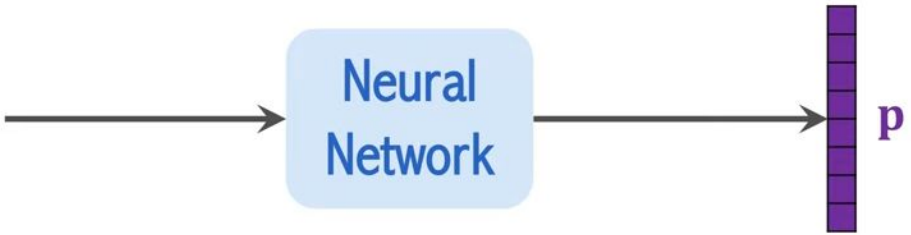
Vision Transformers

(for image classification)

Image Classification



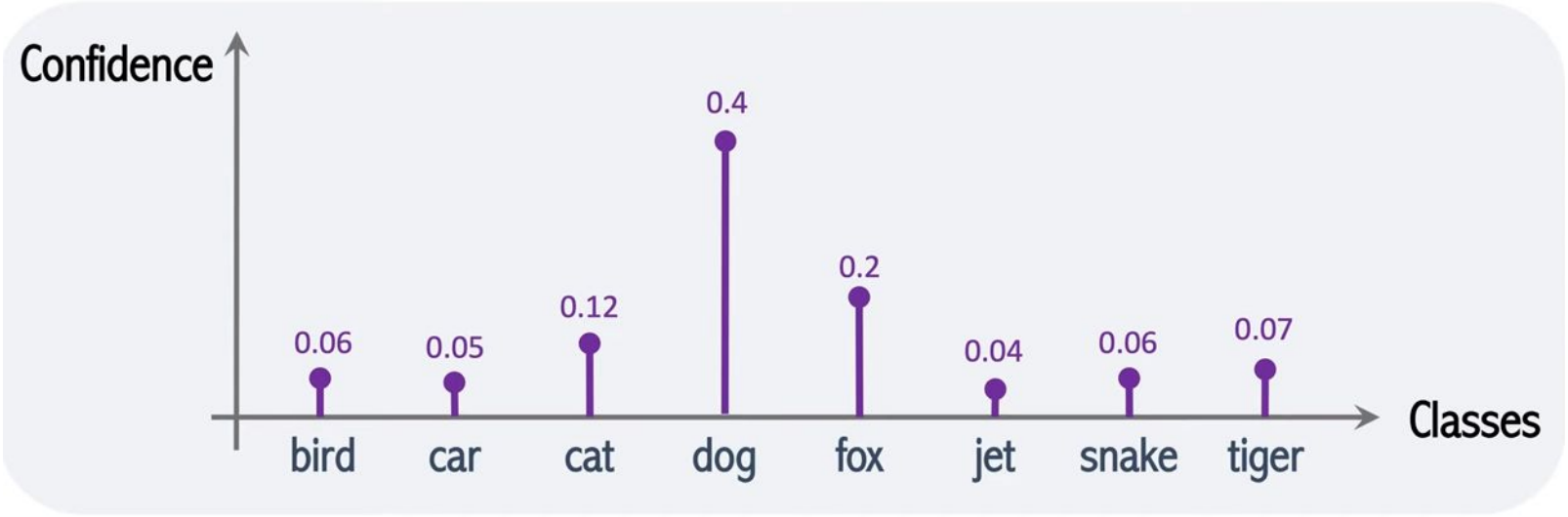
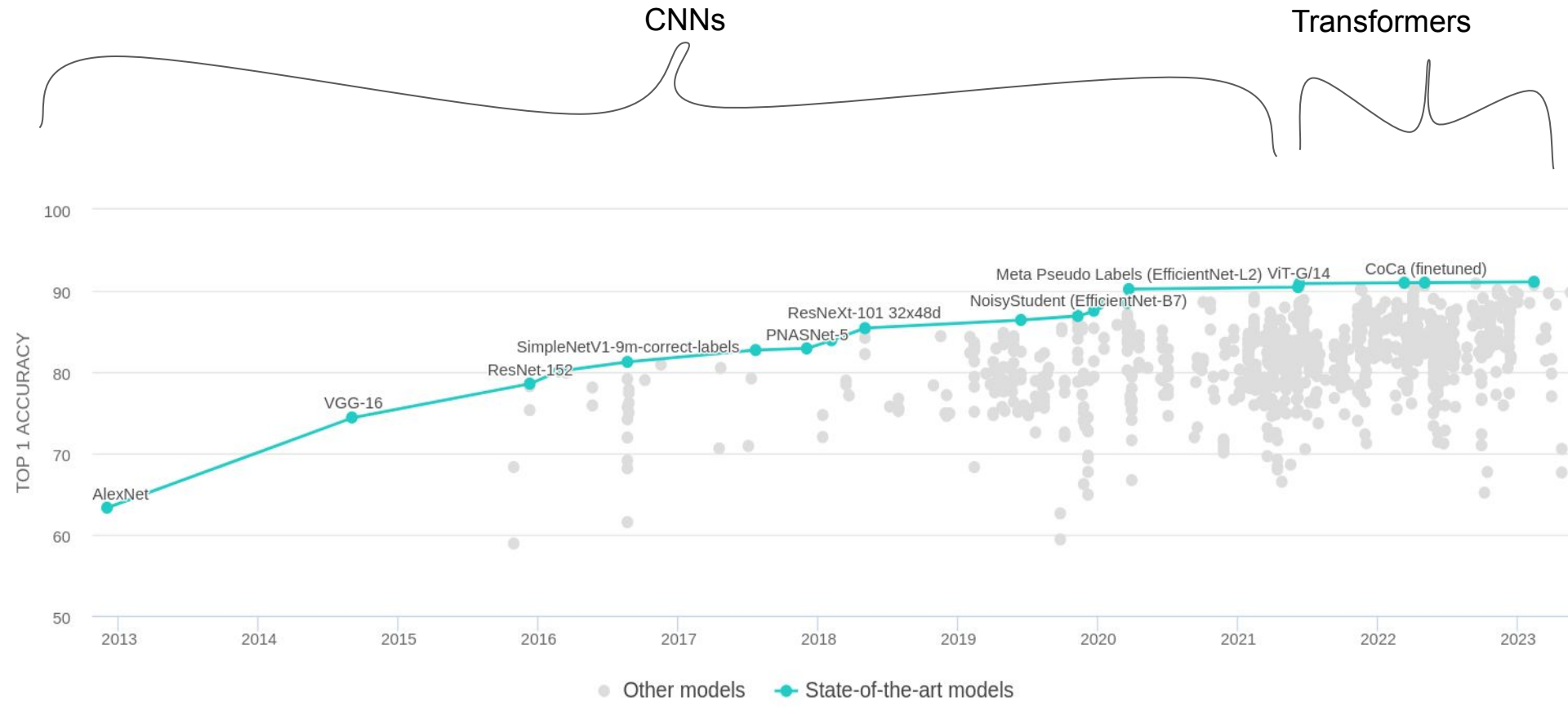
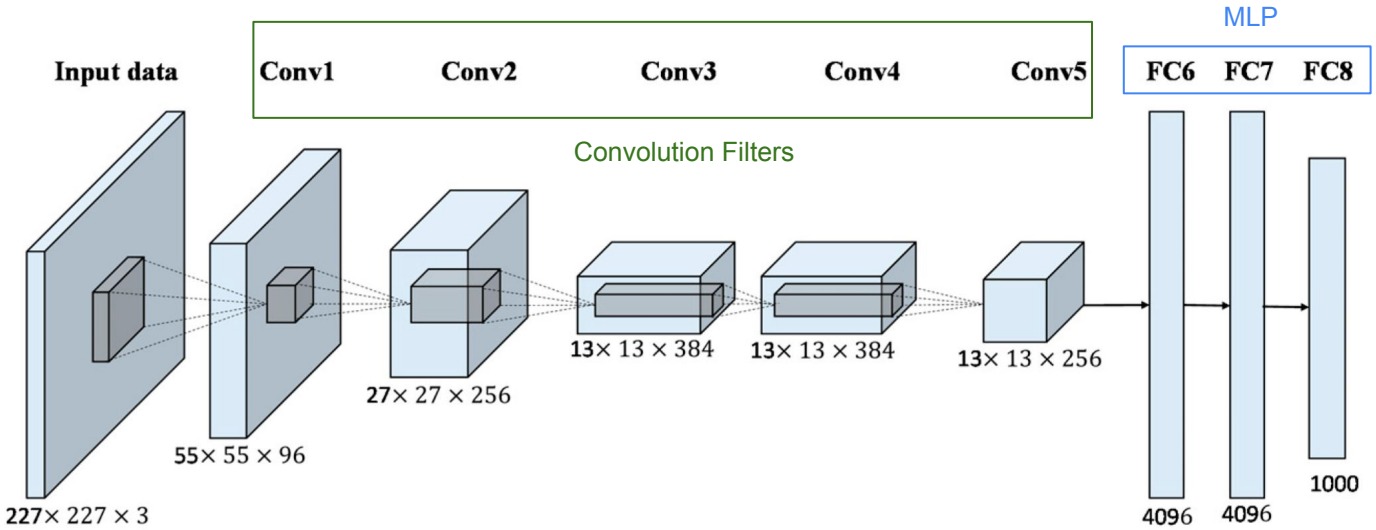


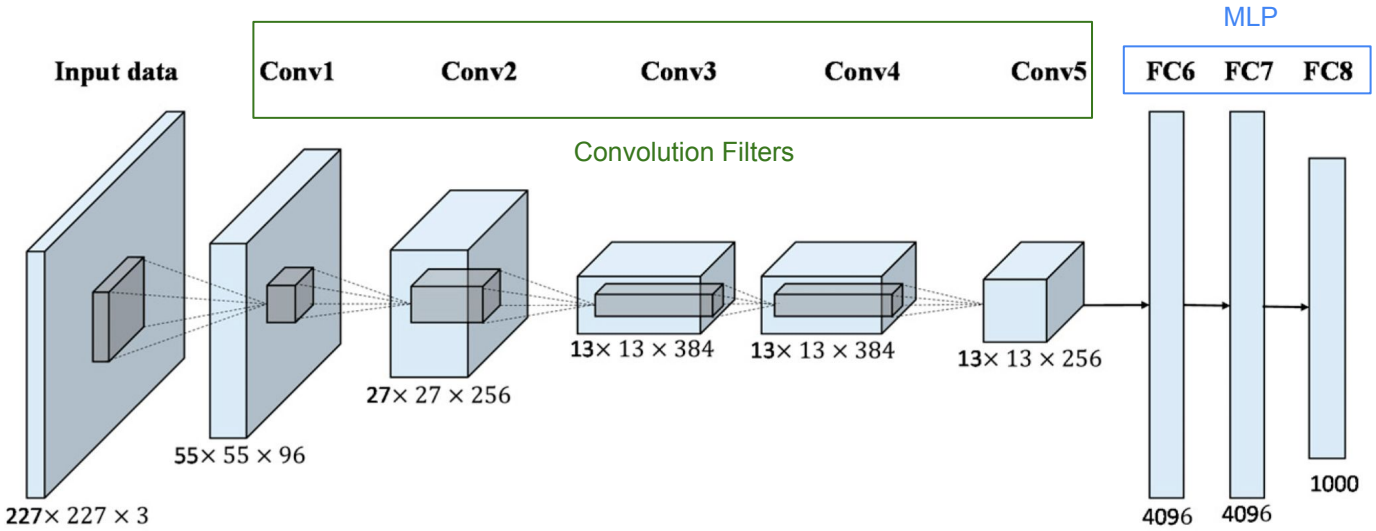
Image Classification SotA

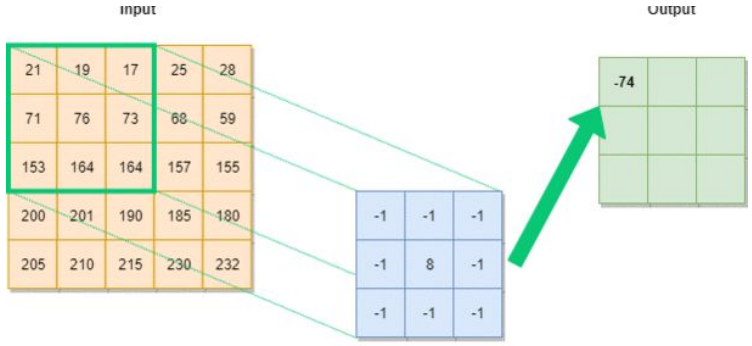


CNN

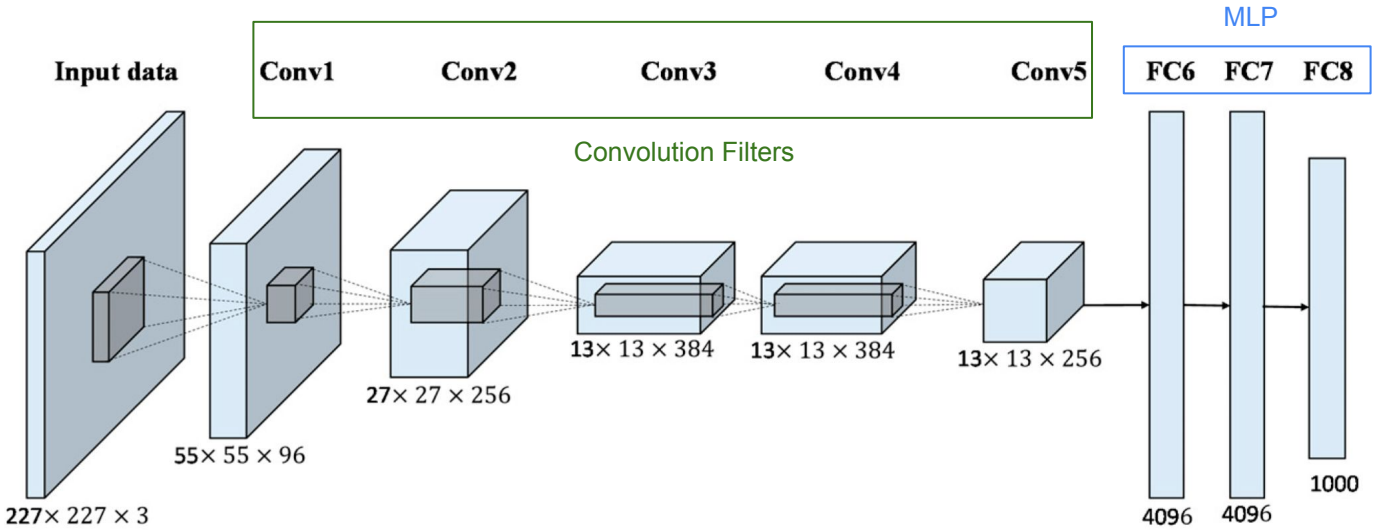

MLP
Input data
Conv1
Conv2
Conv3
Conv4
Conv5
FC6
FC7
FC8
Convolution Filters
227× 227 × 3
55× 55 × 96
27× 27 × 256
13× 13 × 384
13× 13 × 384
13× 13 × 256
4096
4096
1000

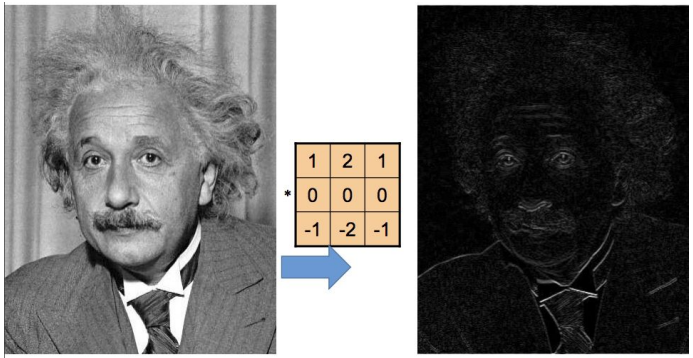
CNN

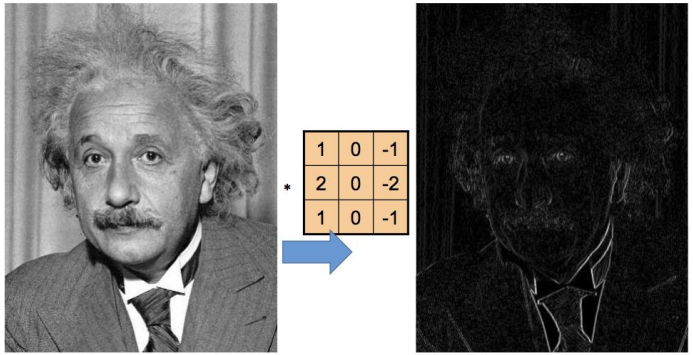

MLP
Input data
Conv1
Conv2
Conv3
Conv4
Conv5
FC6
FC7
FC8
Convolution Filters
227× 227 × 3
55× 55 × 96
27× 27 × 256
13× 13 × 384
13× 13 × 384
13× 13 × 256
4096
4096
1000


Input
Output
21 19 17 25 28
71 76 73 68 59
153 164 164 157 155
200 201 190 185 180
205 210 215 230 232
-1 -1 -1
-1 8 -1
-1 -1 -1
-74

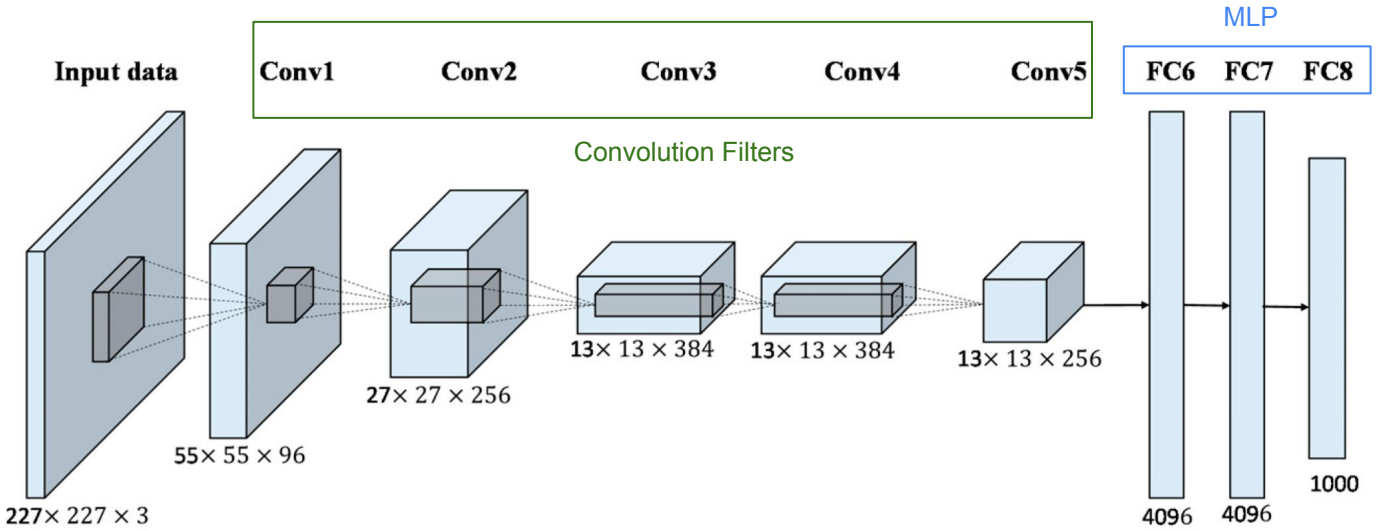
CNN

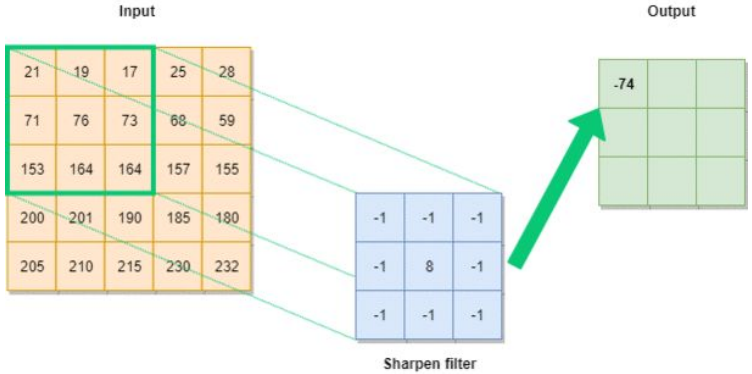

MLP
Input data
Conv1
Conv2
Conv3
Conv4
Conv5
FC6
FC7
FC8
Convolution Filters
227× 227 × 3
55× 55 × 96
27× 27 × 256
13× 13 × 384
13× 13 × 384
13× 13 × 256
4096
4096
1000

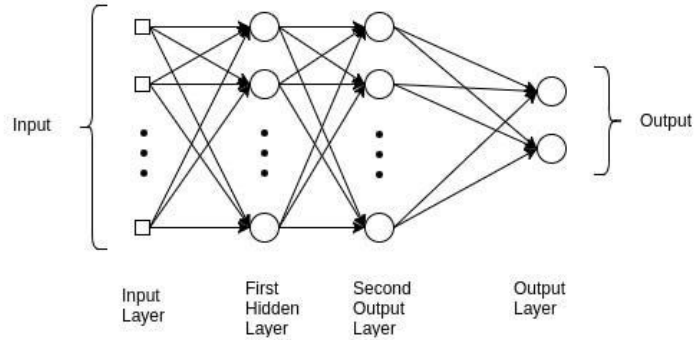

1 2 1
0 0 0
-1 -2 -1
Horizontal Edge
(absolute value)


1 0 -1
2 0 -2
1 0 -1
Vertical Edge
(absolute value)

CNN


MLP
Input data
Conv1
Conv2
Conv3
Conv4
Conv5
FC6
FC7
FC8
Convolution Filters
227× 227 × 3
55× 55 × 96
27× 27 × 256
13× 13 × 384
13× 13 × 384
13× 13 × 256
4096
4096
1000


Input
Output
21 19 17 25 28
71 76 73 68 59
153 164 164 157 155
200 201 190 185 180
205 210 215 230 232
-74
-1 -1 -1
-1 8 -1
-1 -1 -1
Sharpen filter

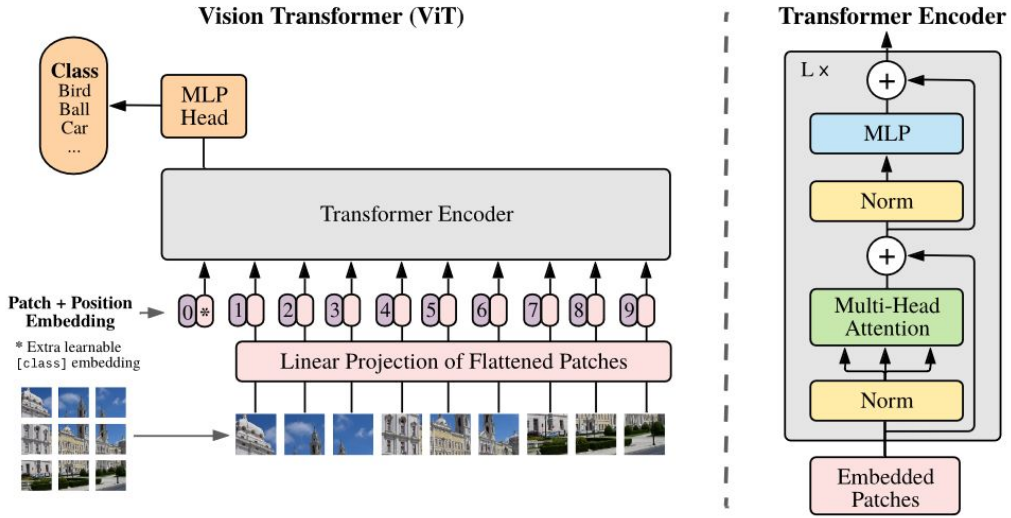

Input
Output
Input Layer
First Hidden Layer
Second Output Layer
Output Layer

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

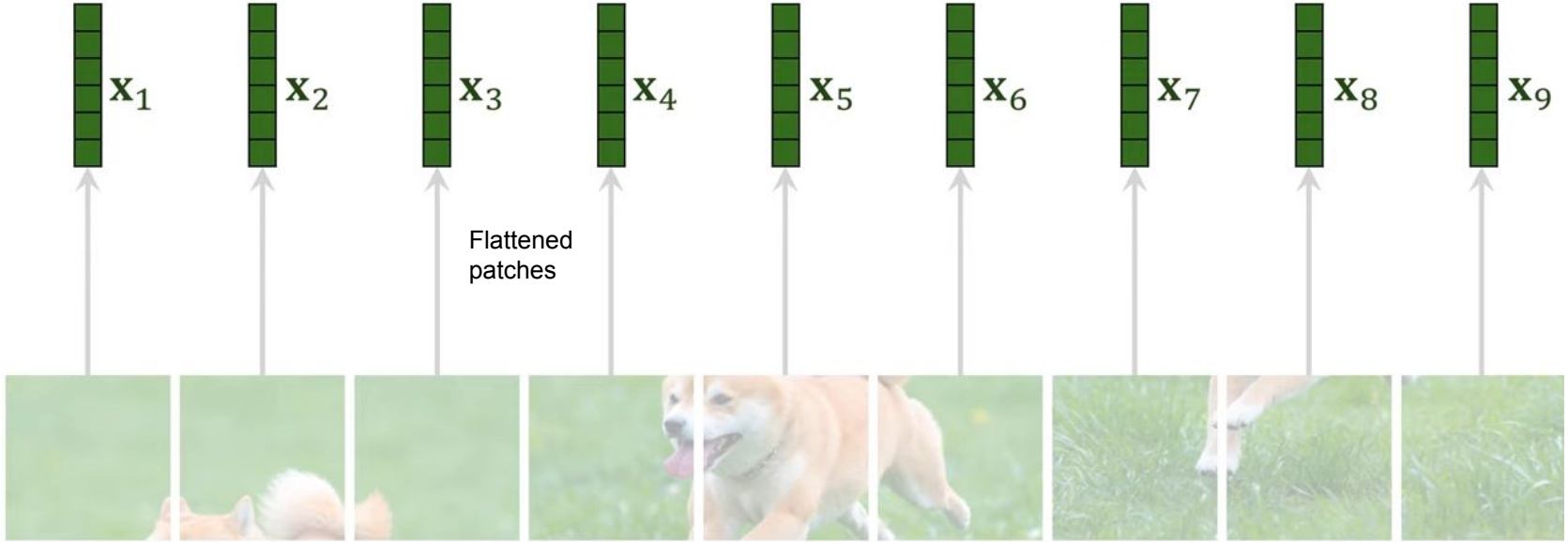
Alexey Dosovitskiy[*,†], **Lucas Beyer**[*], **Alexander Kolesnikov**[*], **Dirk Weissenborn**[*], **Xiaohua Zhai**[*], **Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, Neil Houlsby**[*,†]

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

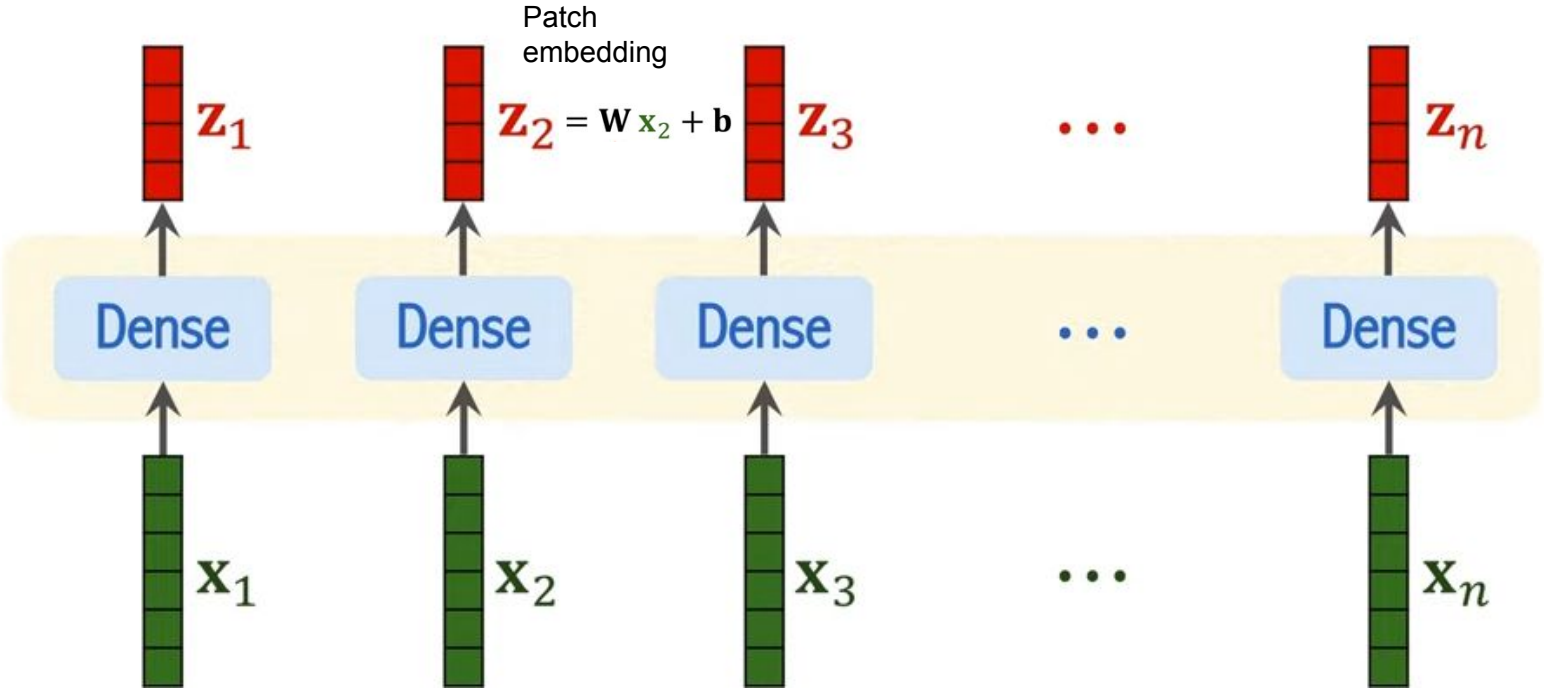
Google Research, Brain Team



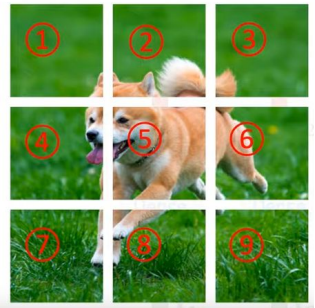
Embedding

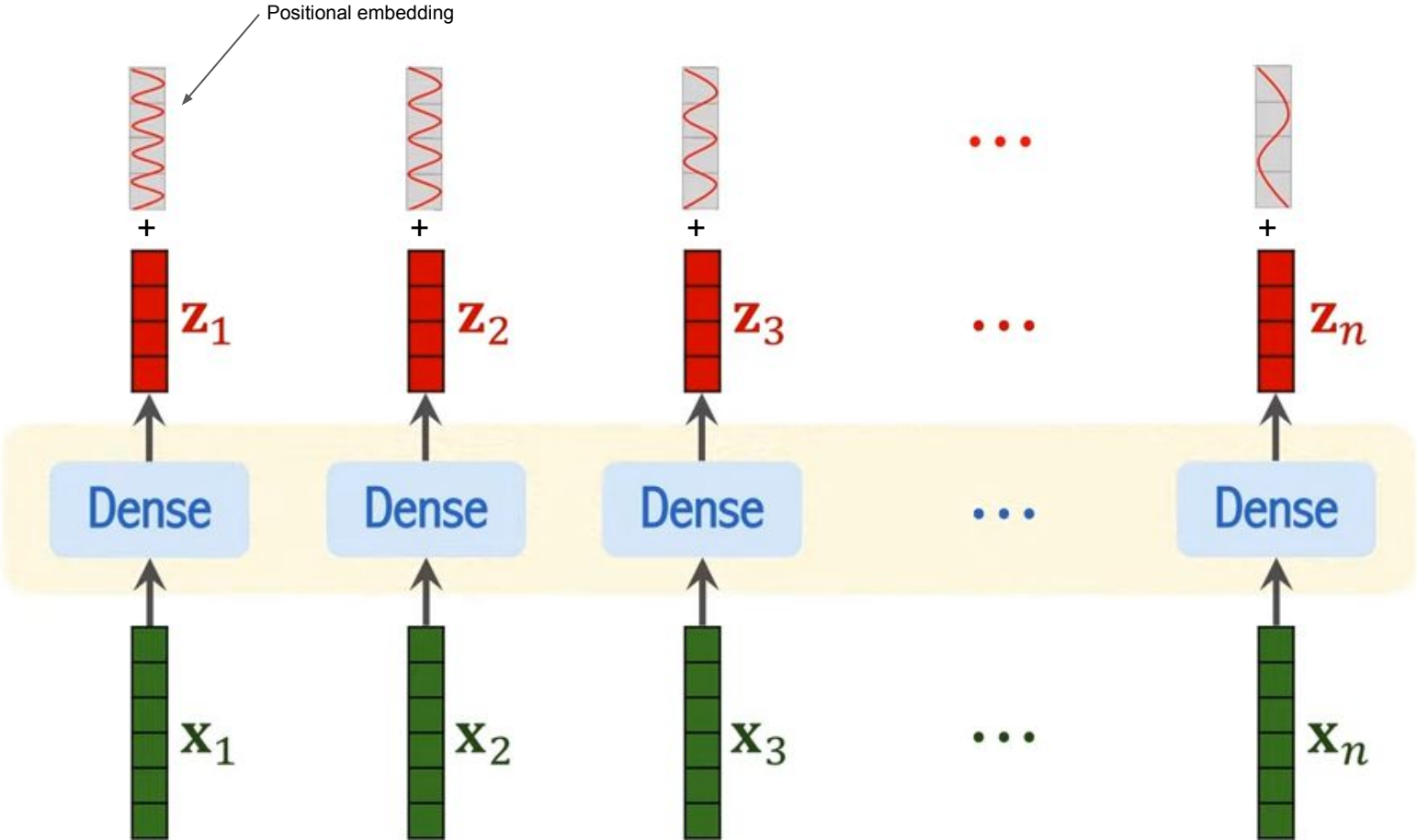

x_1
x_2
x_3
x_4
x_5
x_6
x_7
x_8
x_9
Flattened patches
Split image into patches

Embedding

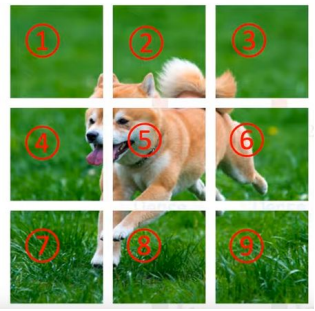

Patch
embedding
$\mathbf{z}_1$
$\mathbf{z}_2 = \mathbf{W}\,\mathbf{x}_2 + \mathbf{b}$
$\mathbf{z}_3$
$\mathbf{z}_n$
Dense
Dense
Dense
Dense
$\mathbf{x}_1$
$\mathbf{x}_2$
$\mathbf{x}_3$
$\mathbf{x}_n$

Embedding


1
2
3
4
5
6
7
8
9


Positional embedding
+
+
+
+
z_1
z_2
z_3
z_n
Dense
Dense
Dense
Dense
x_1
x_2
x_3
x_n

Embedding

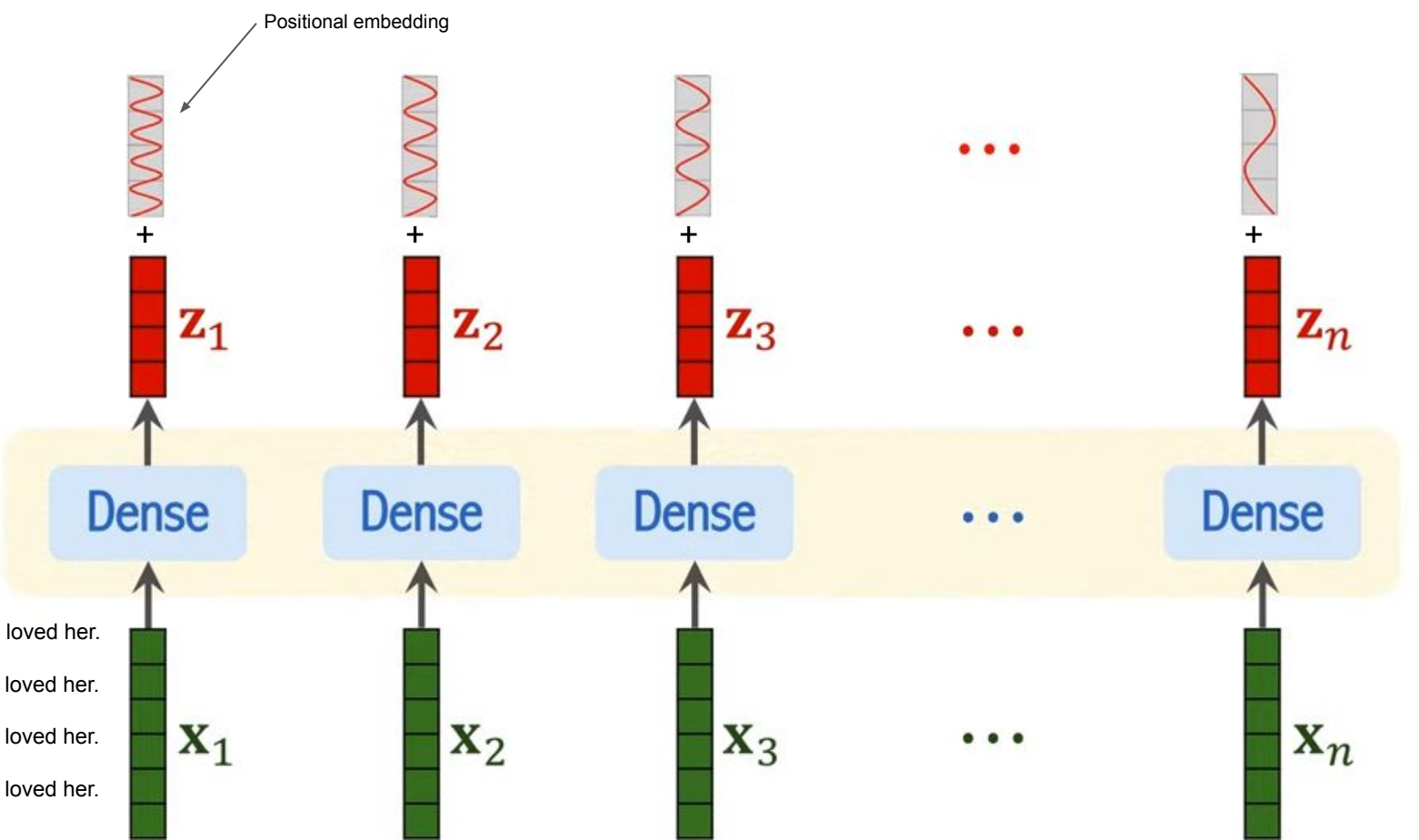


Only he told his mistress that he loved her.

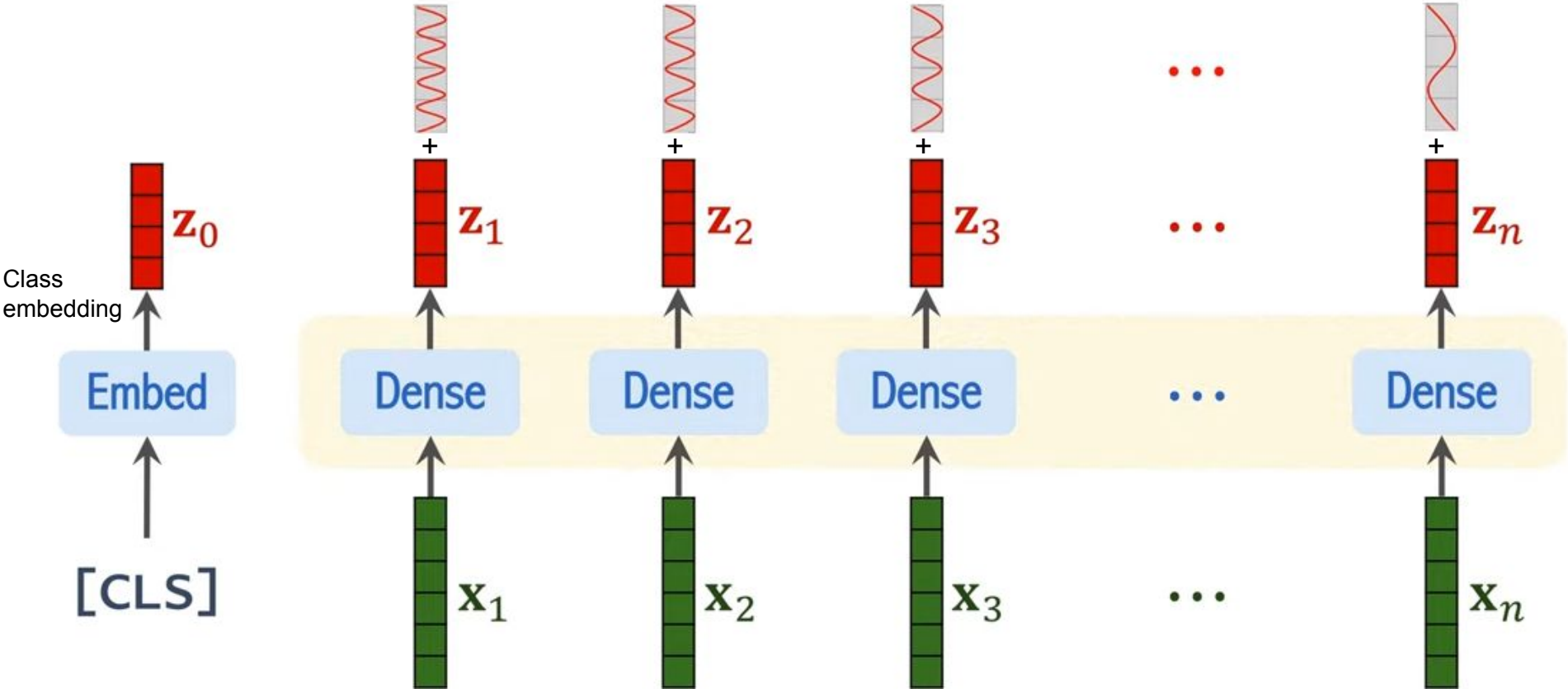
He told only his mistress that he loved her.

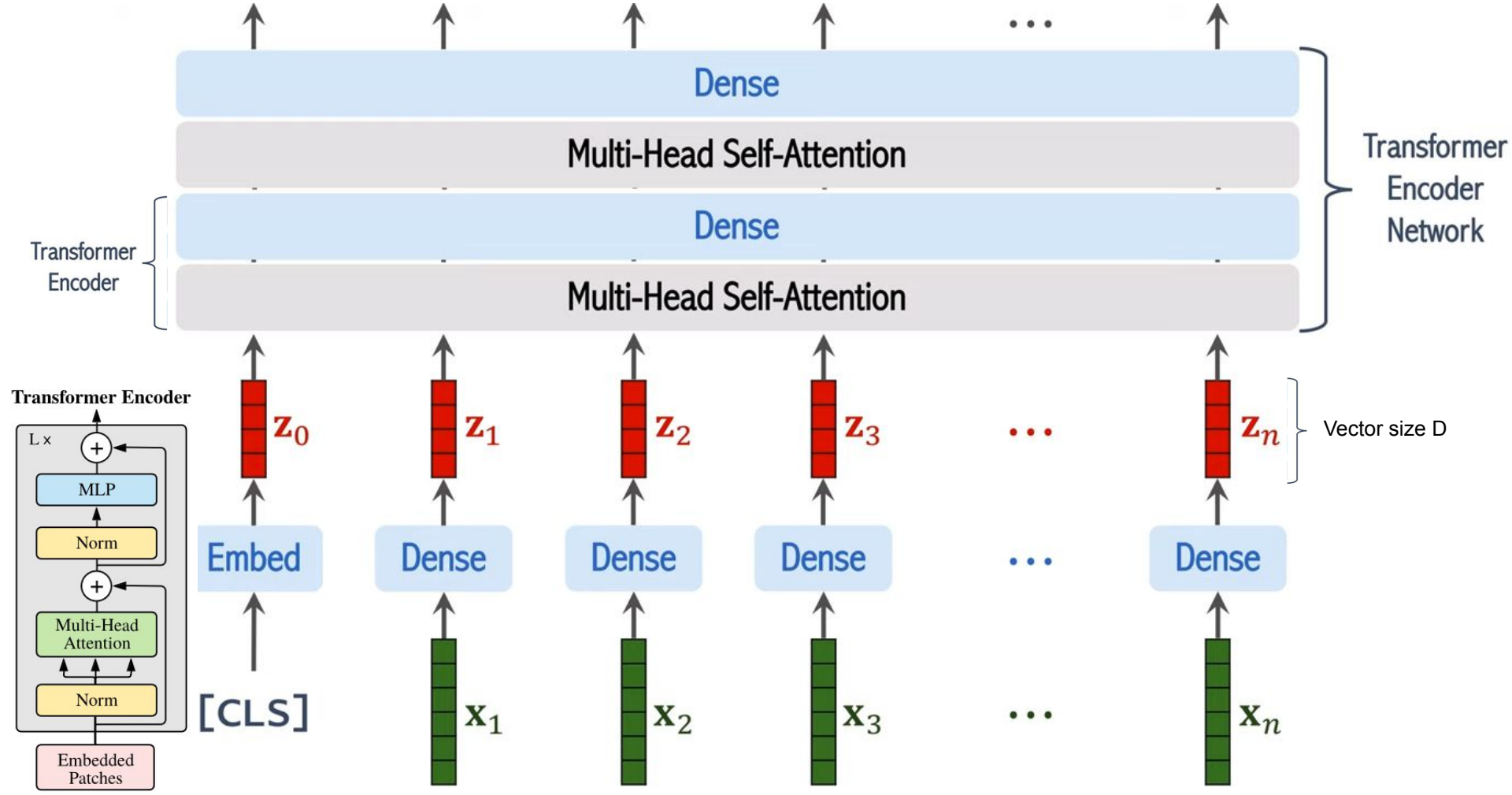
He told his only mistress that he loved her.

He told his mistress that only he loved her.



Embedding


Class embedding
z_0
z_1
z_2
z_3
z_n
Embed
Dense
Dense
Dense
Dense
[CLS]
x_1
x_2
x_3
x_n


Dense
Multi-Head Self-Attention
Dense
Multi-Head Self-Attention
Transformer Encoder
Transformer Encoder Network
Transformer Encoder
L ×
MLP
Norm
Multi-Head Attention
Norm
Embedded Patches
z_0 z_1 z_2 z_3 ... z_n
Vector size D
Embed
Dense
Dense
Dense
...
Dense
[CLS]
x_1 x_2 x_3 ... x_n

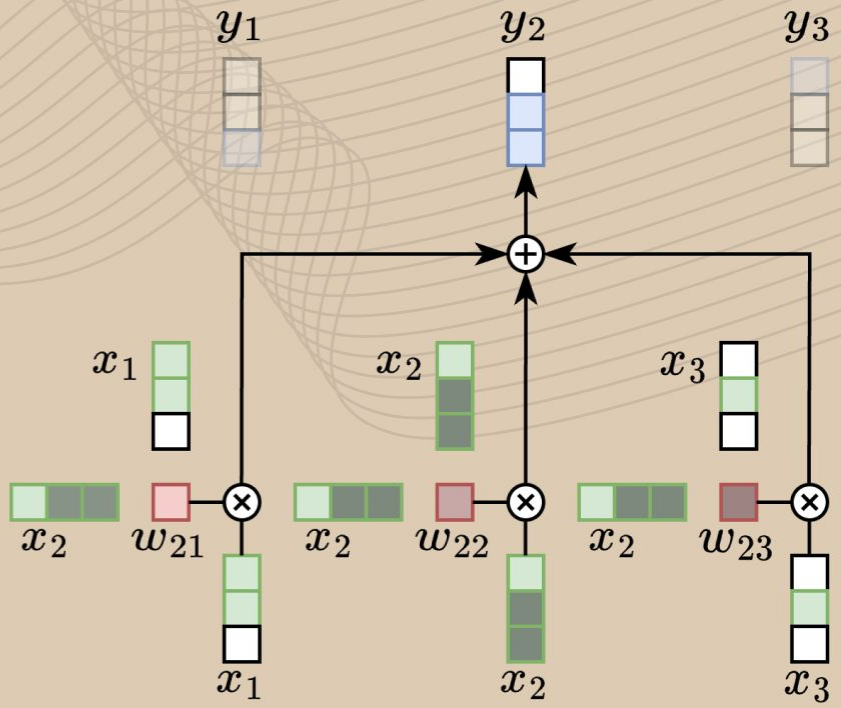
¿Qué es el Self-Attention?

- Es la operación fundamental en cualquier arquitectura de los Transformer
- Es una operación entre secuencias de vectores
- Es la única operación que propaga información entre vectores

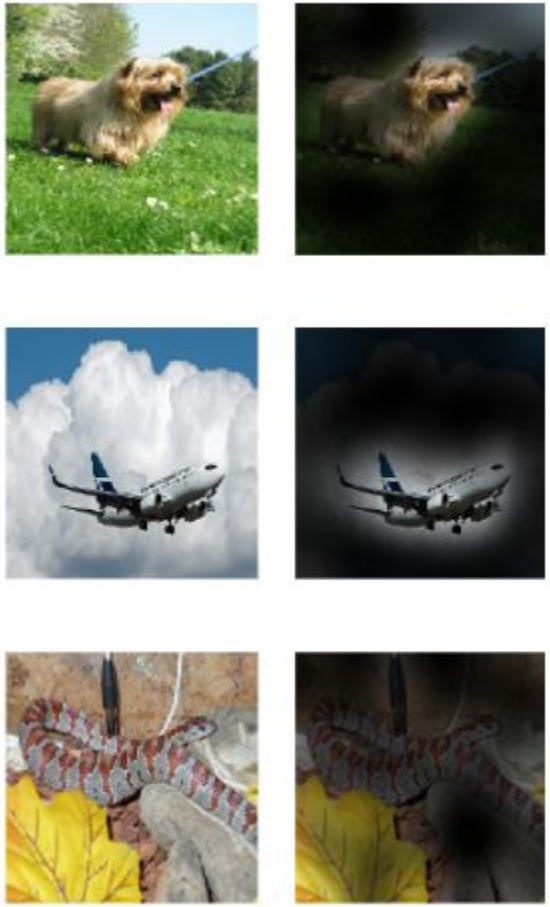
$$y_i = \sum_j w_{ij} x_j$$

$$w'_{ij} = x_i^T x_j$$

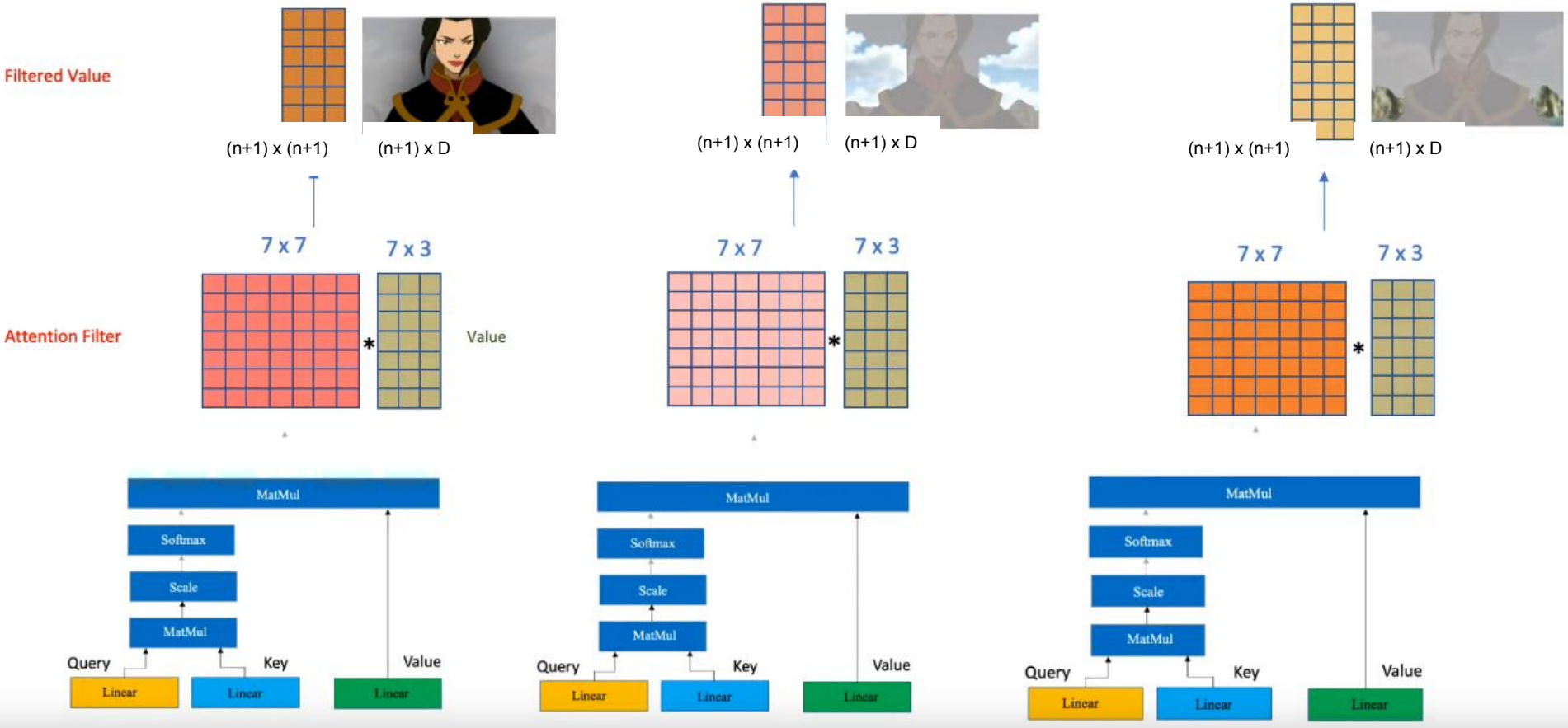
$$w_{ij} = \frac{e^{w'_{ij}}}{\sum_j e^{w'_{ij}}} \quad (Softmax)$$



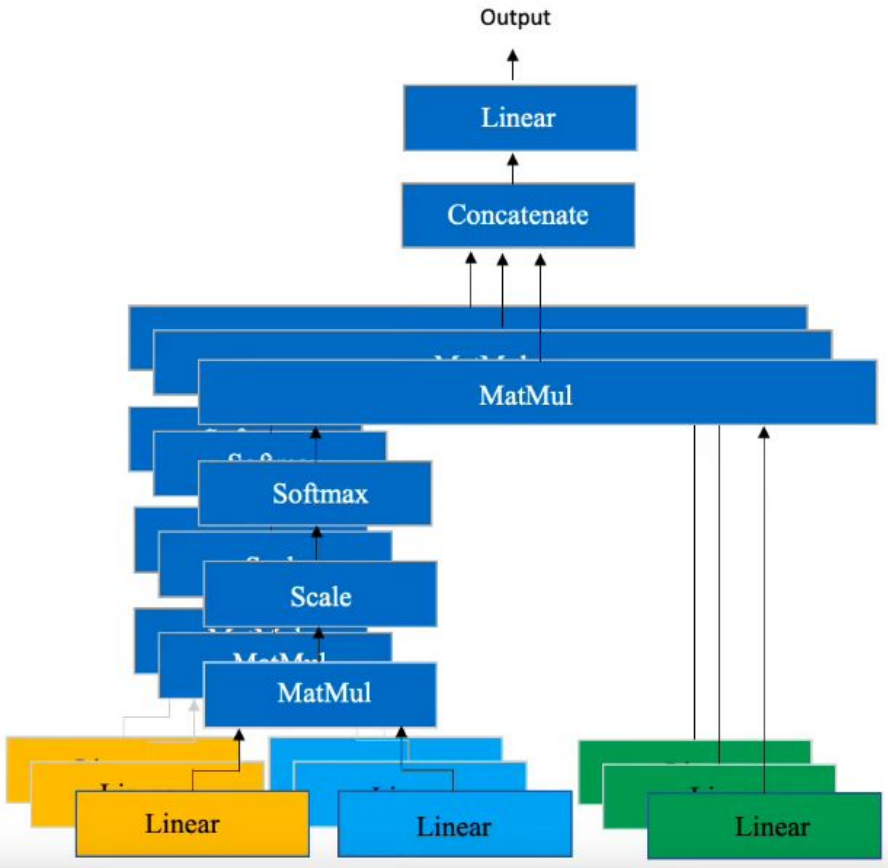
Input Attention

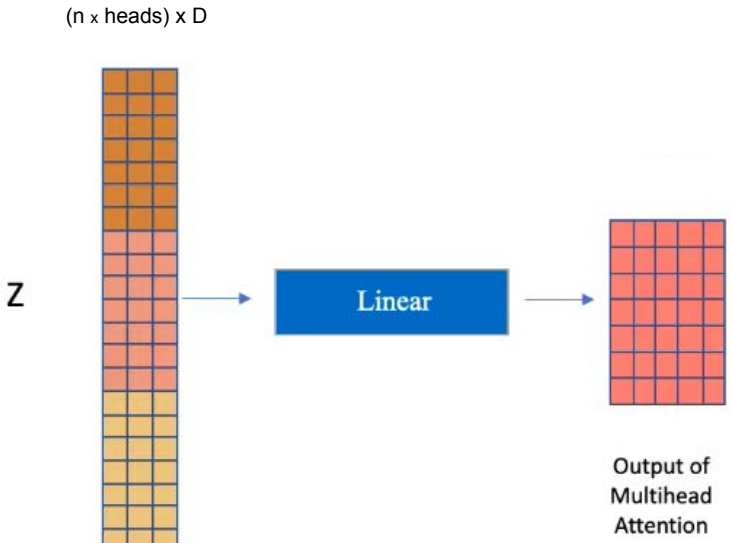


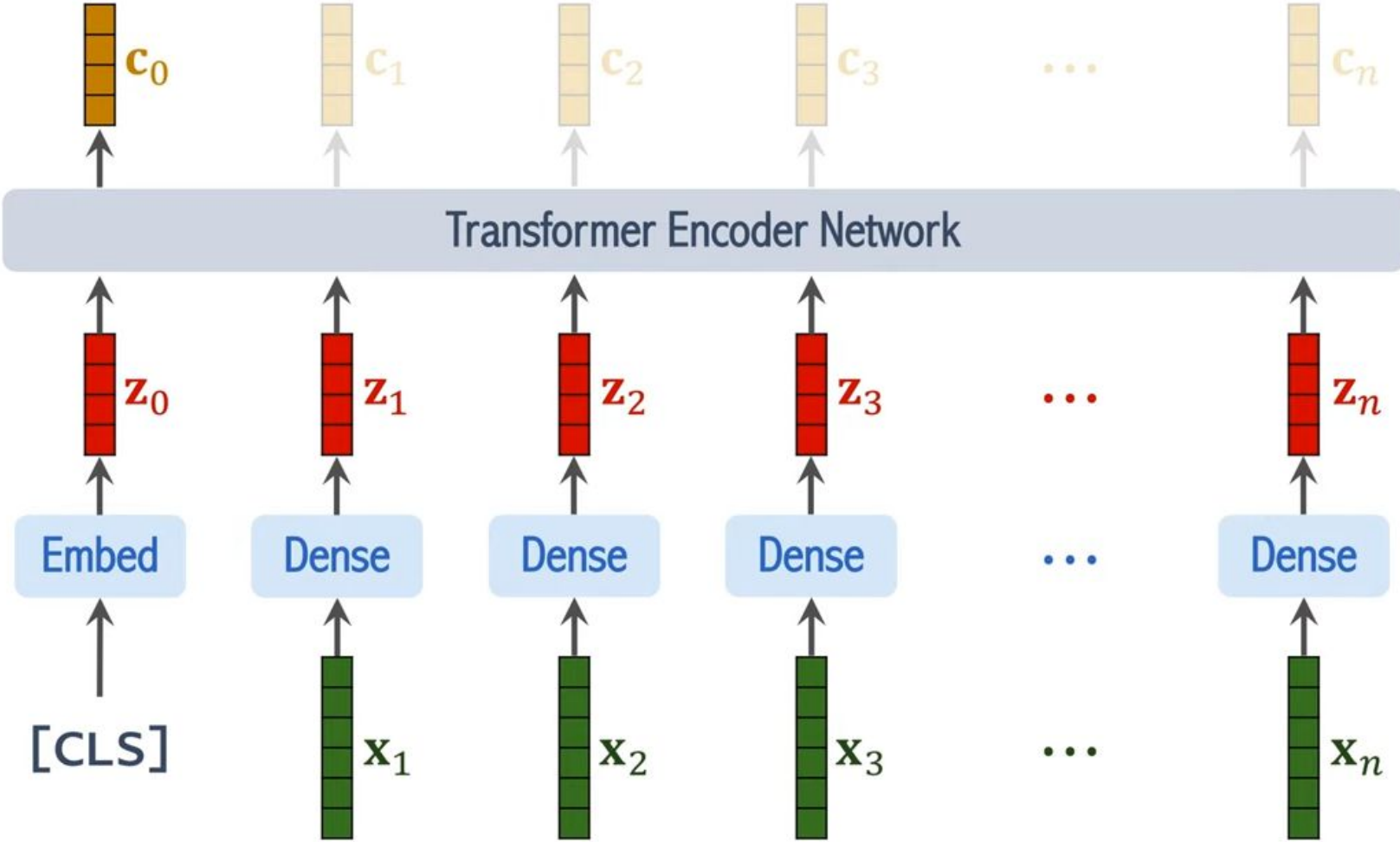
Multi-head self-attention

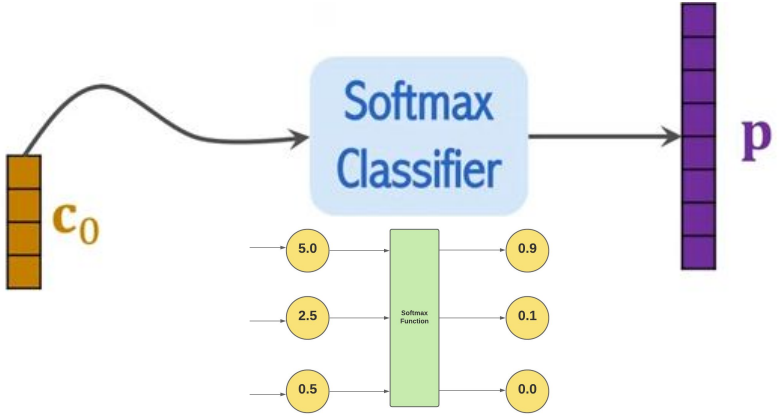


Multi-head self-attention

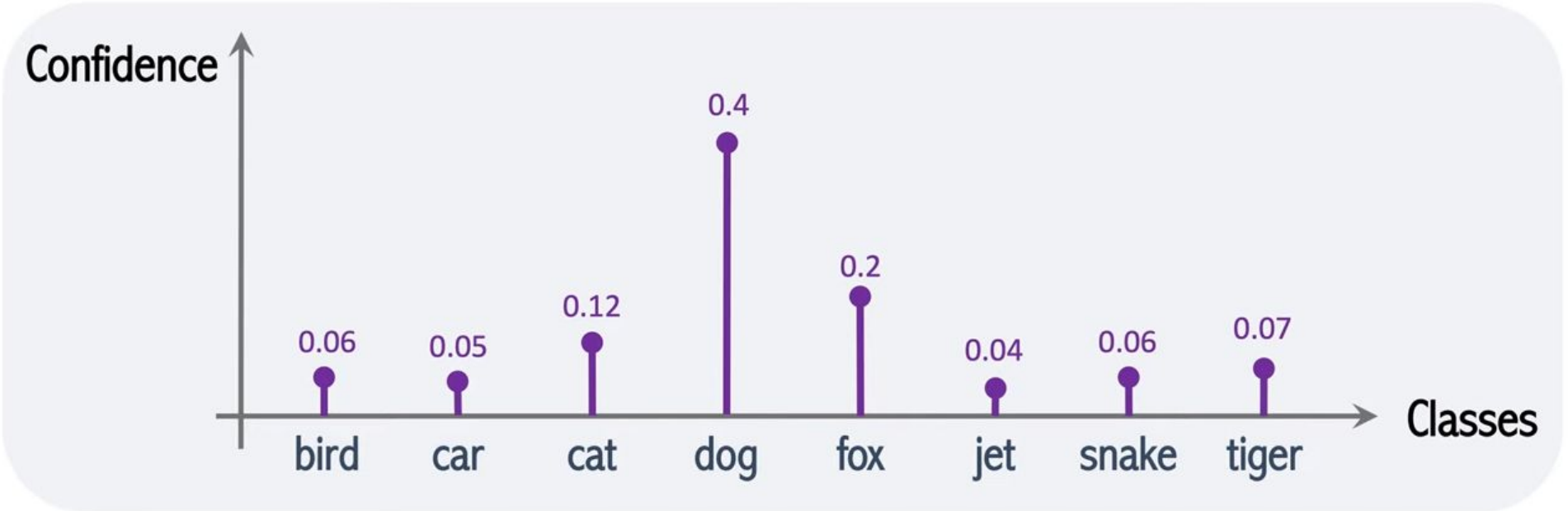


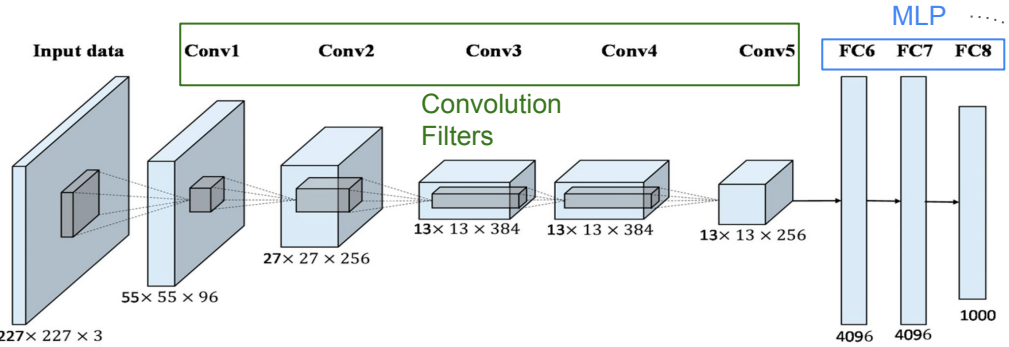


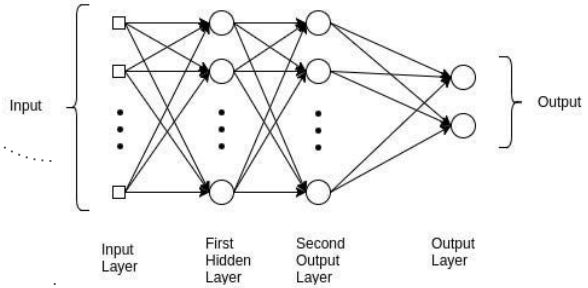

c_0
c_1
c_2
c_3
c_n
Transformer Encoder Network
z_0
z_1
z_2
z_3
z_n
Embed
Dense
Dense
Dense
Dense
[CLS]
x_1
x_2
x_3
x_n

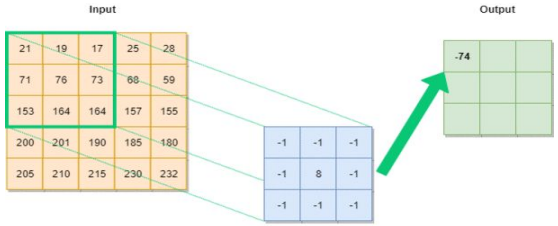


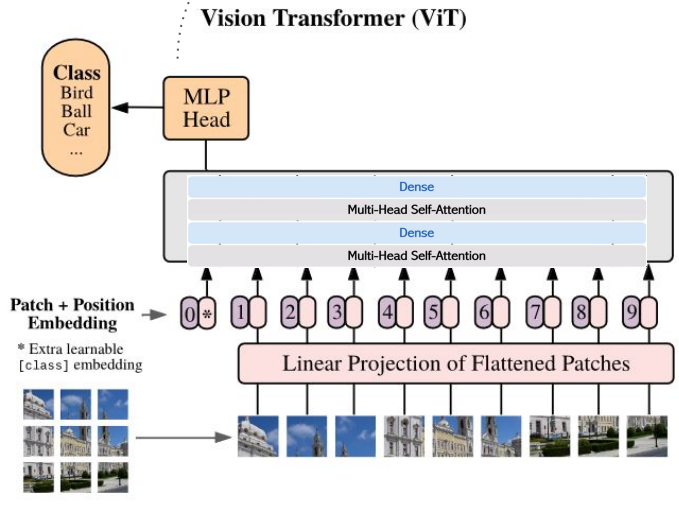
$$\text{Loss} = -\sum_{i=1}^{\text{output size}} y_i \cdot \log \hat{y}_i$$

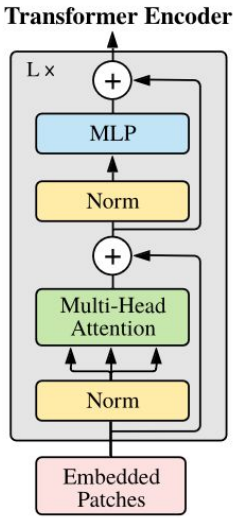


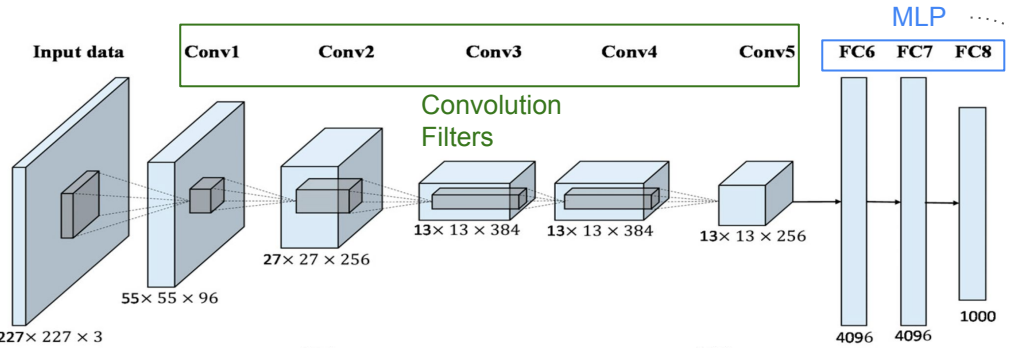

Input data
Conv1
Conv2
Conv3
Conv4
Conv5
FC6
FC7
FC8
MLP
Convolution Filters
227× 227 × 3
55× 55 × 96
27× 27 × 256
13× 13 × 384
13× 13 × 384
13× 13 × 256
4096
4096
1000

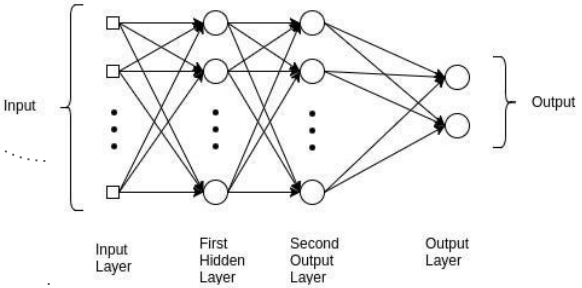

Input
Output
Input Layer
First Hidden Layer
Second Output Layer
Output Layer

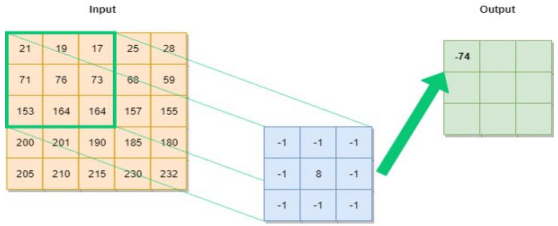

Input
Output
21 19 17 25 28
71 76 73 68 59
153 164 164 157 155
200 201 190 185 180
205 210 215 230 232
-1 -1 -1
-1 8 -1
-1 -1 -1
-74

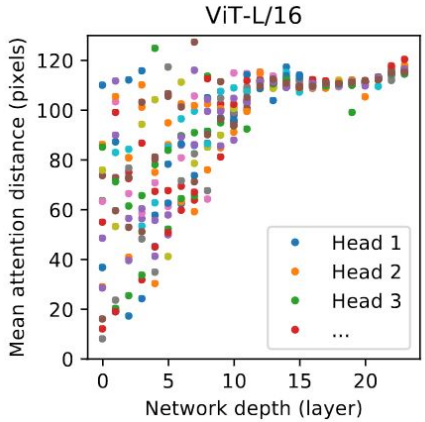

Vision Transformer (ViT)
Class
Bird
Ball
Car
...
MLP Head
Dense
Multi-Head Self-Attention
Dense
Multi-Head Self-Attention
Patch + Position Embedding
* Extra learnable [class] embedding
0 * 1 2 3 4 5 6 7 8 9
Linear Projection of Flattened Patches

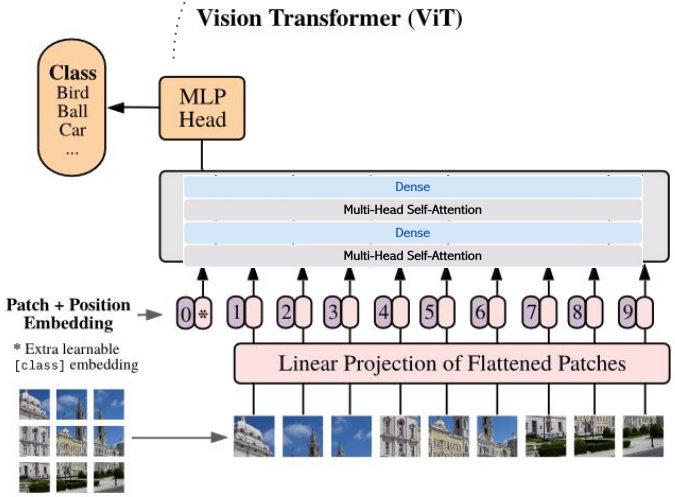

Transformer Encoder
L x
MLP
Norm
Multi-Head Attention
Norm
Embedded Patches

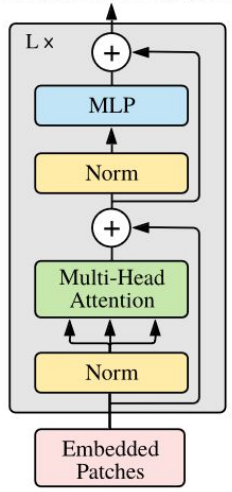

Input data
Conv1
Conv2
Conv3
Conv4
Conv5
FC6
FC7
FC8
MLP
Convolution Filters
227× 227 × 3
55× 55 × 96
27× 27 × 256
13× 13 × 384
13× 13 × 384
13× 13 × 256
4096
4096
1000

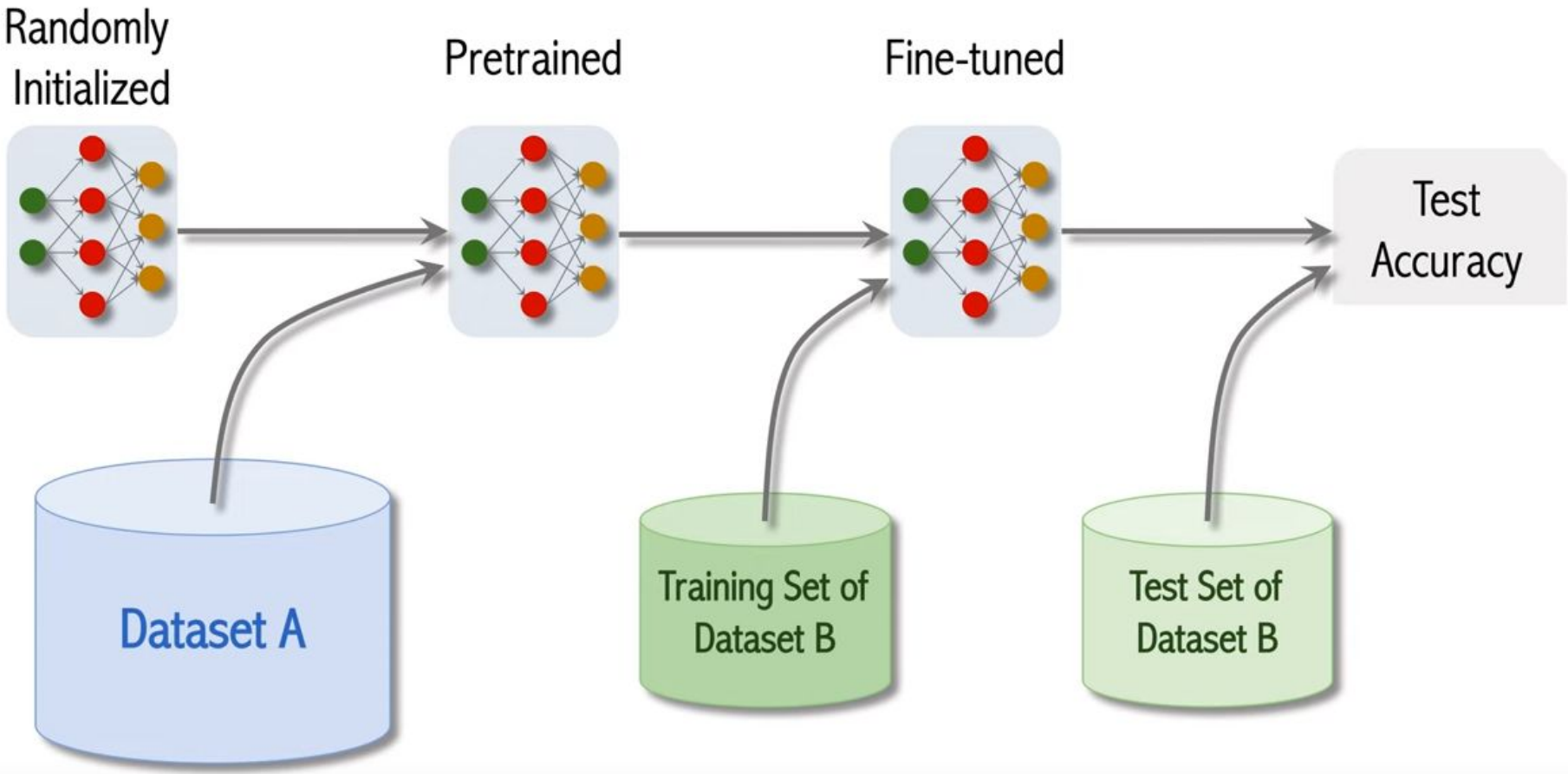

Input
Output
Input Layer
First Hidden Layer
Second Output Layer
Output Layer


Input
Output
21 19 17 25 28
71 76 73 68 59
153 164 164 157 155
200 201 190 185 180
205 210 215 230 232
-1 -1 -1
-1 8 -1
-1 -1 -1
-74


ViT-L/16
Mean attention distance (pixels)
Network depth (layer)
Head 1
Head 2
Head 3
...


Vision Transformer (ViT)
Class
Bird
Ball
Car
...
MLP Head
Dense
Multi-Head Self-Attention
Dense
Multi-Head Self-Attention
Patch + Position Embedding
* Extra learnable [class] embedding
Linear Projection of Flattened Patches


Transformer Encoder
L x
MLP
Norm
Multi-Head Attention
Norm
Embedded Patches


Randomly Initialized
Pretrained
Fine-tuned
Test Accuracy
Dataset A
Training Set of Dataset B
Test Set of Dataset B

	# of Images	# of Classes
ImageNet (Small)	1.3 Million	1 Thousand
ImageNet-21K (Medium)	14 Million	21 Thousand
JFT (Big)	300 Million	18 Thousand

- Pretrain the model on Dataset A, fine-tune the model on Dataset B, and evaluate the model on Dataset B.

- 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.