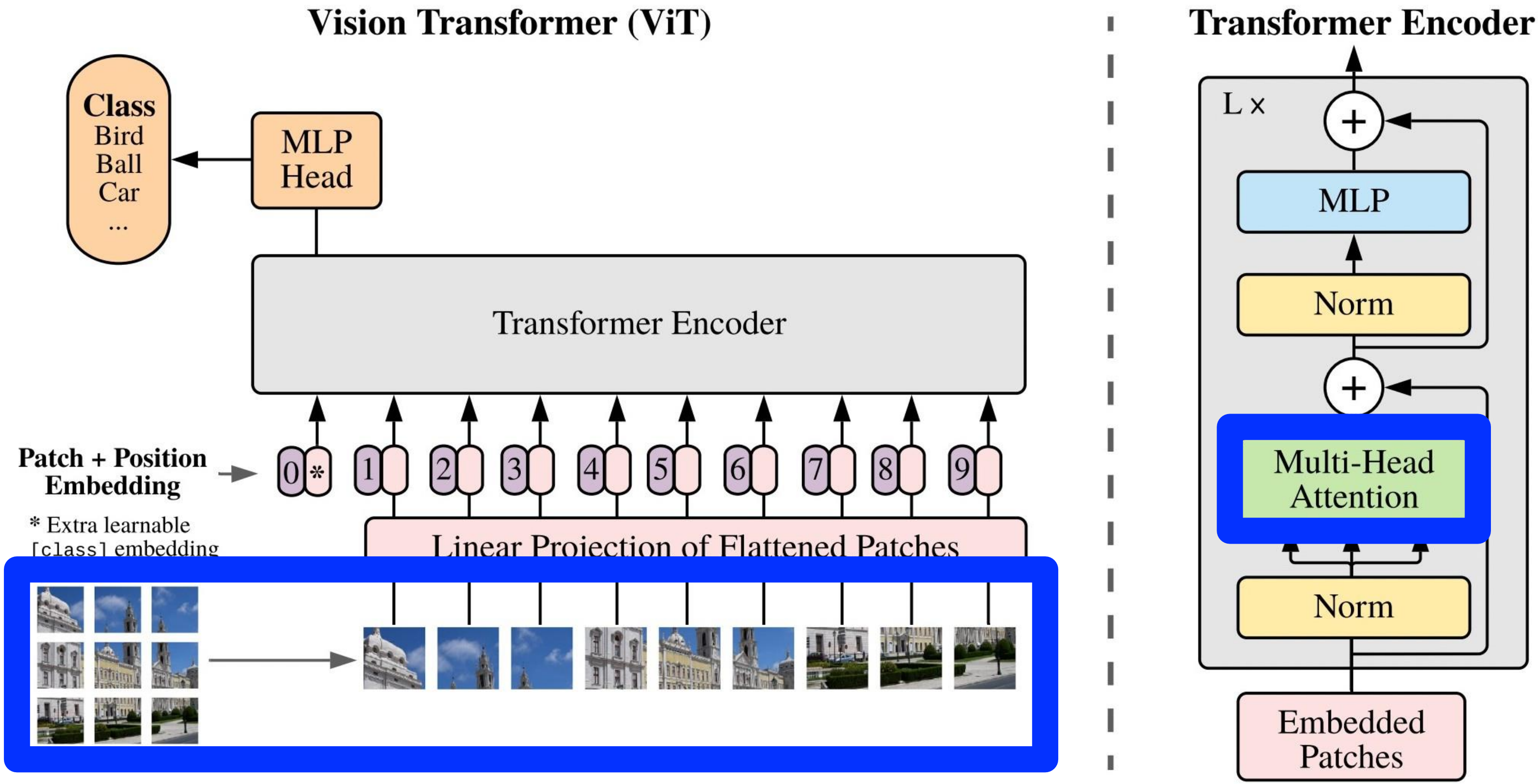
Computing Multi-Head Attention: Example





https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

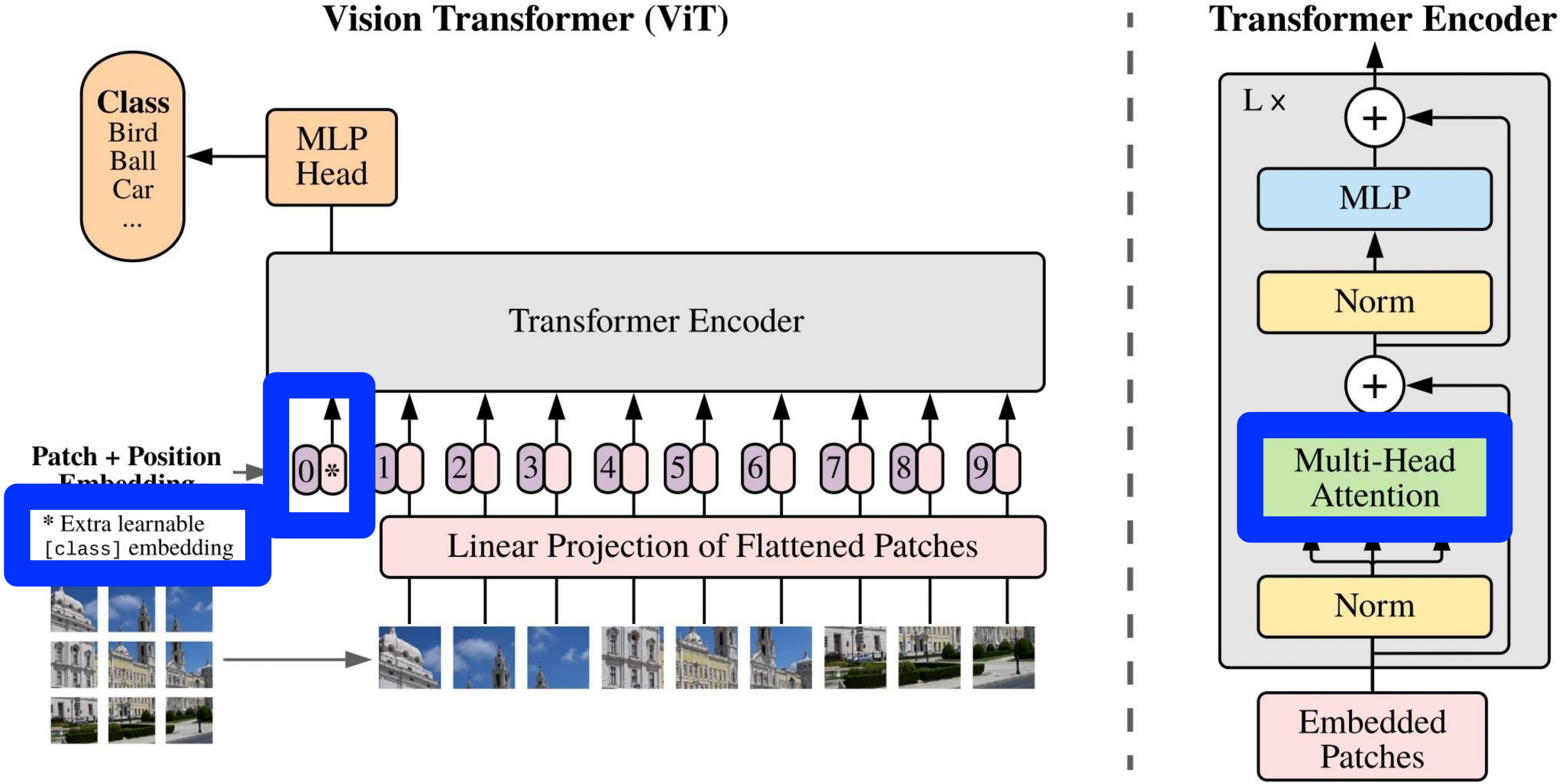
ViT Solution: Input Patches Instead of Pixels



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

ViT Solution: Use [CLS] for Image Classification

[CLS] token represents entire image



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

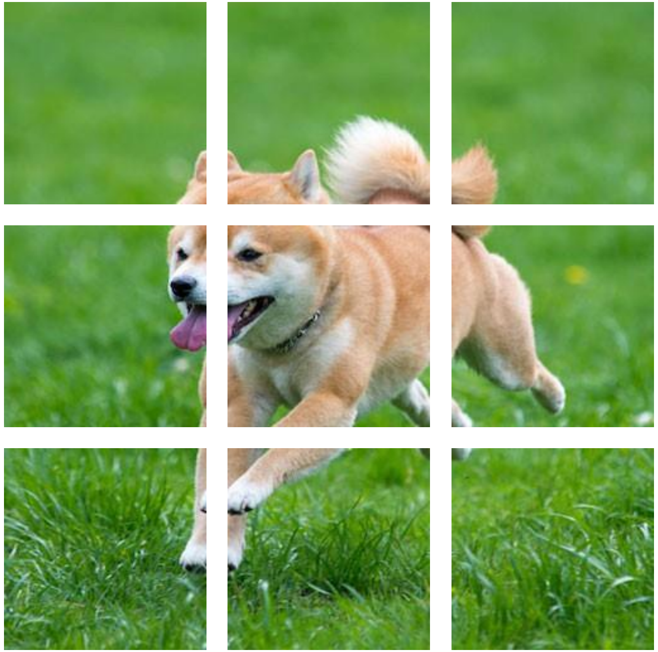
Standard Transformer on Patches

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

Dog image is free for use

Image Ref. http://wangshusen.github.io/

Standard Transformer on Patches



Split Image into Patches

- Here, the patches do not overlap.
- The patches can overlap.
- User specifies:
  - patch size, e.g., 16×16;
  - stride, e.g., 16×16.

9 input patches, each of shape 3x16x16

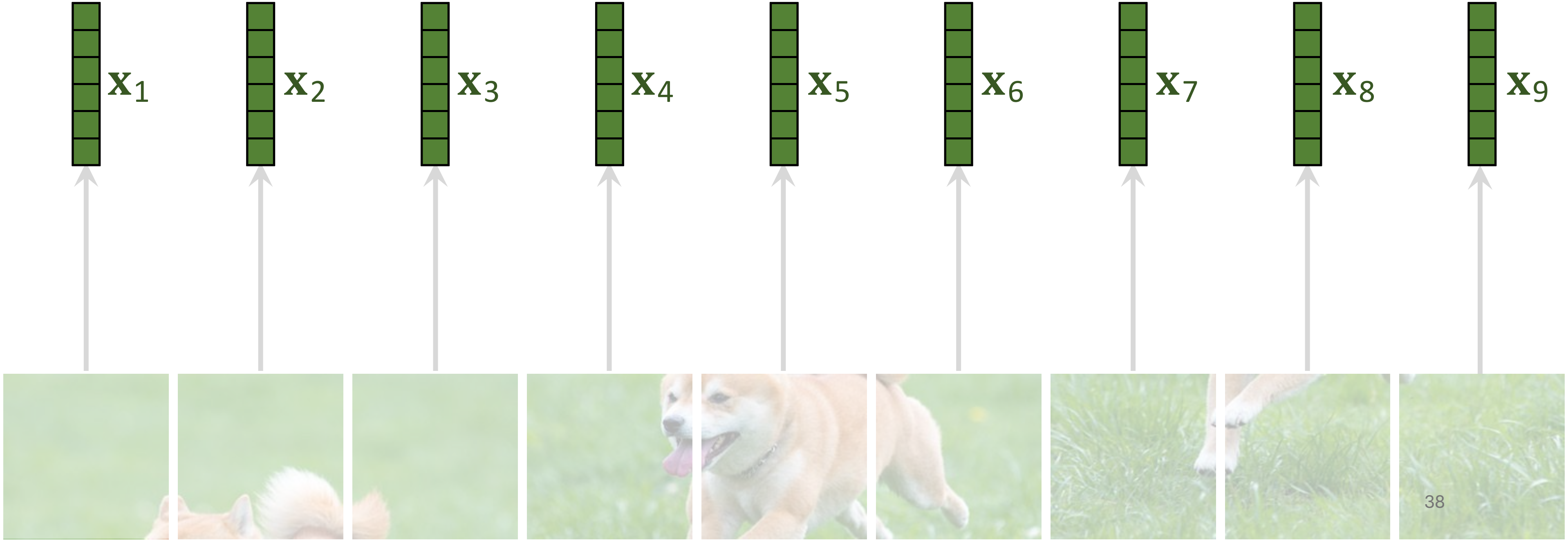
Dosovitskiy et al, “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”, ICLR 2021

Image Ref. http://wangshusen.github.io/

Vision Transformer Architecture Explained:

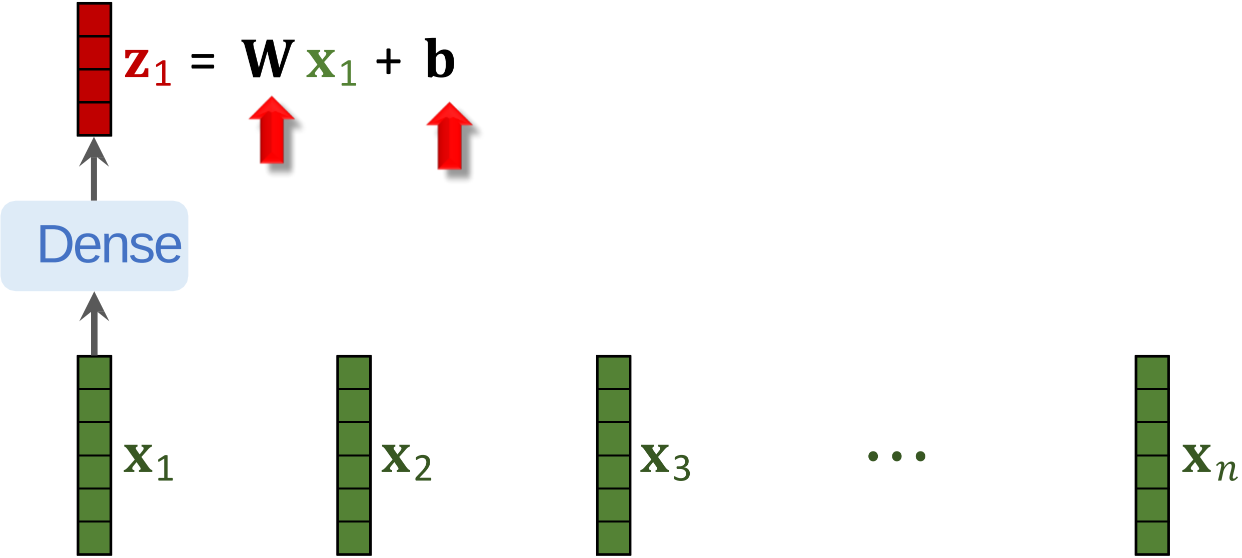
Vectorization

If the patches are $d_1 \times d_2 \times d_3$ tensors, then the vectors are $d_1 d_2 d_3 \times 1$.

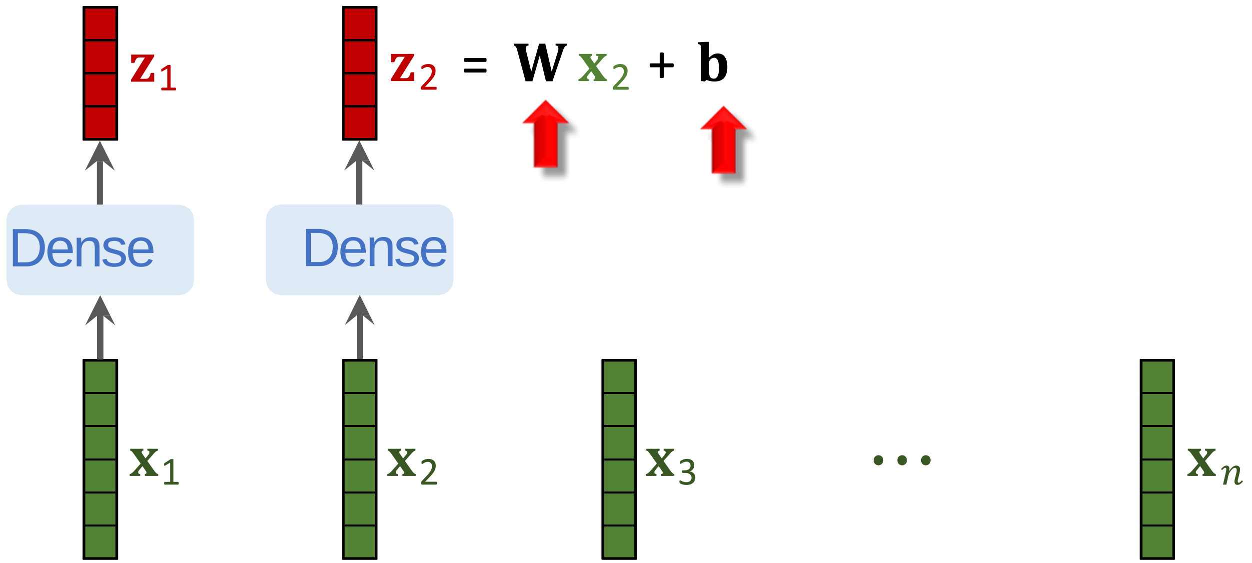


Vision Transformer Architecture Explained:

$\mathbf{z}_1 = \mathbf{W}\,\mathbf{x}_1 + \mathbf{b}$



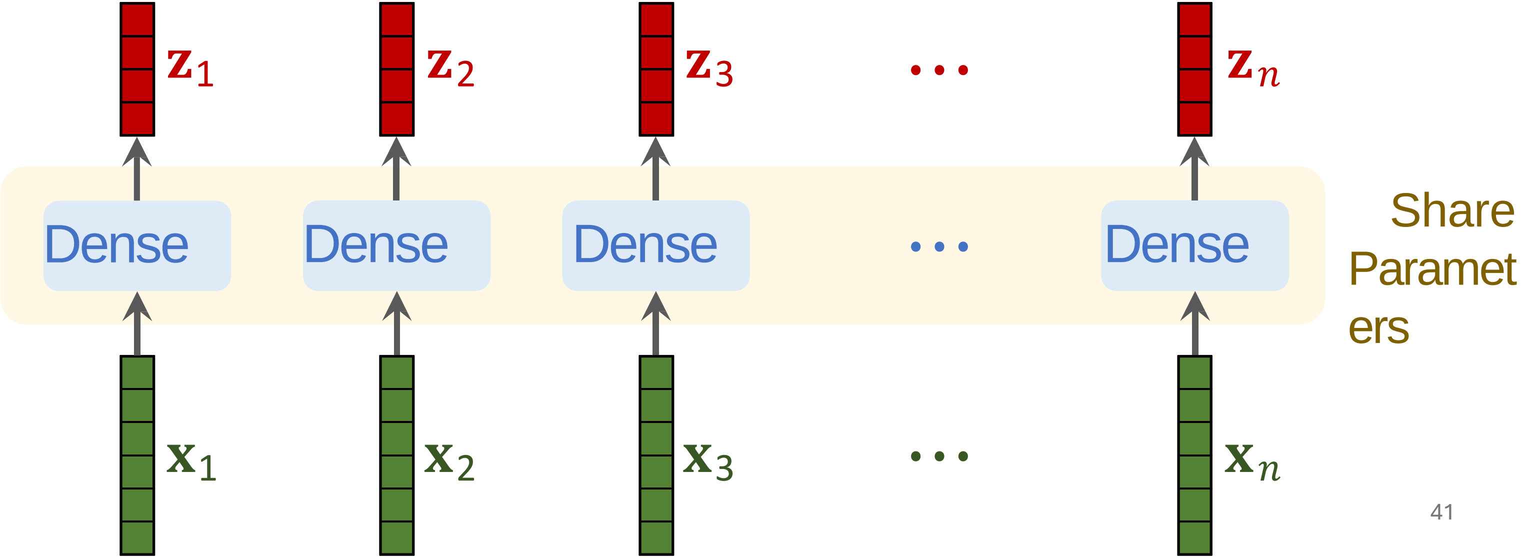
Vision Transformer Architecture Explained:



Vision Transformer Architecture Explained:

Linear projection to D-dimensional vector

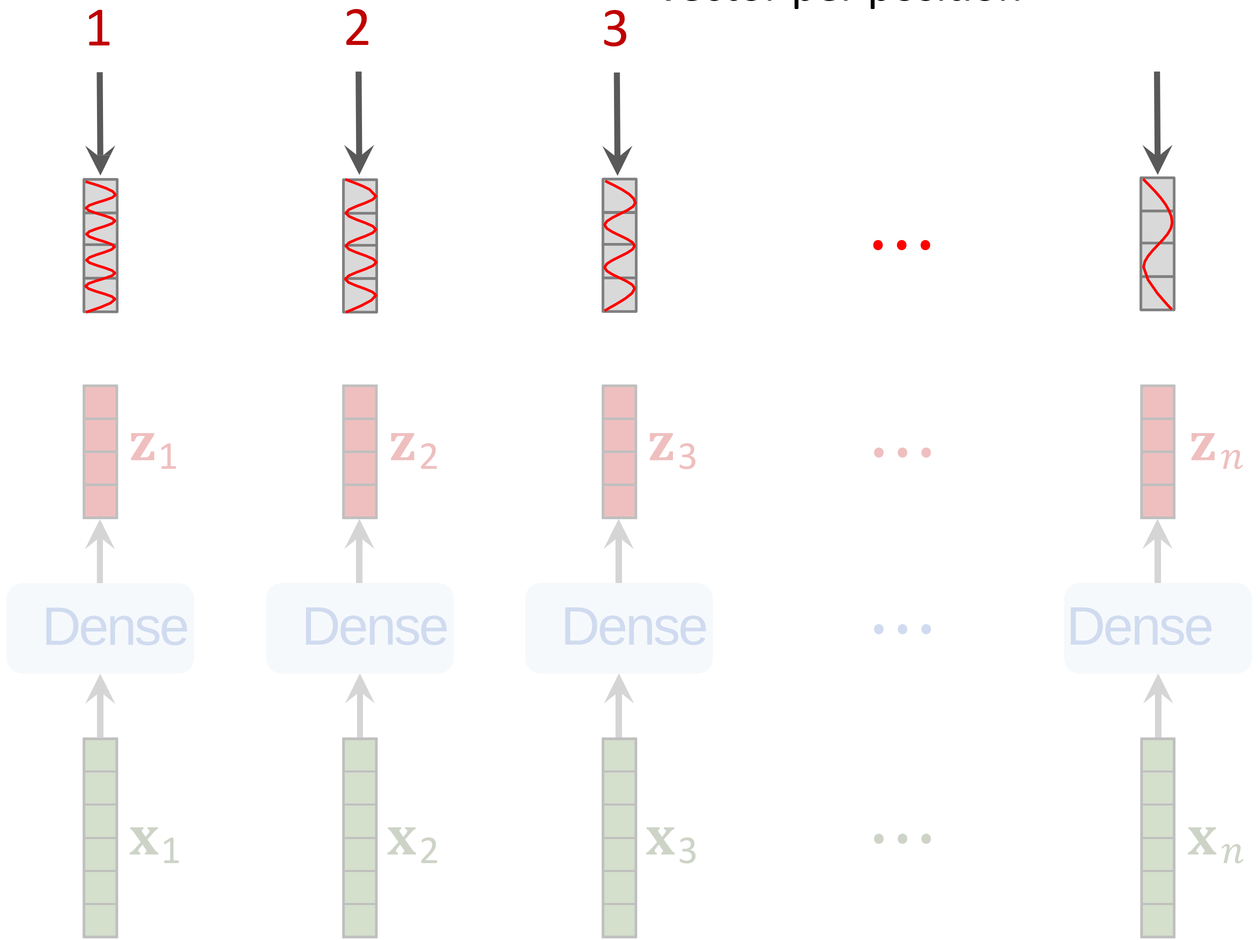
N input patches, each of shape 3x16x16



Vision Transformer Architecture Explained:

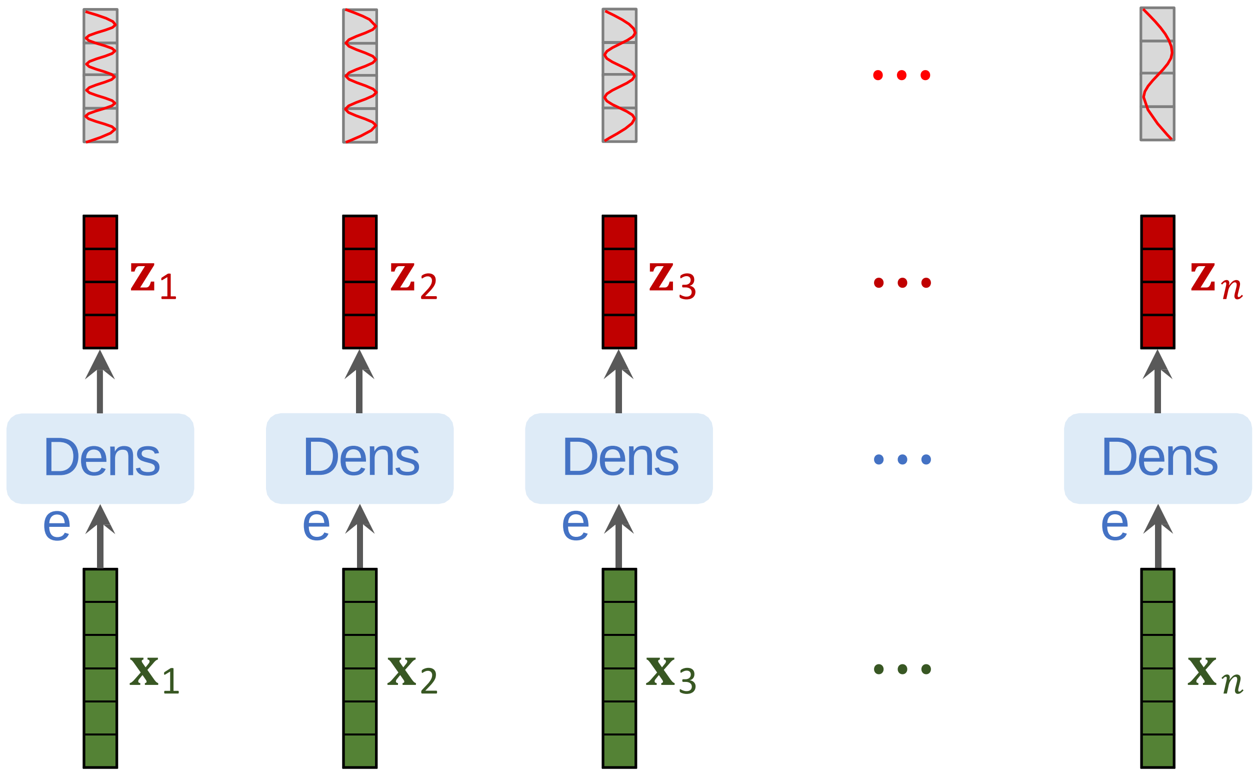
Add positional embedding: learned D- dim vector per position

Positional Encoding:



Vision Transformer Architecture Explained:

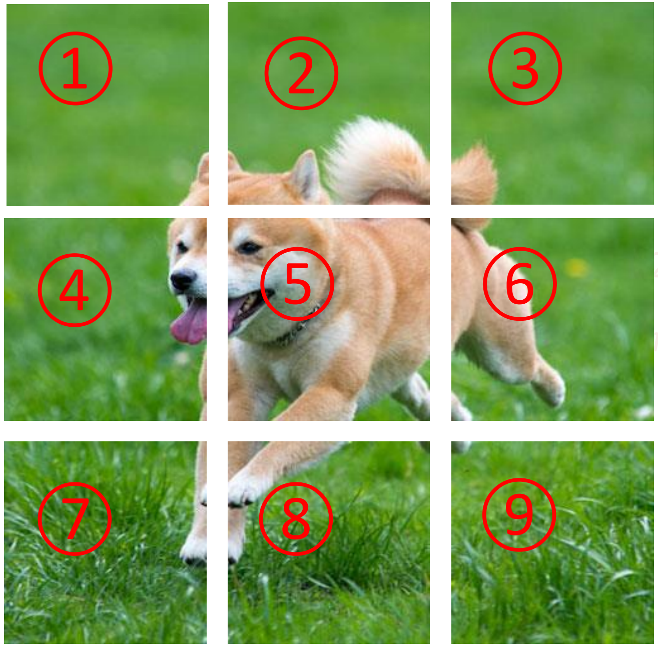
Add positional encoding vectors to $\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_n$.



Vision Transformer Architecture Explained:

Add positional encoding vectors to $\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_n$. (Why?)

- 3% drop in accuracy is observed, if we do not apply positional encoding
- Positional encoding encoding can be 1D or 2D, but no significant improvement is found in using 2D encoding, hence the paper uses 1D position encoding



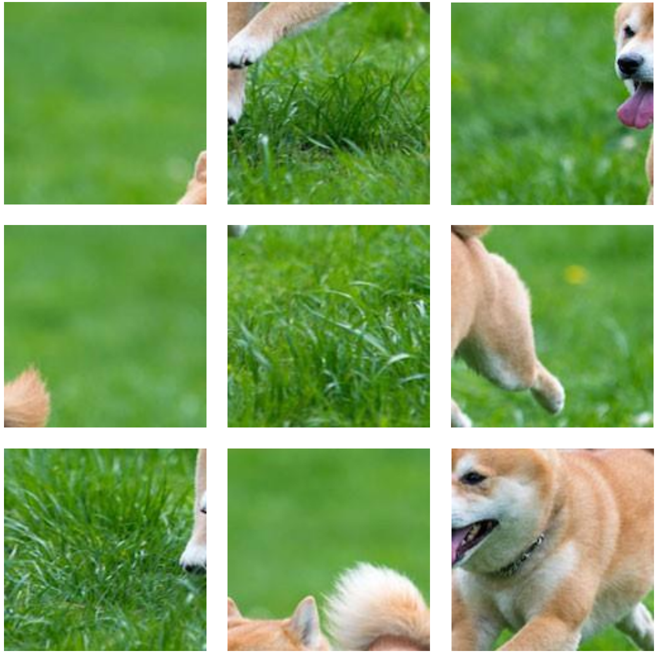
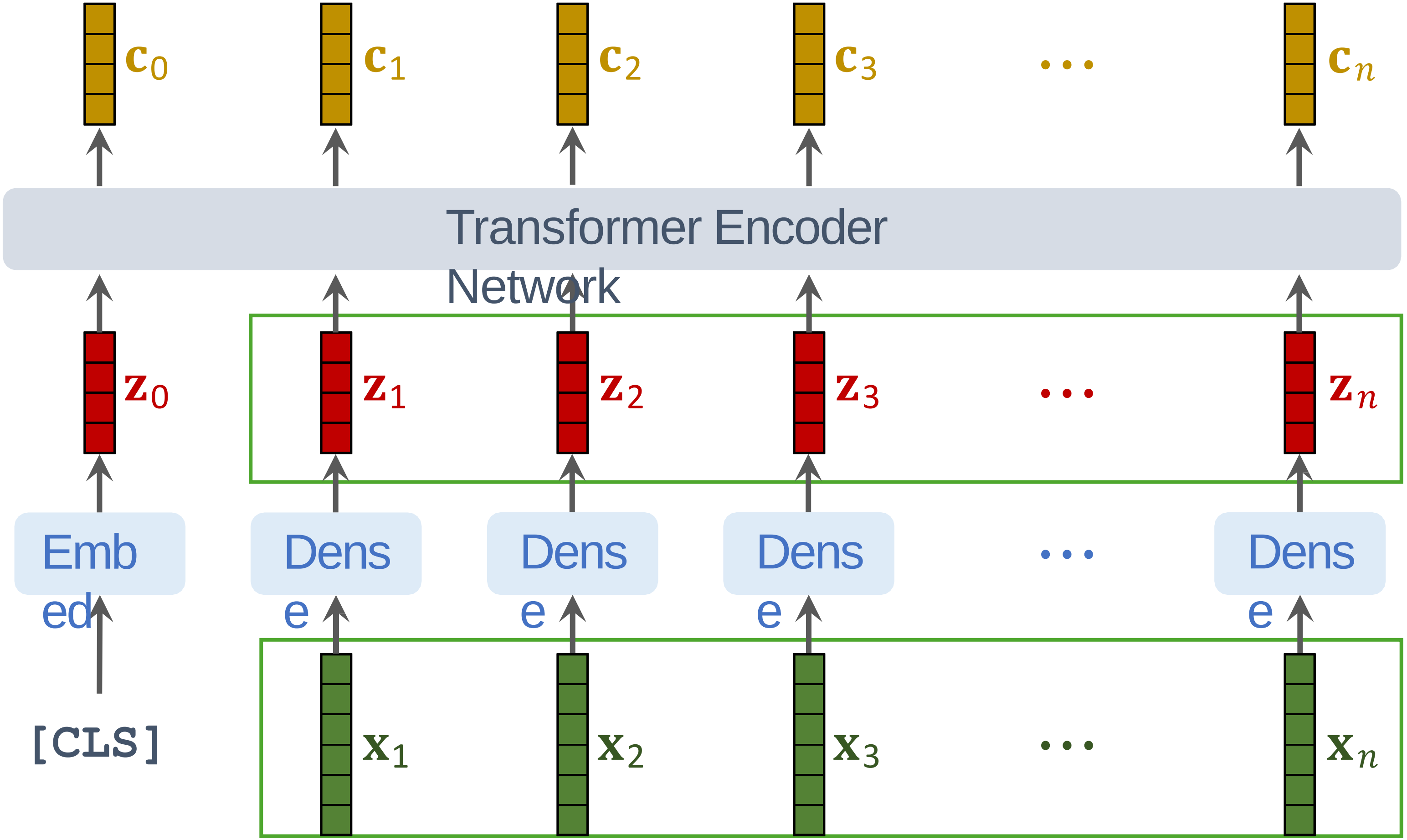
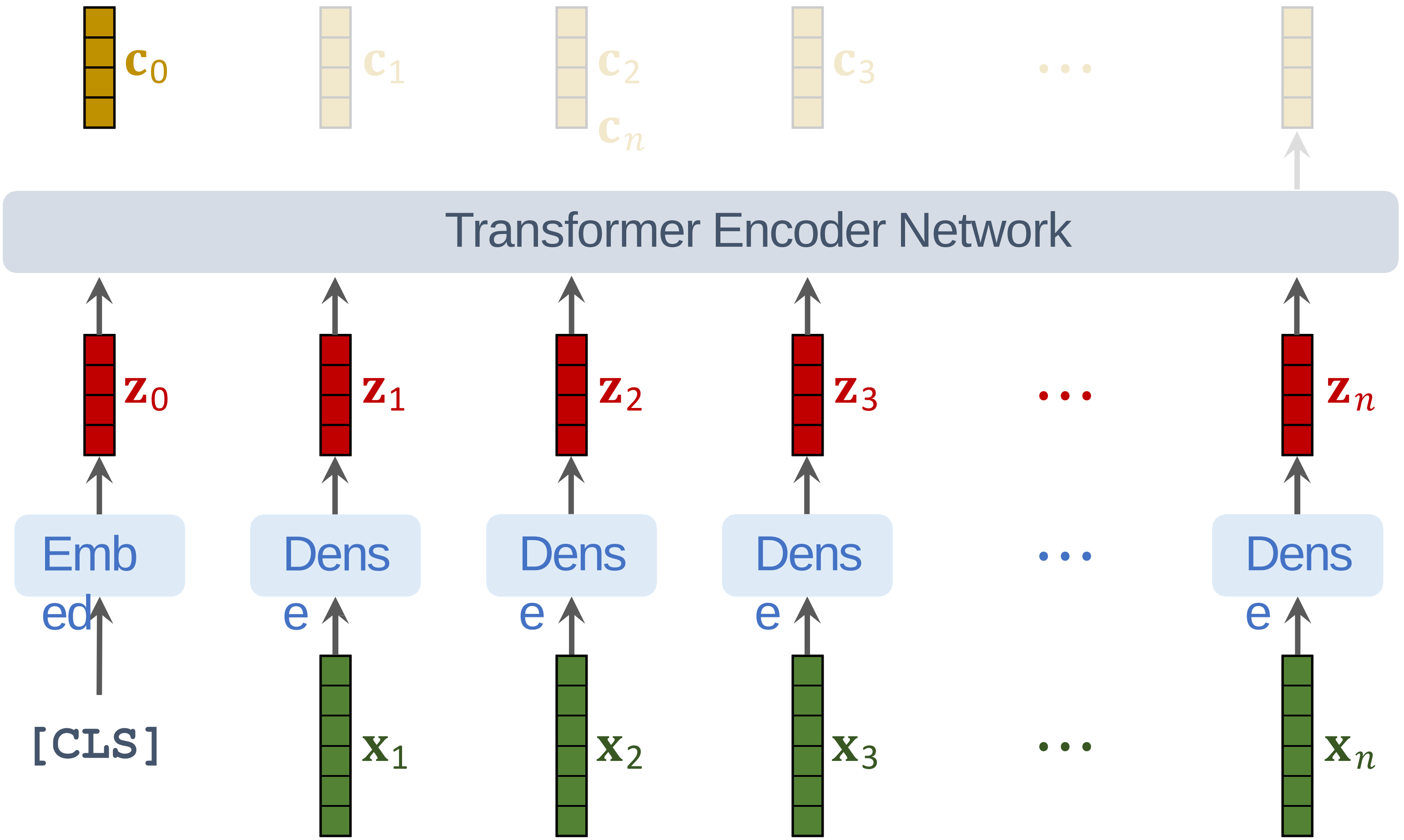


Image Ref. http://wangshusen.github.io/

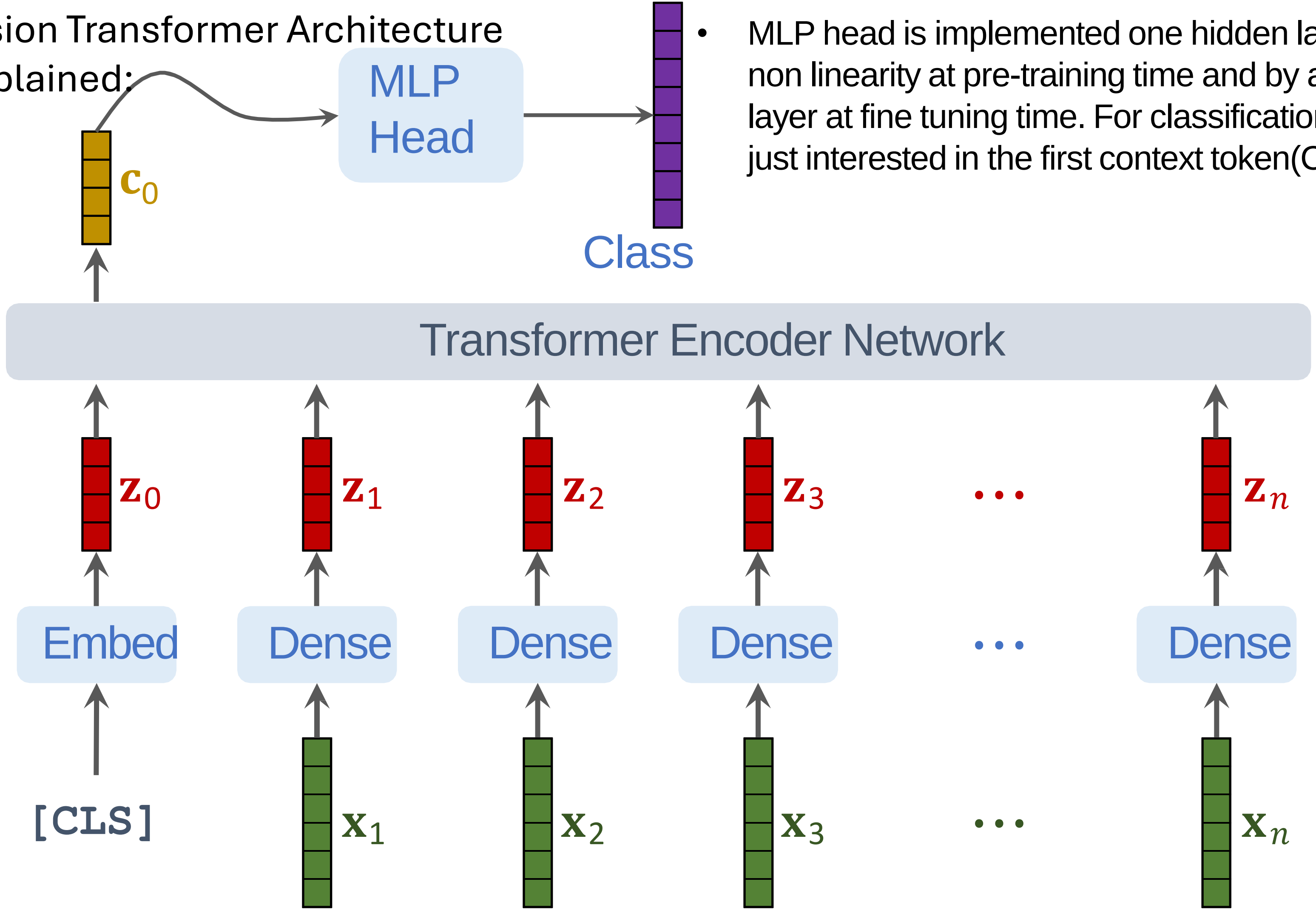
Vision Transformer Architecture Explained:



Vision Transformer Architecture Explained:



Vision Transformer Architecture Explained:



- MLP head is implemented one hidden layer and tanh as non linearity at pre-training time and by a single linear layer at fine tuning time. For classification task we are just interested in the first context token(C0).

Vision Transformer Architecture

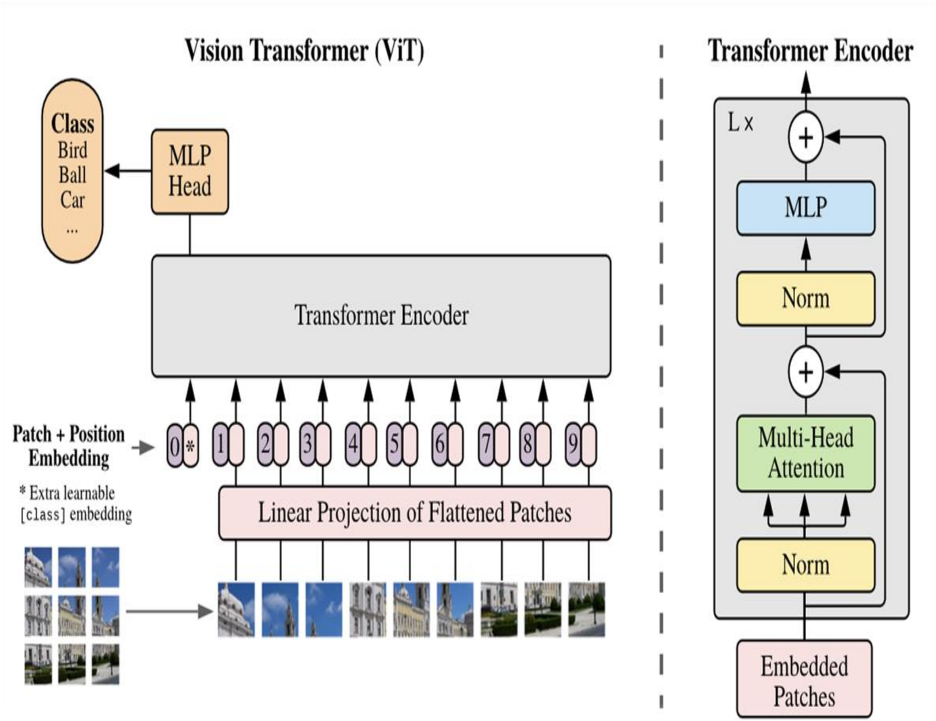


Image Patches $\equiv$ Tokens (Words) in NLP

$x \in \mathbb{R}^{H \times W \times C} \rightarrow$ image

$x_p \in \mathbb{R}^{N \times (P^2 C)} \rightarrow$ sequence of flattened 2D patches (reshape x)

$P \times P \rightarrow$ resolution of each image patch

$N = \dfrac{HW}{P^2} \rightarrow$ resulting number of patches

$D \rightarrow$ latent vector size

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

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

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

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

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

Training:

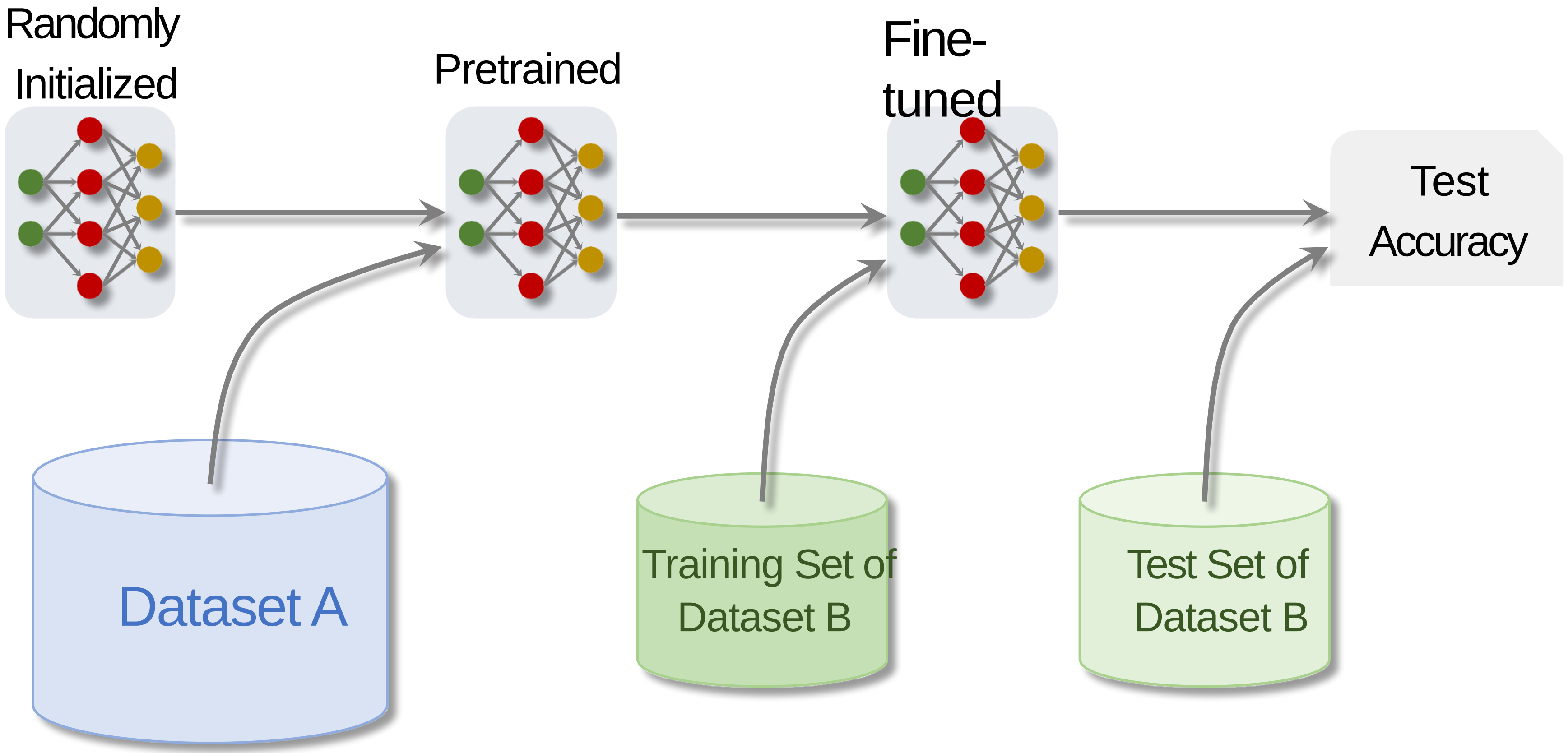


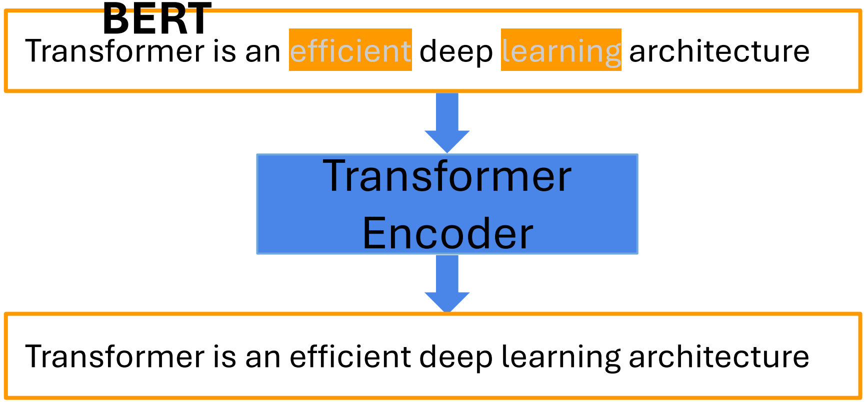
Image Ref. http://wangshusen.github.io/

Self-supervision:

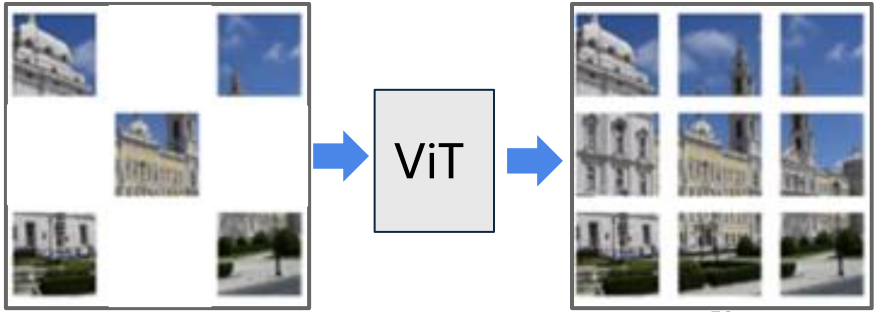
Most of Transformers success in NLP is the result of large-scale self-supervised pre-training where Transformer is trained on massive unlabelled data from the web.

Using masked work prediction technique that were used in BERT(randomly masking words in input sentence), ViT designers also tried the same technique where they masked 50% of patches(masked patch prediction) but achieved less performance than supervised pre-training(79.9% ACC on ImageNet while supervised pre-training is ~85%).

Masked-word prediction in BERT

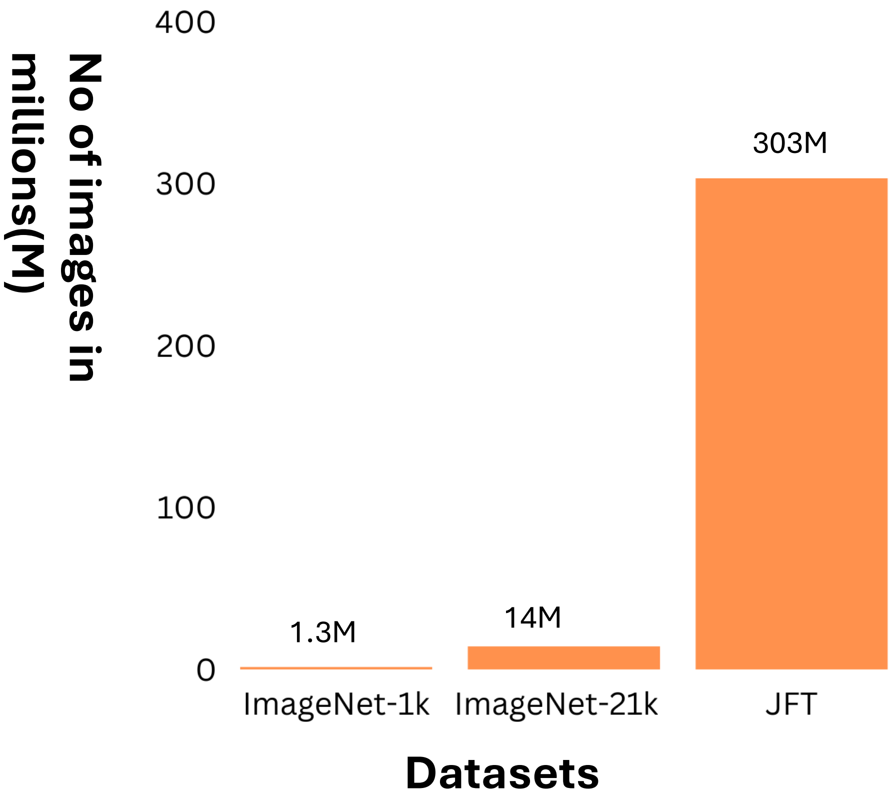


Masked patch prediction ViT



Training Dataset Vision Transformer

Vision Transformer(ViT) was pre-trained on 3 datasets of varying size and scale.



Pre-training datasets

- ImageNet-1K: 1.3M images, 1K classes
- ImageNet-21k: 14M images, 21K classes
- JFT: 303M images, 18K classes

ImageNet-1k and ImageNet-21K are also used for fine-tuning!

Pre-Training Vision Transformer

Vision Transformer(ViT) was pre-trained with same configurations of as BERT.

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

Table 1: Details of Vision Transformer model variants.

All the models are pre-trained using ADAM optimizer with batch size of 4096

Fine-Tuning Vision Transformer

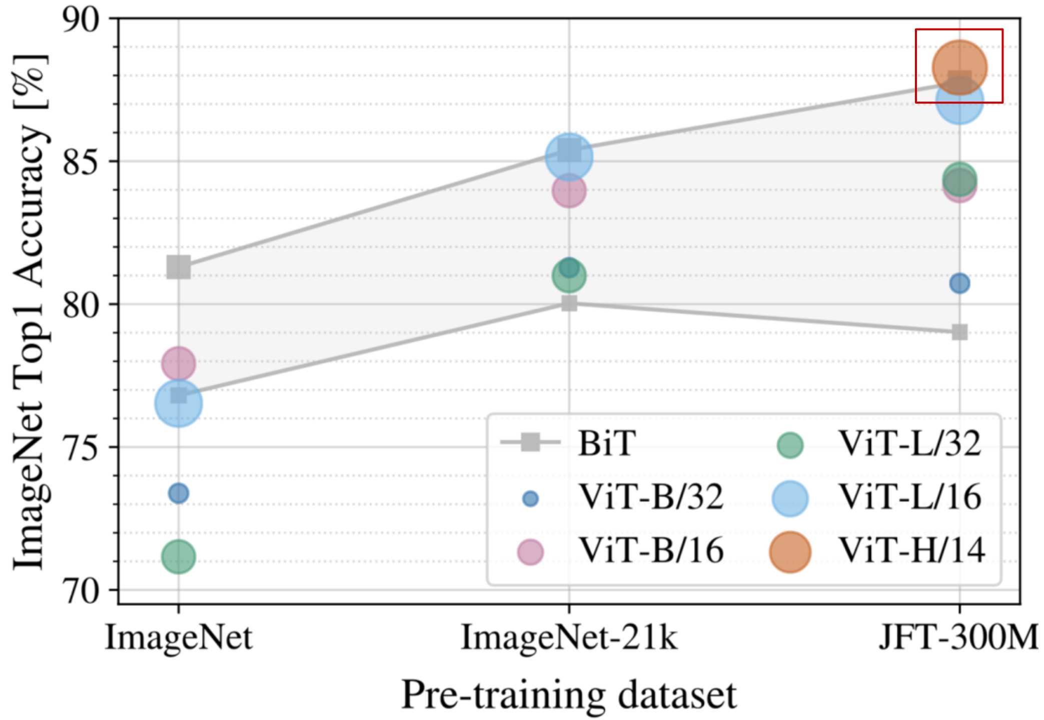
- All the models are fine tuned using SGD optimizer with monentum and batch size of 512 for ViT-L/16 and 518 for ViT-H/14.

	Ours-JFT (ViT-H/14)	Ours-JFT (ViT-L/16)	Ours-I21k (ViT-L/16)	BiT-L (ResNet152x4)	Noisy Student (EfficientNet-L2)
ImageNet	**88.55** ± 0.04	87.76 ± 0.03	85.30 ± 0.02	87.54 ± 0.02	88.4/88.5*
ImageNet ReaL	**90.72** ± 0.05	90.54 ± 0.03	88.62 ± 0.05	90.54	90.55
CIFAR-10	**99.50** ± 0.06	99.42 ± 0.03	99.15 ± 0.03	99.37 ± 0.06	–
CIFAR-100	**94.55** ± 0.04	93.90 ± 0.05	93.25 ± 0.05	93.51 ± 0.08	–
Oxford-IIIT Pets	**97.56** ± 0.03	97.32 ± 0.11	94.67 ± 0.15	96.62 ± 0.23	–
Oxford Flowers-102	99.68 ± 0.02	**99.74** ± 0.00	99.61 ± 0.02	99.63 ± 0.03	–
VTAB (19 tasks)	**77.63** ± 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

Image Classification Accuracies

- Pretrained on ImageNet (small), ViT is slightly worse than ResNet.
- Pretrained on ImageNet-21K (medium), ViT is comparable to ResNet.
- Pretrained on JFT (large), ViT is slightly better than ResNet.

Vision Transformer vs SOTA CNN(ResNet)

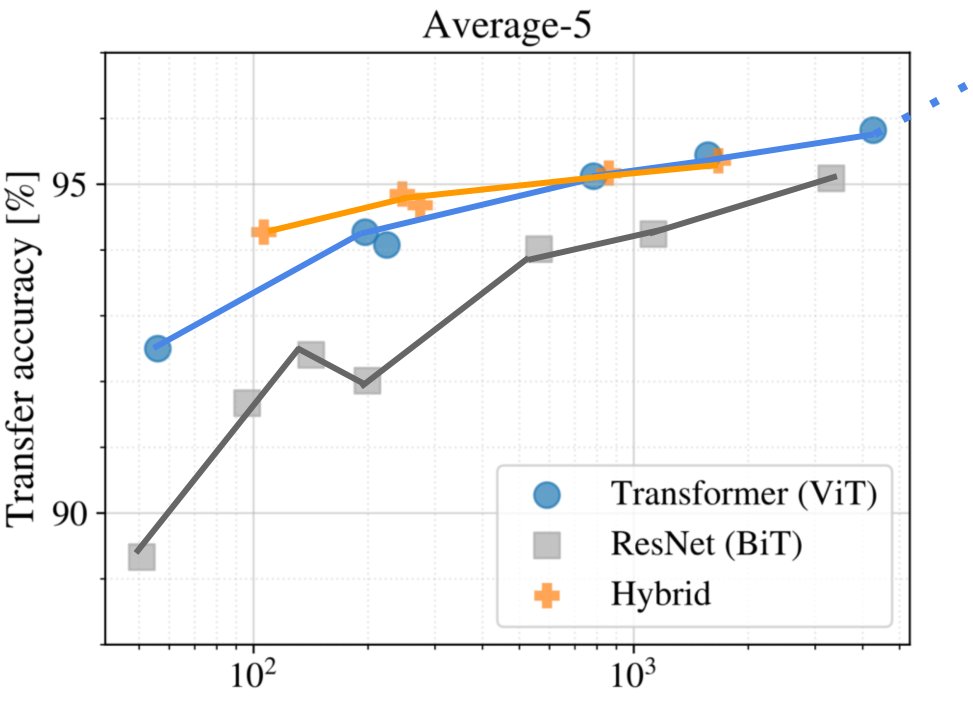


B: Base, L: Large, H: Huge
14, 16, 32: Patch size(the smaller patch size, the more the patches, and the bigger the model) >> **N = HW/PP**
Ex: ViT-B/16: Base ViT with **16x16** patch size

Results

- On small pre-training dataset(ImageNet-1k, 1.3M images), ResNet performs better than ViT due to CNN spatial inductive biases that compensate for small dataset.
- On medium pre-training dataset(Imagenet-21k, 14M images), ViTs and ResNet performance are almost similar although ViTs perform slightly better.
- On large pre-training dataset(JFT, 303M images), large ViT outperforms ResNet and show no sign of plateau.

Vision Transformer vs SOTA CNN(ResNet)



Pre-training compute(in ExaFLOPs)

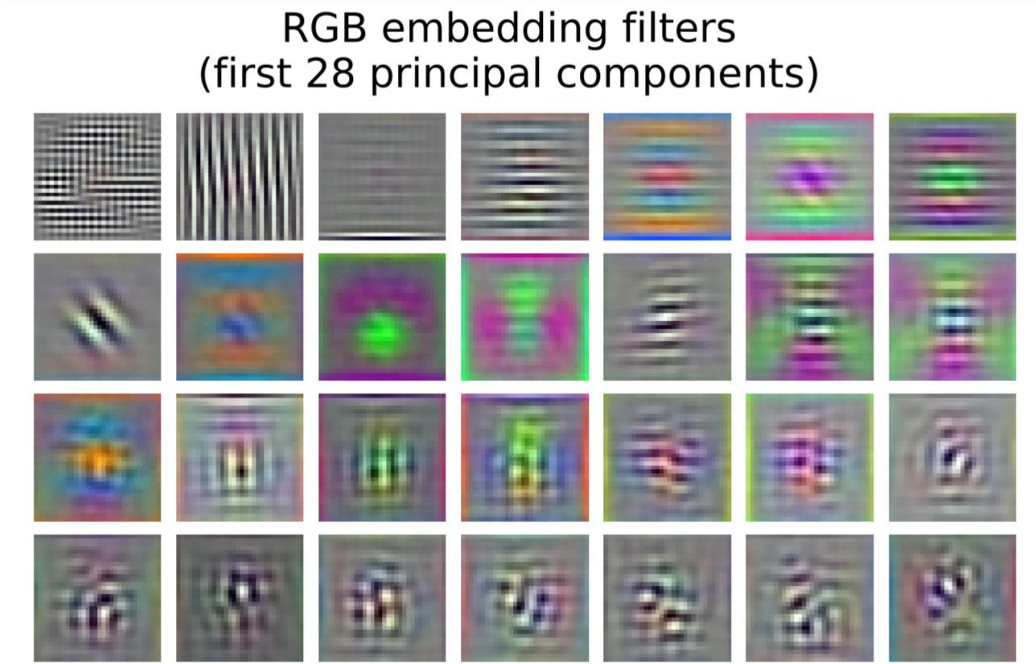
Exa: 10^{18}, 1 FLOP = 1 multiply-add(wx+b) per second
FLOPs: floating point operations per second

Pre-training compute

- ViT clearly outperforms ResNet on performance/compute trade-off.
- ViT uses approximately 2-4x less compute to achieve the same transfer accuracy(average of all downstream datasets).
- Hybrid(CNN+ViT) slightly outperforms ViT on relatively small compute, but vanishes on large compute budget.
- ViT shows extreme scaling behavior. Its performance doesn't seem to saturate for increased compute.

Inspecting ViT Representation

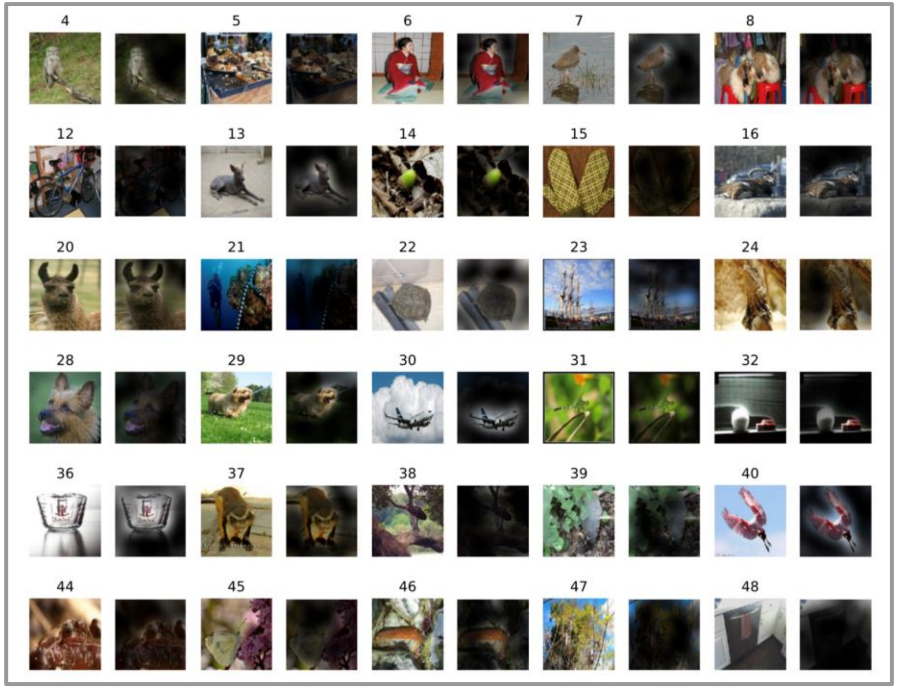
Vision Transformer shows remarkable performance when trained on massive datasets.



The visualized linear embedding of flattened patches shows that the first layer of ViT(linear projection) learns the low level features(such as edges, blobs) of the input image much like ConvNets do!

Inspecting ViT Representation

Vision Transformer shows remarkable performance when trained on massive datasets. How does it processes images internally?



On global level, ViT attends to the meaningful part of the image and ignore the rest.

Conclusion

Vison Transformer matches or exceeds the state of art on many image classification datasets, while being relatively cheap to pre-train.

While initial results are encouraging, we need to analyse the performane of ViT on other computer vison tasks, such as detection and segmentation.

End of the video

Image: Google AI blog

Thank You!