Transformer Model

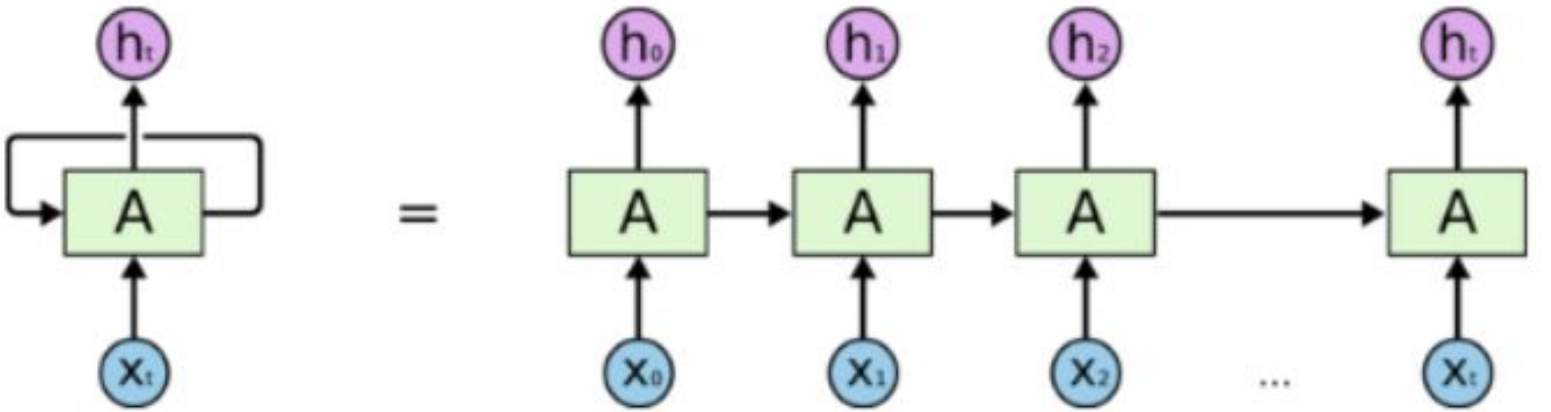
Architecture

Advantages and Disadvantages

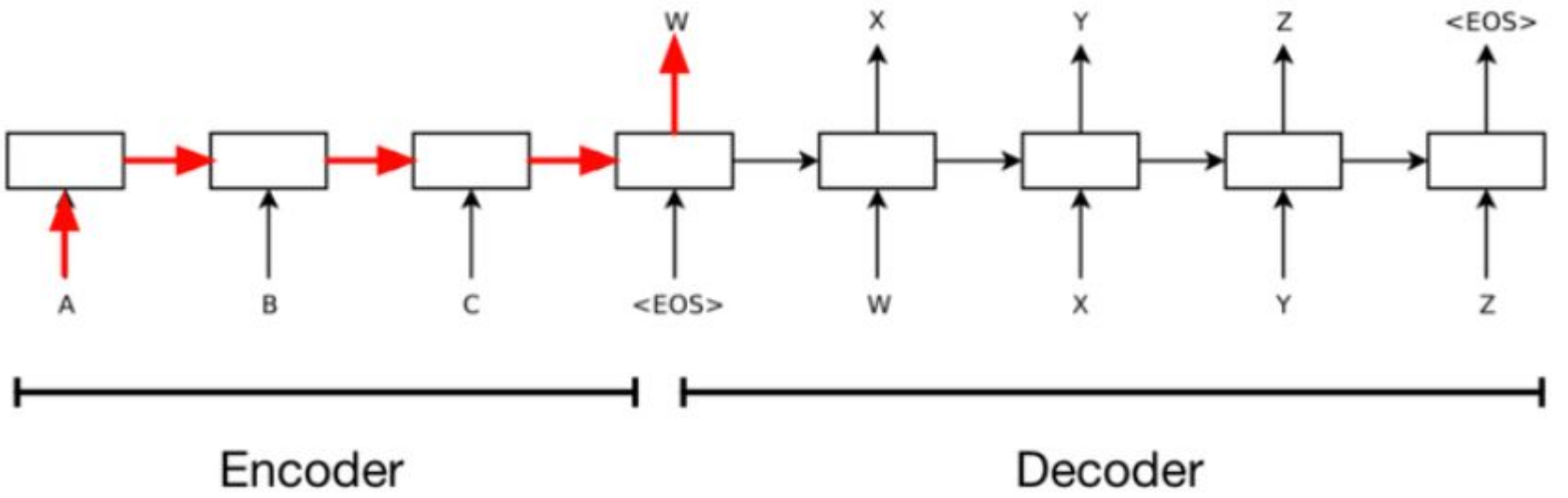
Performance Comparison

RNN's

- Can handle time series data
- Suffers from vanishing / exploding gradient problem
- Solution? LSTM

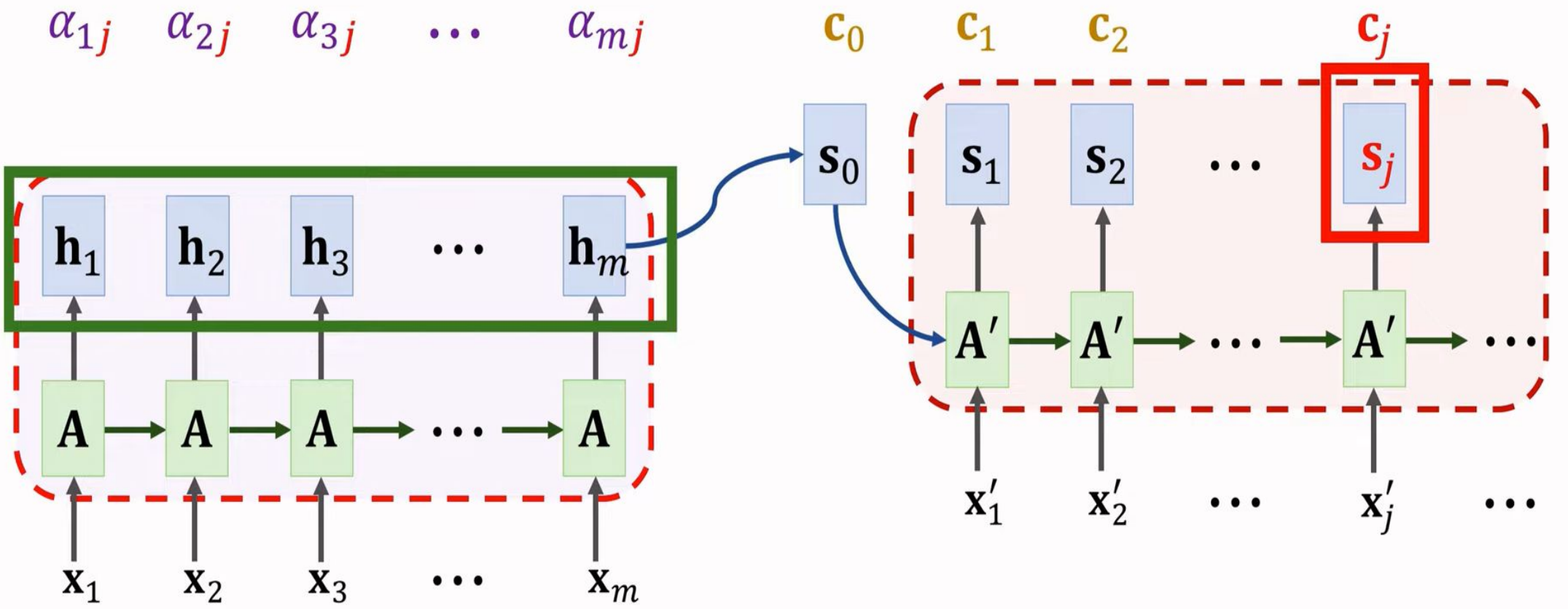


Encoder & Decoder



Attention for Seq2Seq Model

Weights: $\alpha_{ij} = \text{align}(\mathbf{h}_i, \mathbf{s}_j)$.



Attention for Seq2Seq Model

Query: $\mathbf{q}_{:j} = \mathbf{W}_Q \mathbf{s}_j$, Key: $\mathbf{k}_{:i} = \mathbf{W}_K \mathbf{h}_i$, Value: $\mathbf{v}_{:i} = \mathbf{W}_V \mathbf{h}_i$.

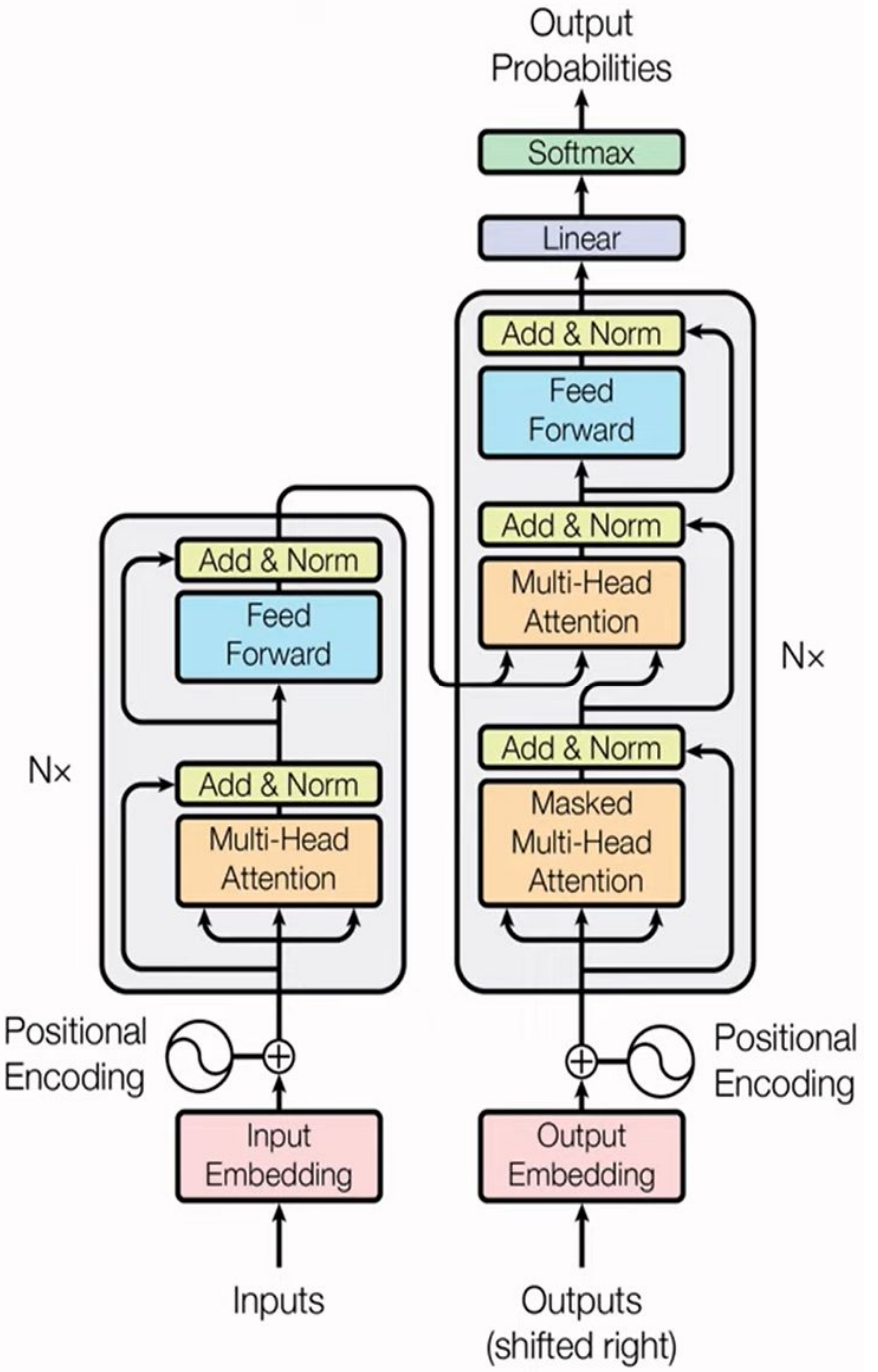
Weights: $\boldsymbol{\alpha}_{:j} = \mathrm{Softmax}\left(\mathbf{K}^T \mathbf{q}_{:j}\right) \in \mathbb{R}^m$.

Context vector: $\mathbf{c}_j = \alpha_{1j}\mathbf{v}_{:1} + \cdots + \alpha_{mj}\mathbf{v}_{:m}$.

- Query: $\mathbf{q}_{:j} = \mathbf{W}_Q \mathbf{s}_j$. (To match others.)
- Key: $\mathbf{k}_{:i} = \mathbf{W}_K \mathbf{h}_i$. (To be matched.)
- Value: $\mathbf{v}_{:i} = \mathbf{W}_V \mathbf{h}_i$. (To be weighted averaged.)

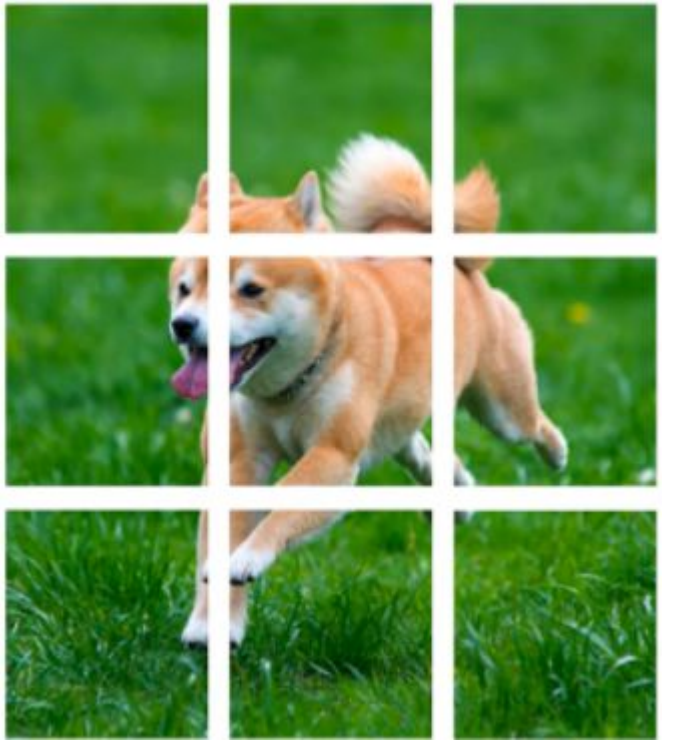
Transformer Model

- Transformer is a Seq2Seq model.
- Transformer is not RNN.
- Purely based on attention and dense layers.
- Higher accuracy than RNNs on large datasets.



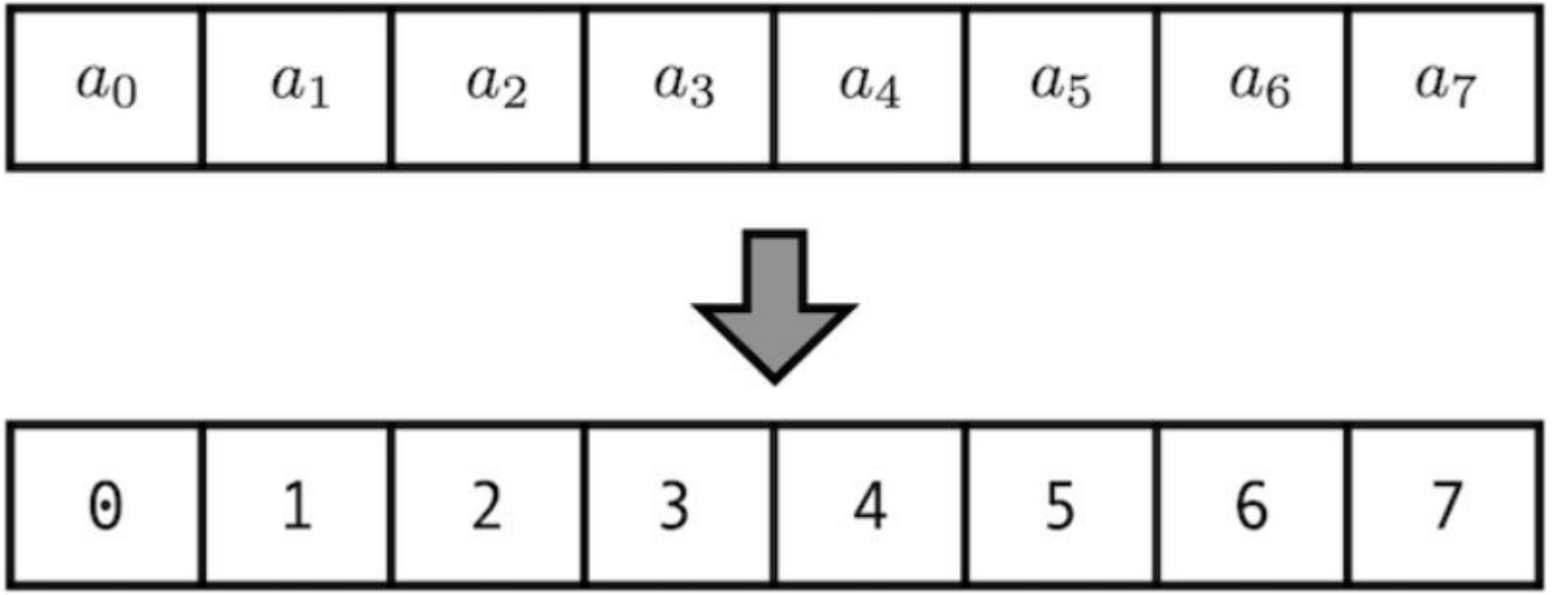
Split Image into Patches

- May also be overlapped
- Number of patches can be calculated:
- N= H*W/ P*P (P = Patch size)



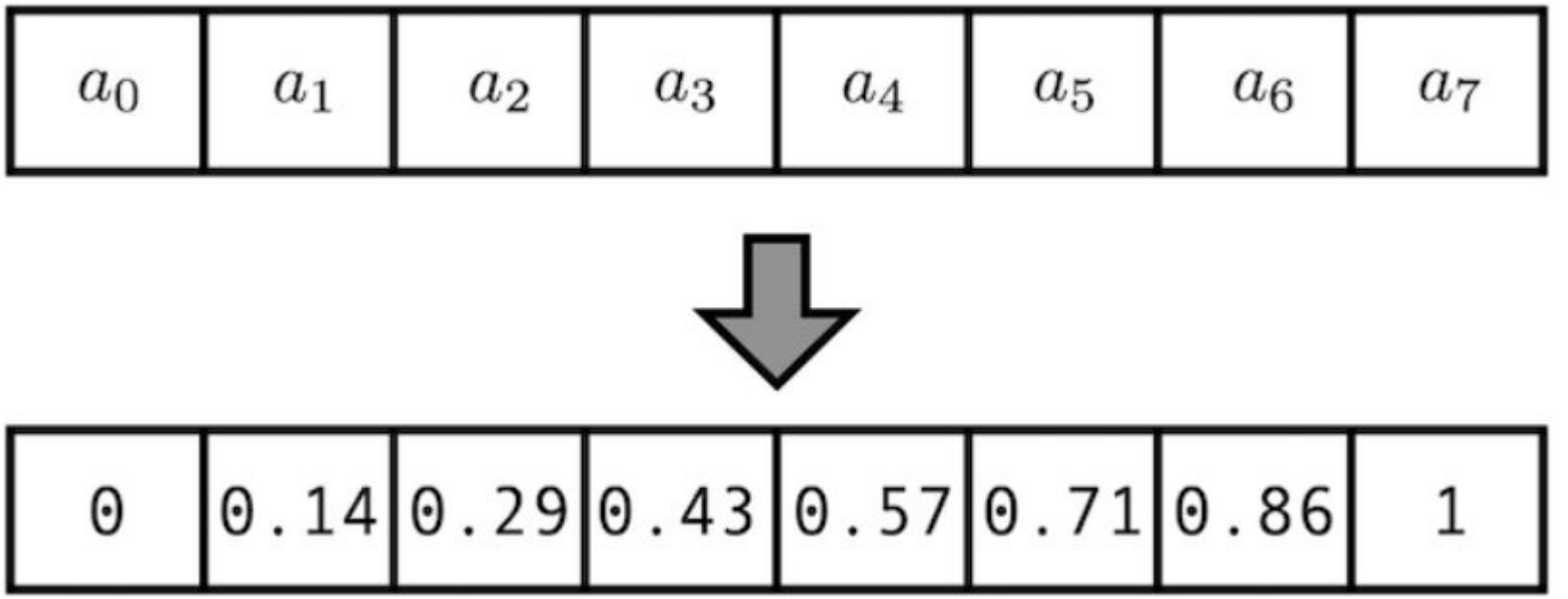
Positional encoding

- Just count



- Consider 500 length sequence ?
- want encoding to be a number in the range [-1,1]

- Normalize the above encoding



- Consider variable sequence length ?
- want encoding to be consistent throughout

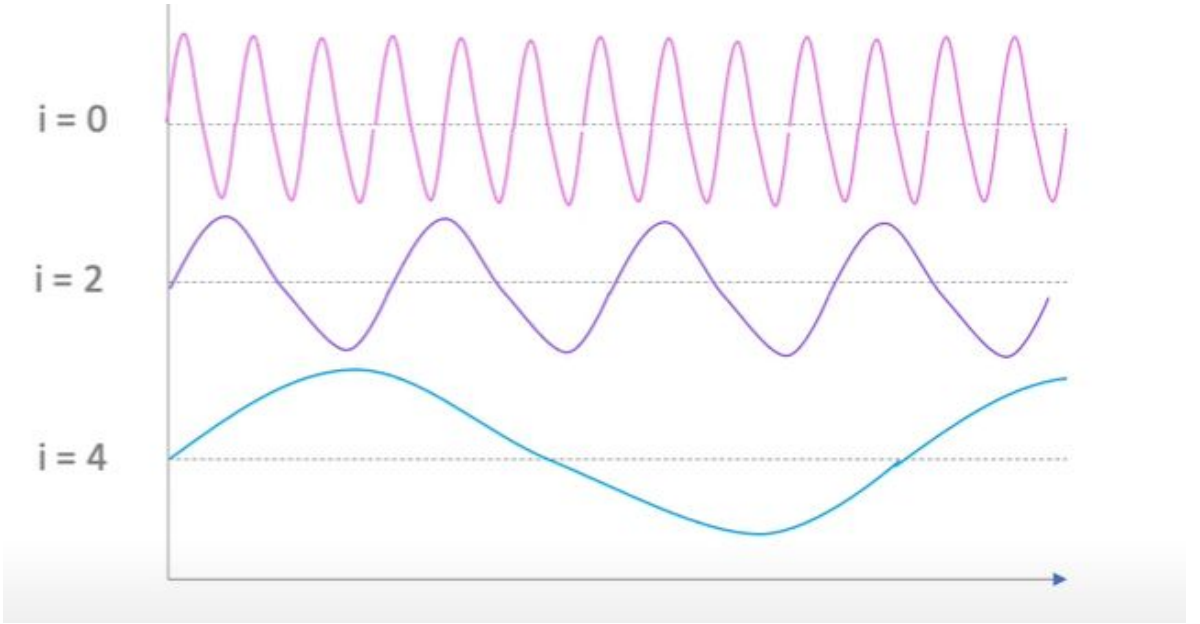
Using frequencies

- Used sine and cosine curves
- Range => [-1,1] and same output on same input
- d = embedding dimension, i = entries of positional encoding vector

$$\vec{p}_t^{\,(i)} = f(t)^{(i)} := \begin{cases} \sin(\omega_k.t), & \text{if } i = 2k \\ \cos(\omega_k.t), & \text{if } i = 2k+1 \end{cases}$$

where

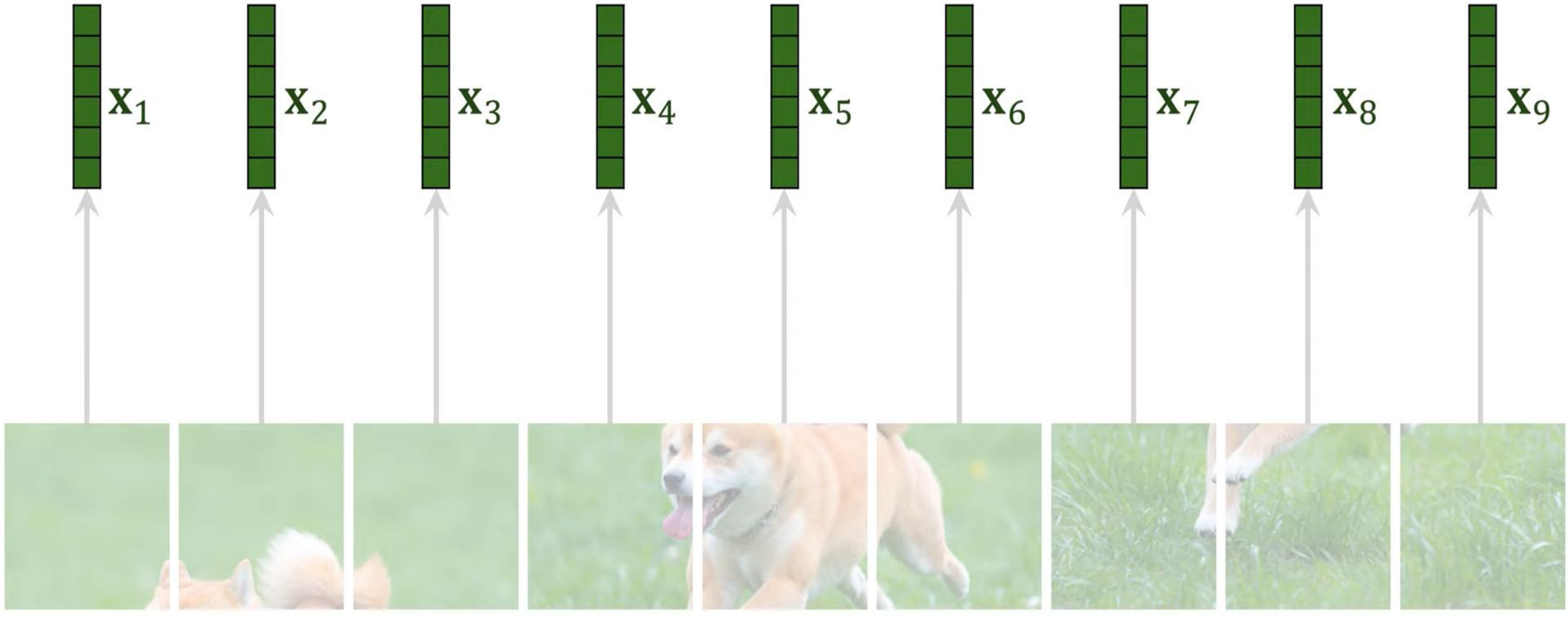
$$\omega_k = \frac{1}{10000^{2k/d}}$$

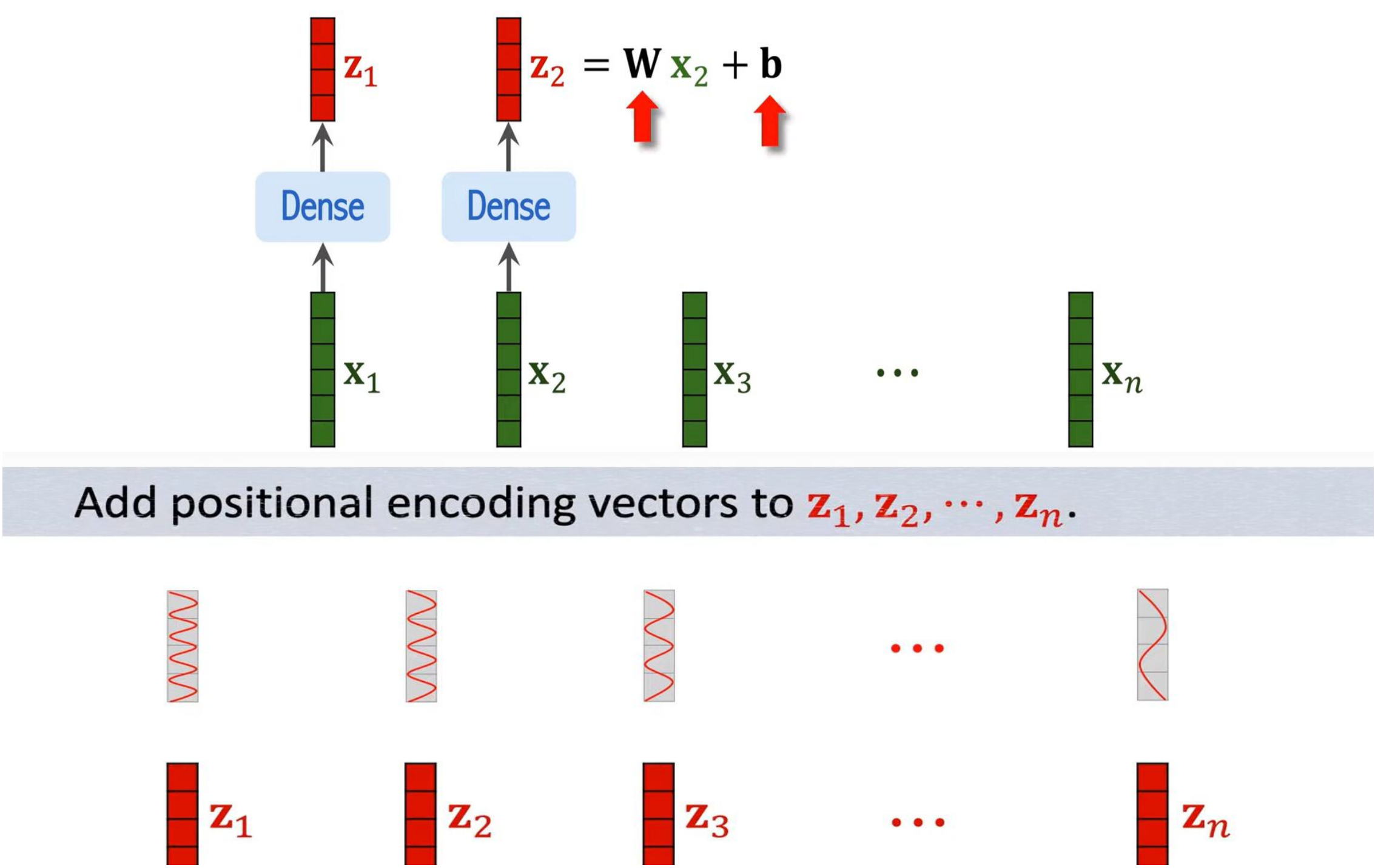


$$\vec{p}_t = \begin{bmatrix} \sin(\omega_1.t) \\ \cos(\omega_1.t) \\ \sin(\omega_2.t) \\ \cos(\omega_2.t) \\ \vdots \\ \sin(\omega_{d/2}.t) \\ \cos(\omega_{d/2}.t) \end{bmatrix}_{d\times 1}$$

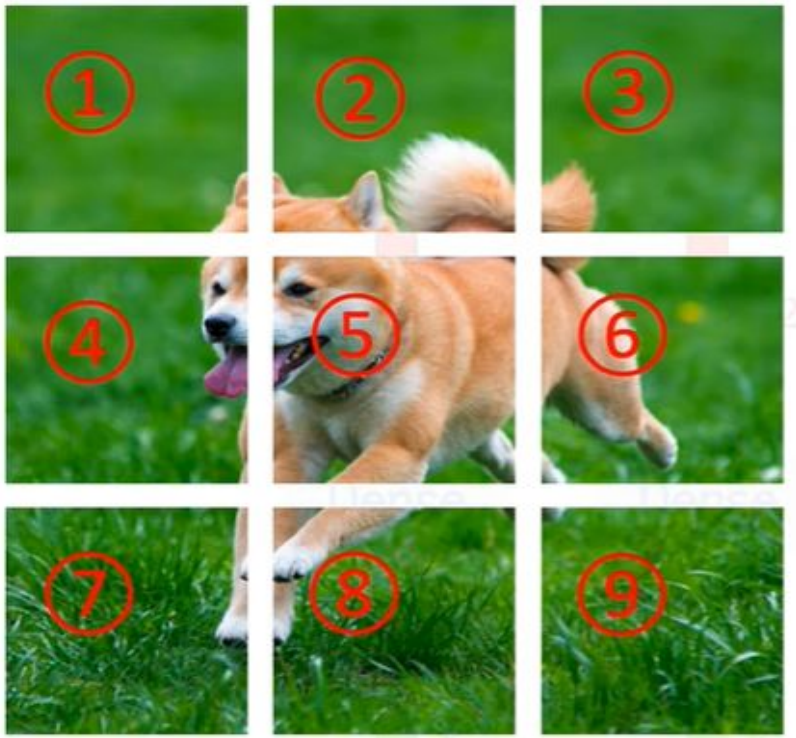
Vectorization

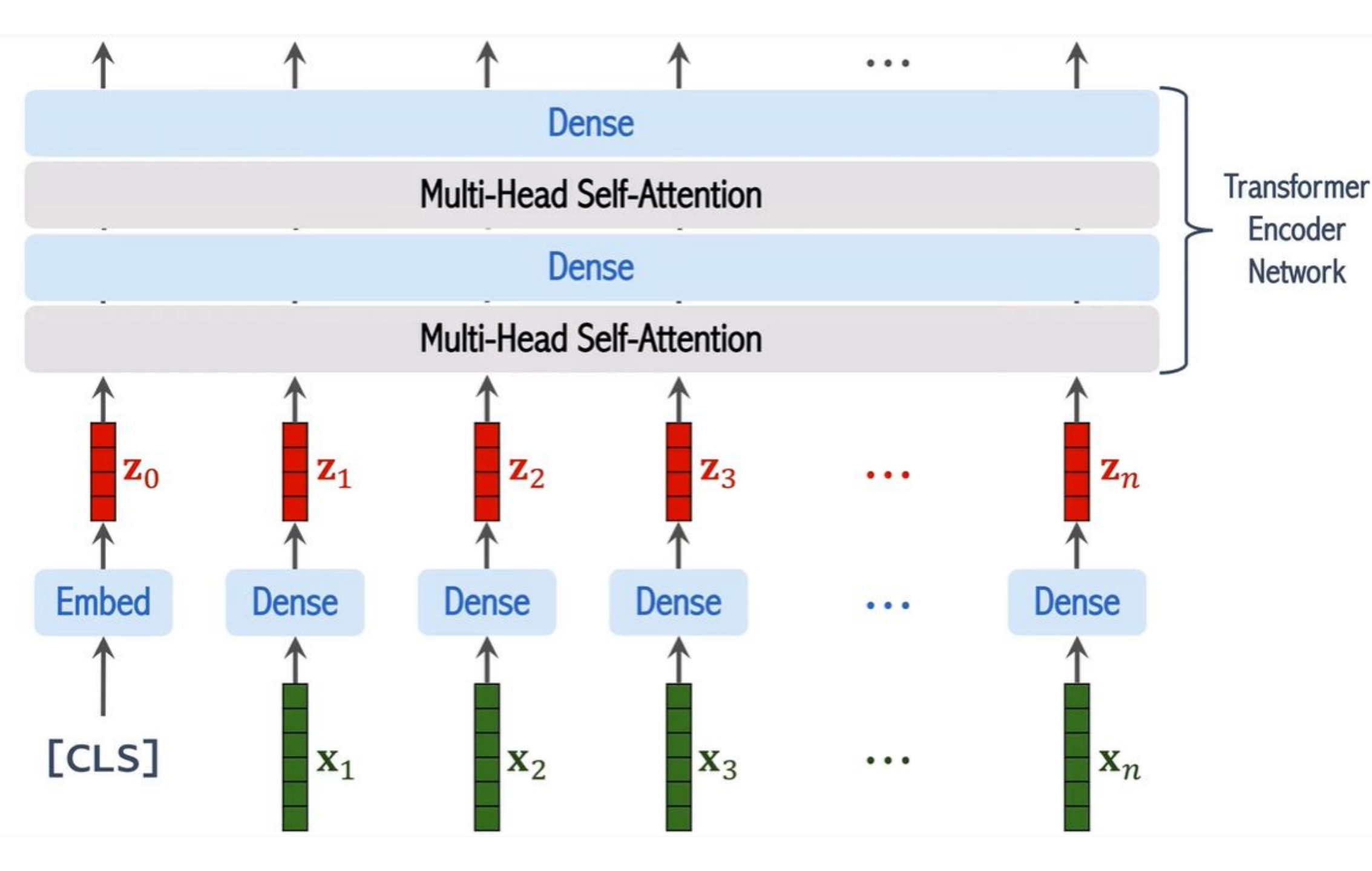
If the patches are $d_1 \times d_2 \times d_3$ tensors, then the vectors are $d_1 d_2 d_3 \times 1$.

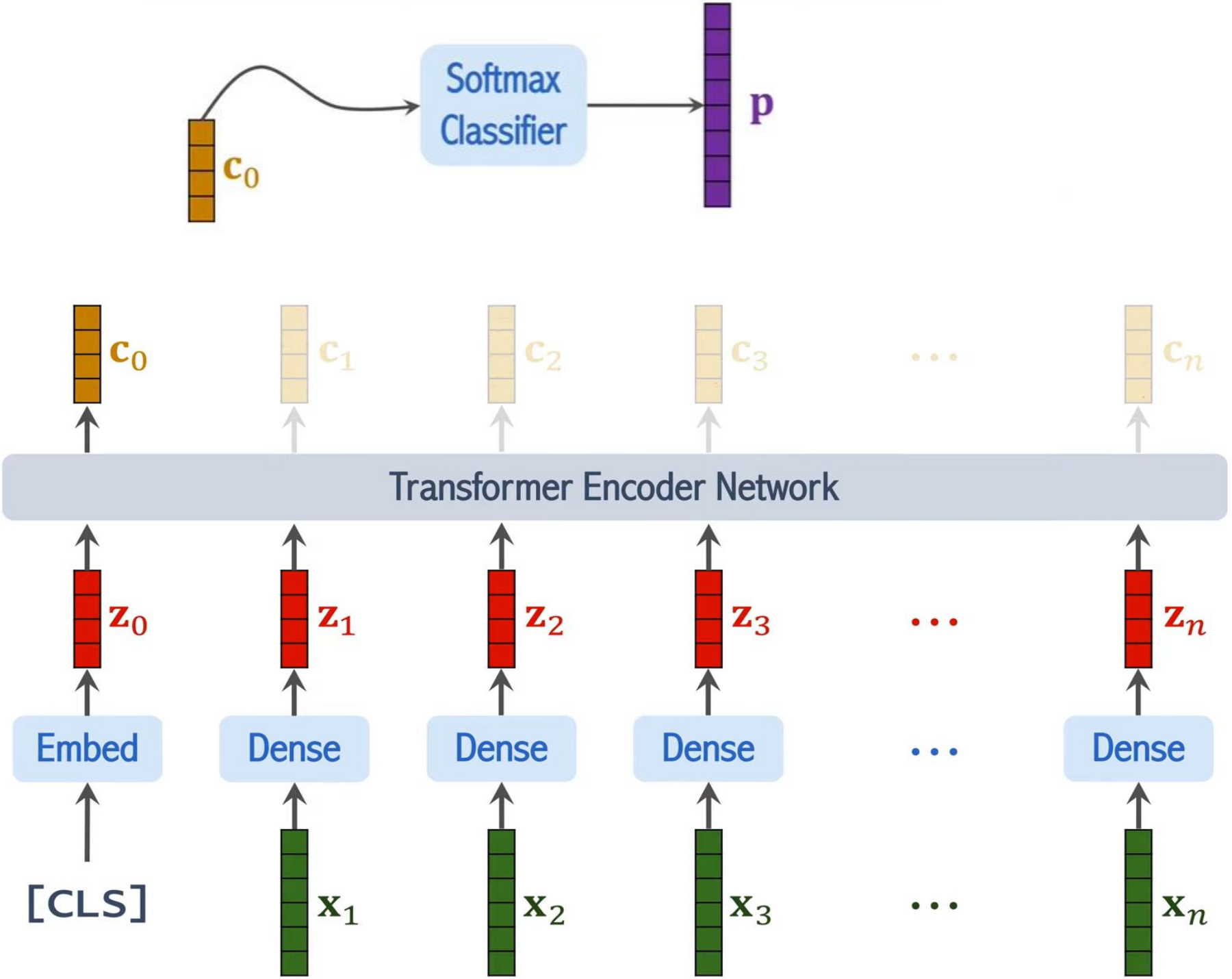



$\mathbf{z}_1$
$\mathbf{z}_2 = \mathbf{W}\,\mathbf{x}_2 + \mathbf{b}$
Dense
Dense
$\mathbf{x}_1$
$\mathbf{x}_2$
$\mathbf{x}_3$
$\cdots$
$\mathbf{x}_n$
Add positional encoding vectors to $\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_n$.
$\mathbf{z}_1$
$\mathbf{z}_2$
$\mathbf{z}_3$
$\cdots$
$\mathbf{z}_n$

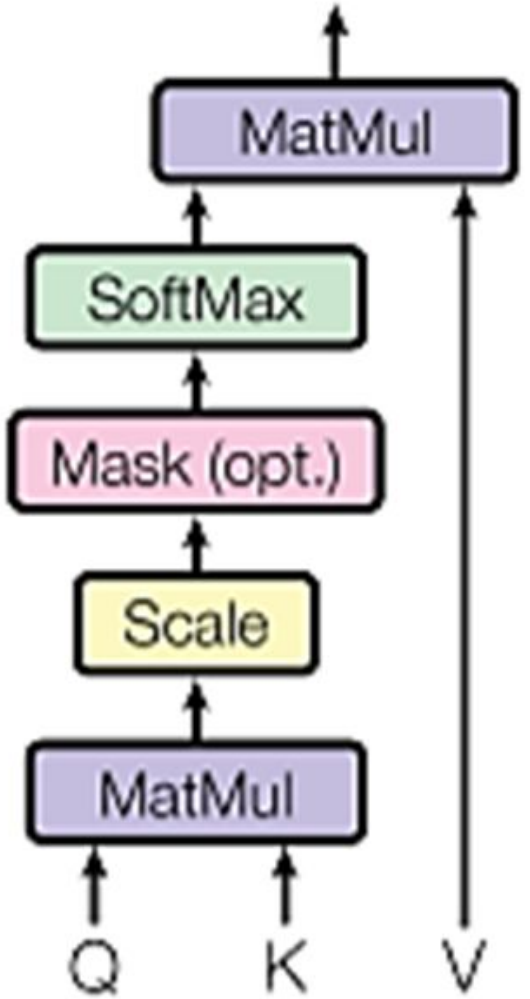
Add positional encoding vectors to $\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_n$. (Why?)




Dense
Multi-Head Self-Attention
Dense
Multi-Head Self-Attention
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


Softmax Classifier
p
c_0
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

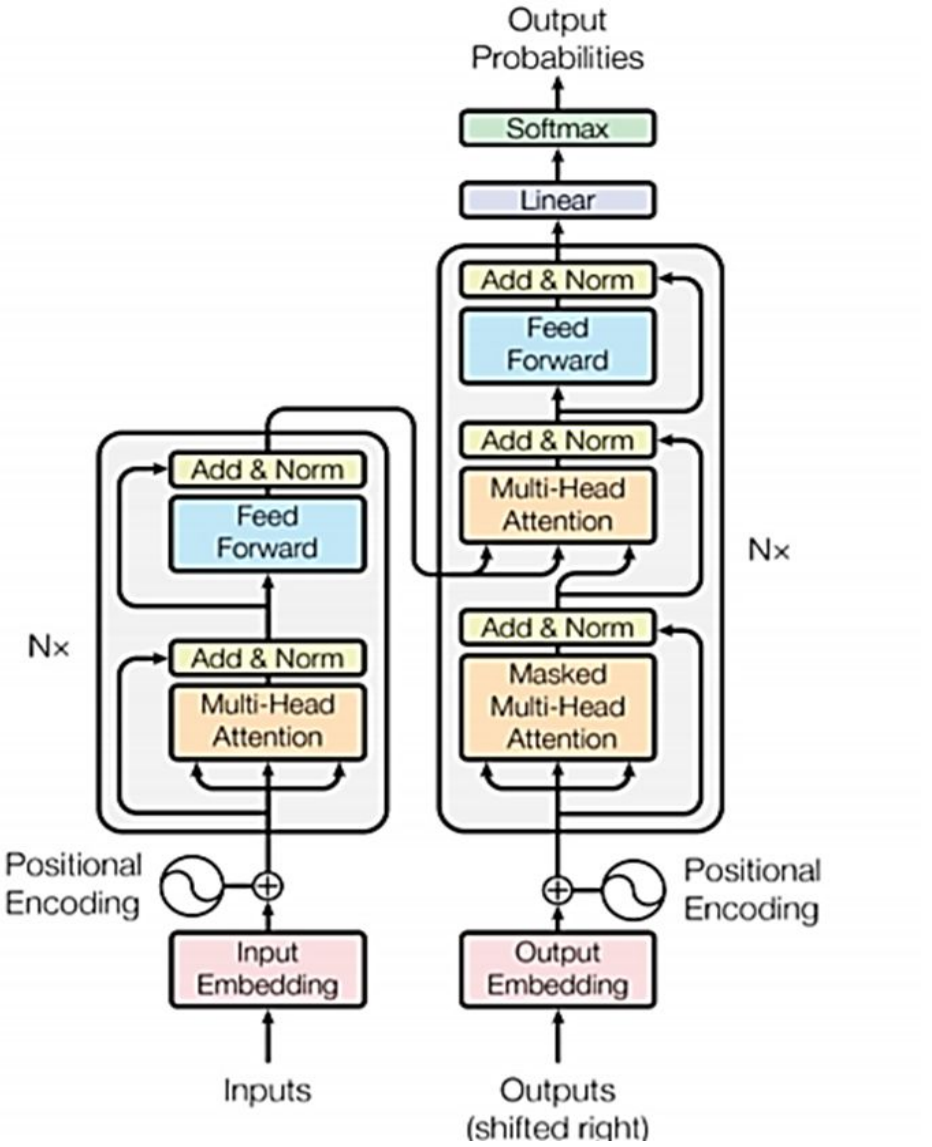
Self Attention Mechanism



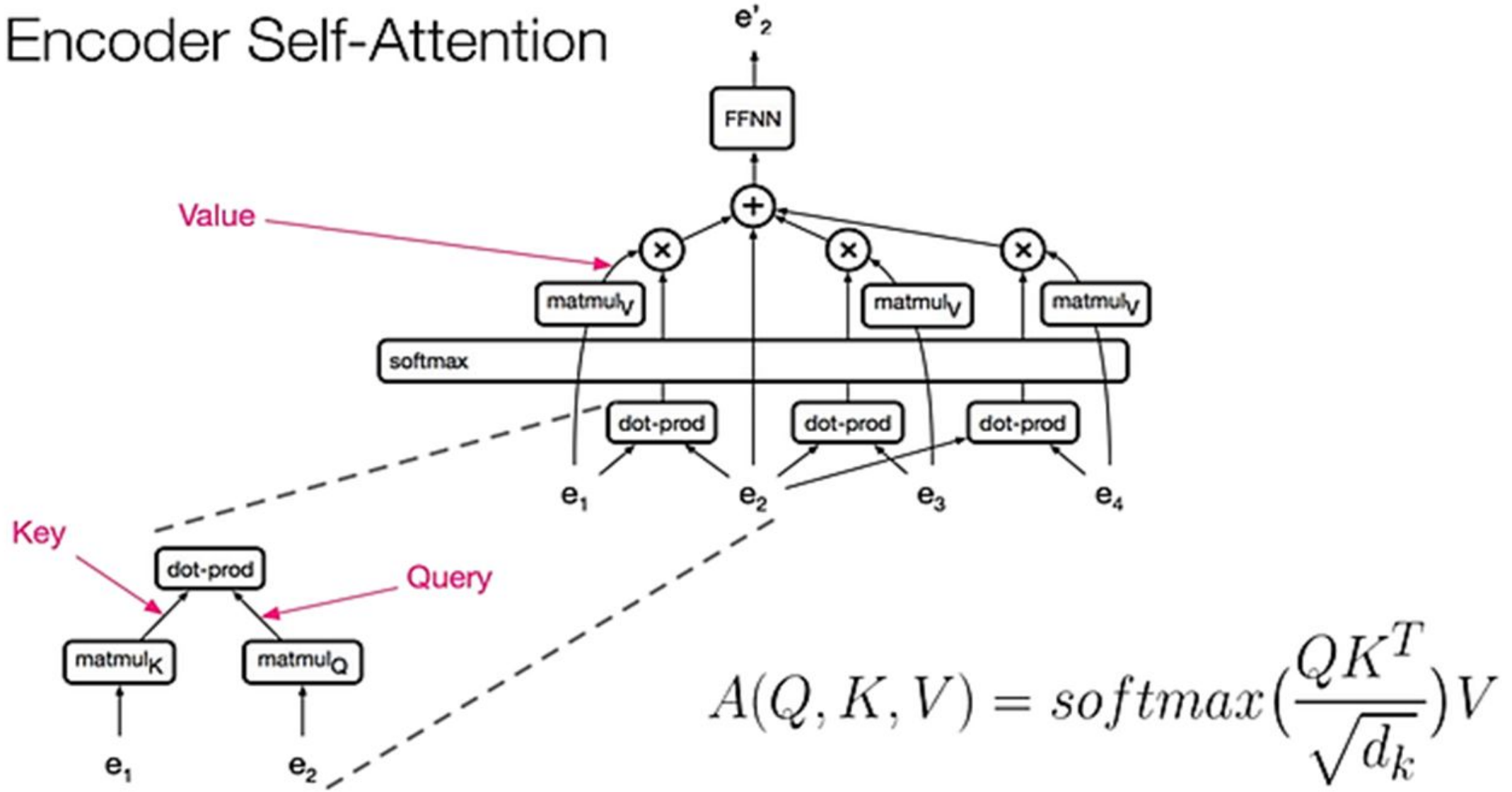
- Multiplying each of the encoder input vectors with three weights matrices (W(Q), W(K), W(V)) that we trained during the training process.
- Multiply the Query vector of the current input with the key vectors from other inputs.
- Divide the score by square root of dimensions of the key vector (d_k).
- Softmax function on all self-attention scores we calculated wrt the query word
- We multiply the value vector.
- Sum up the weighted value vectors.

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

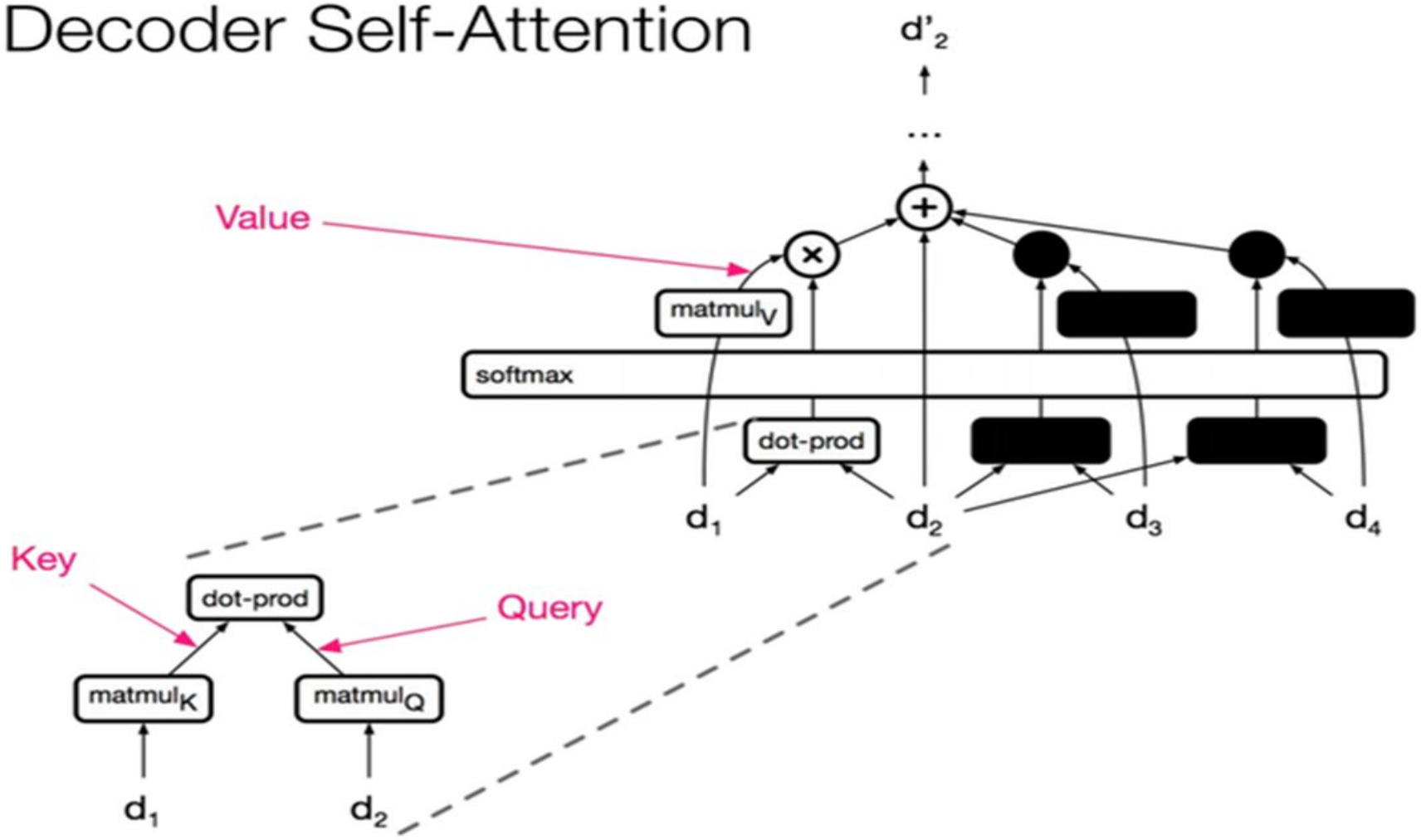
Encoder Decoder Attention



In encoder-decoder attention layer the queries come from the previous decoder layer while the keys and values come from the encoder output. This allows each position in the decoder to give attention to all the positions of the input sequence.

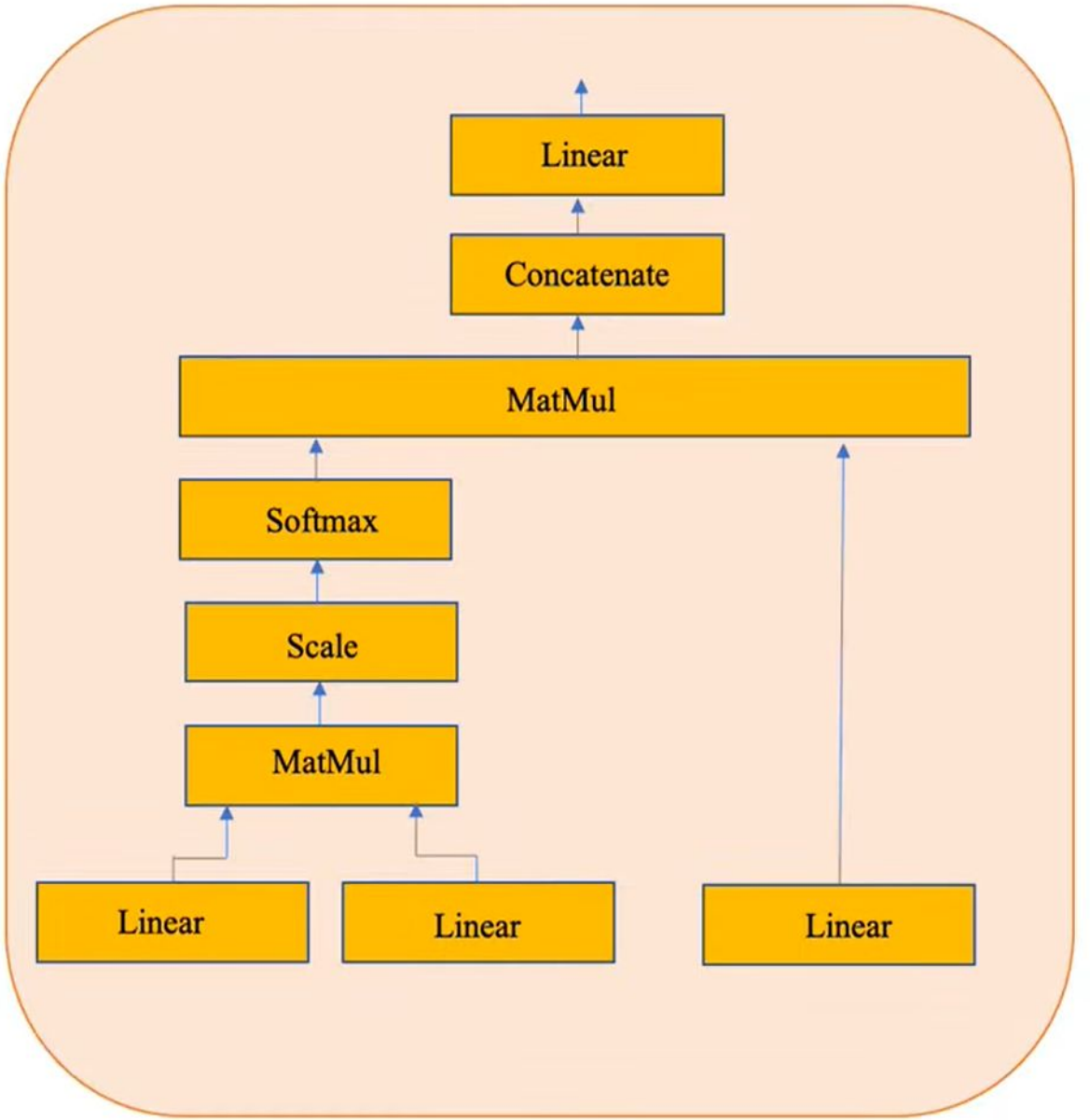


Encoder Self-Attention layer receives key, value, and query input from the output of the previous encoder layer. Each position in the encoder can get attention score from every position in the previous encoder layer.



Decoder Self-Attention is similar to self-attention in encoder where all queries, keys, and values come from the previous layer. The self-attention decoder allows each position to attend each position up to and including that position. The future values are masked with (-Inf). This is known as masked-self attention.

Multi-Head Attention



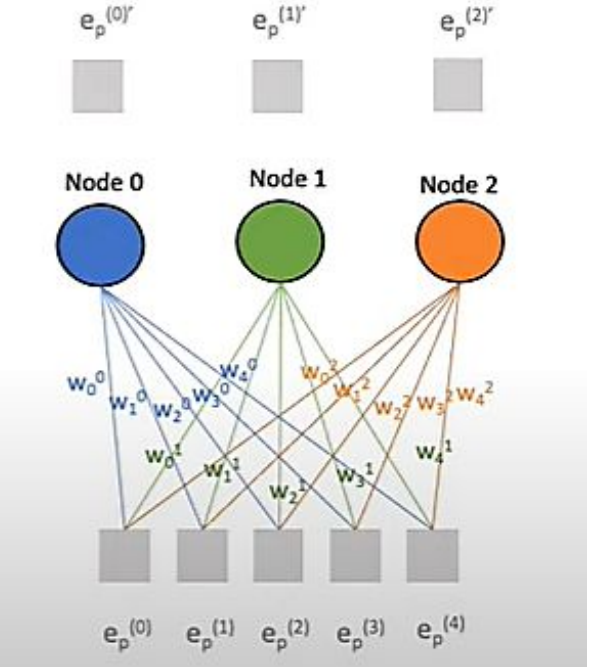
Linear Layer

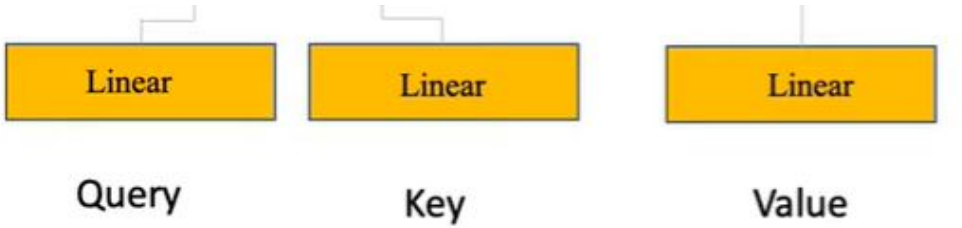
i) Mapping inputs onto the outputs

ii) Changing matrix/vector dimensions

Linear Layer

$W_0^0 = 0.6$



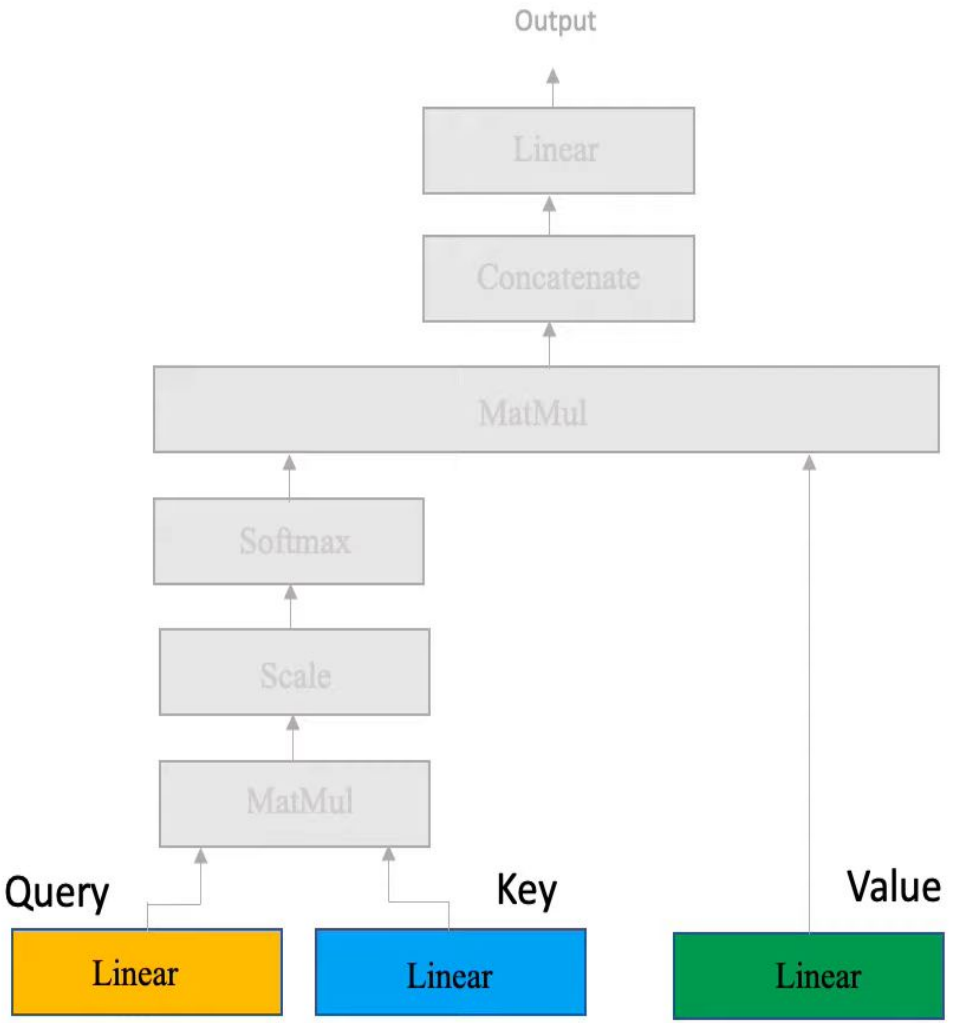


Cosine Similarity

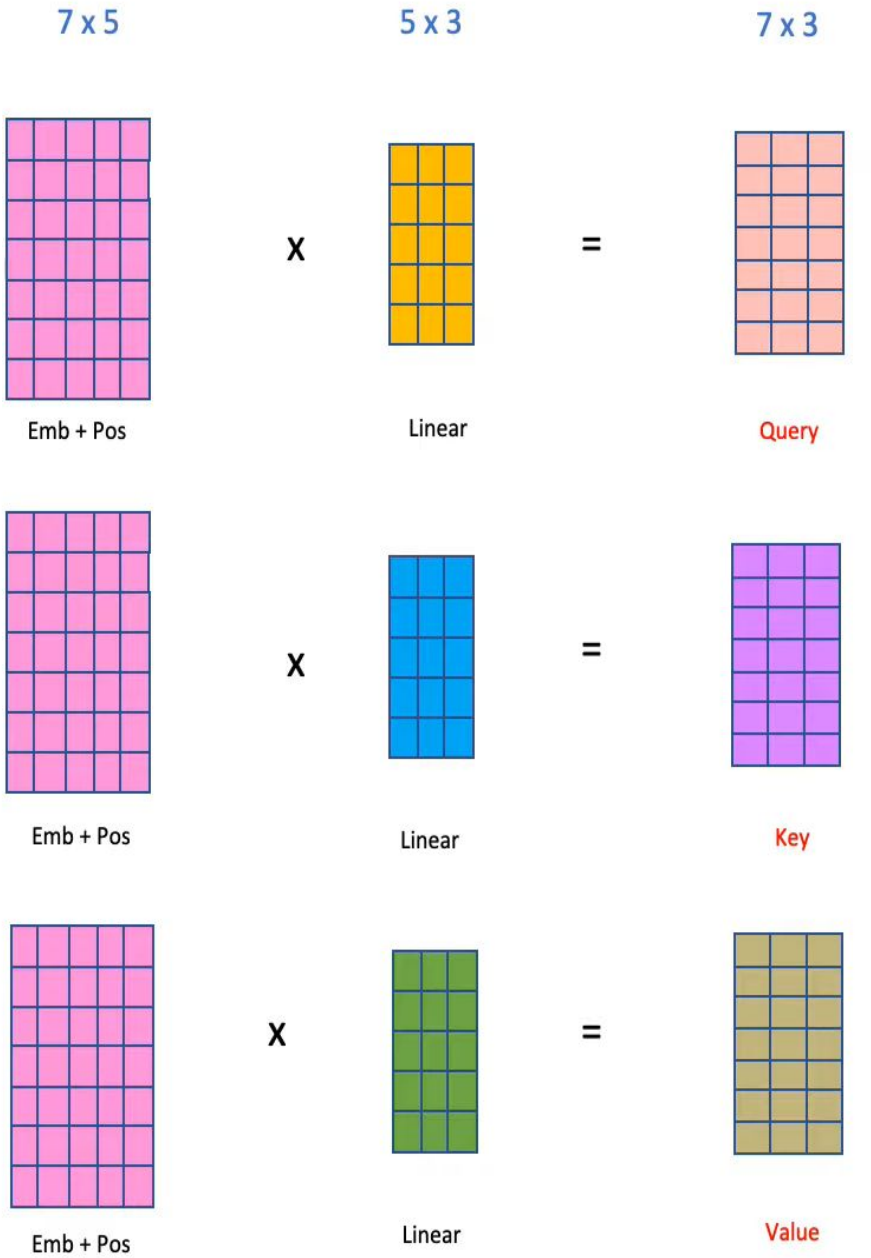
$$\text{Cos}\,(A, B) = \frac{A \,.\, B}{|A|\,|B|}$$

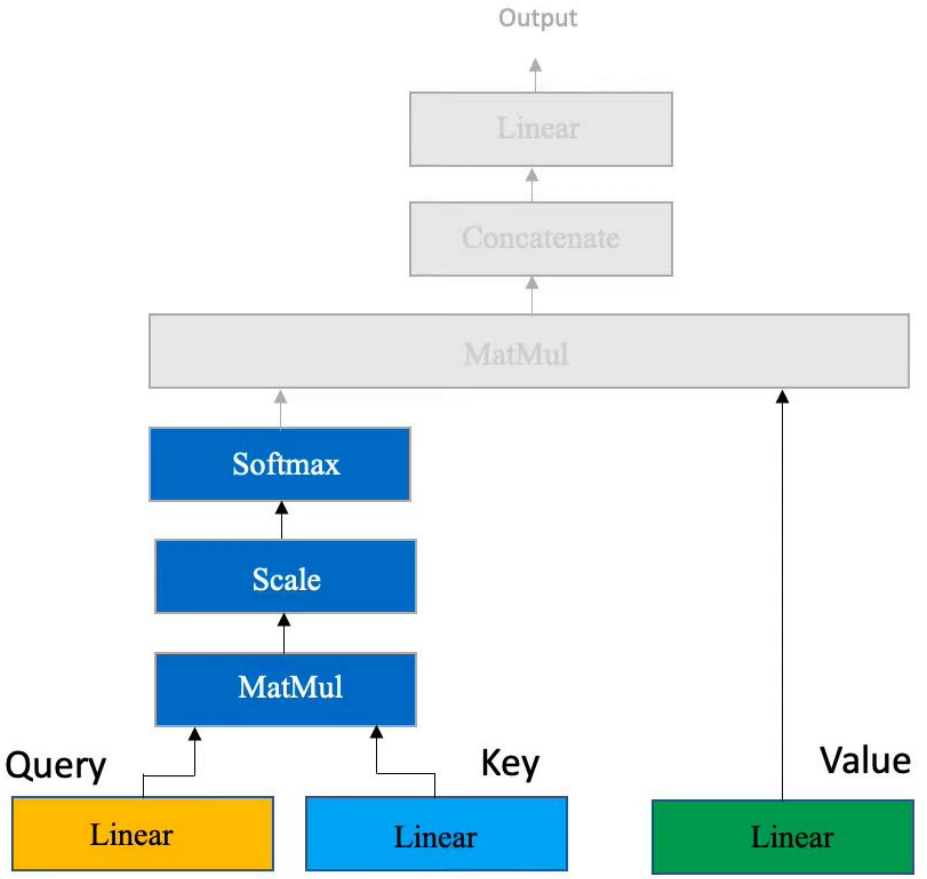
Similarity b/w Matrices

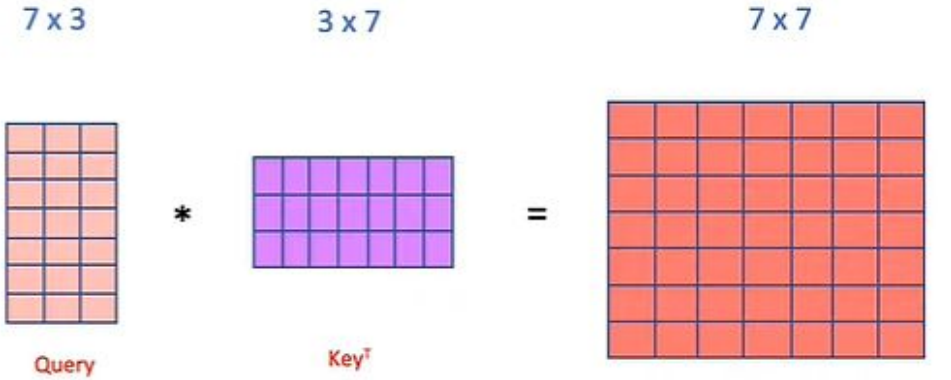
$$\text{similarity}\,(A, B) = \frac{A \,.\, B^{T}}{scaling}$$



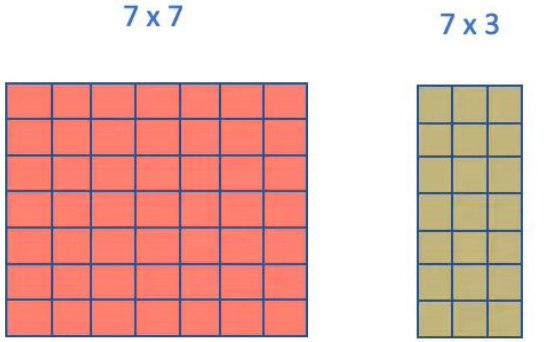
Multi-Head Attention







Attention Filter



$$\text{softmax}\left(\frac{Q.K^T}{\sqrt{d_k}}\right) \quad V$$

Intuition

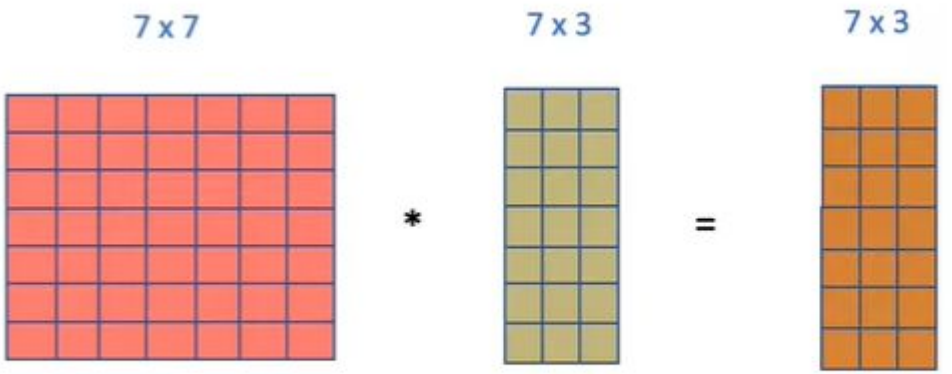
Attention Filter

Original Image

Filtered Image

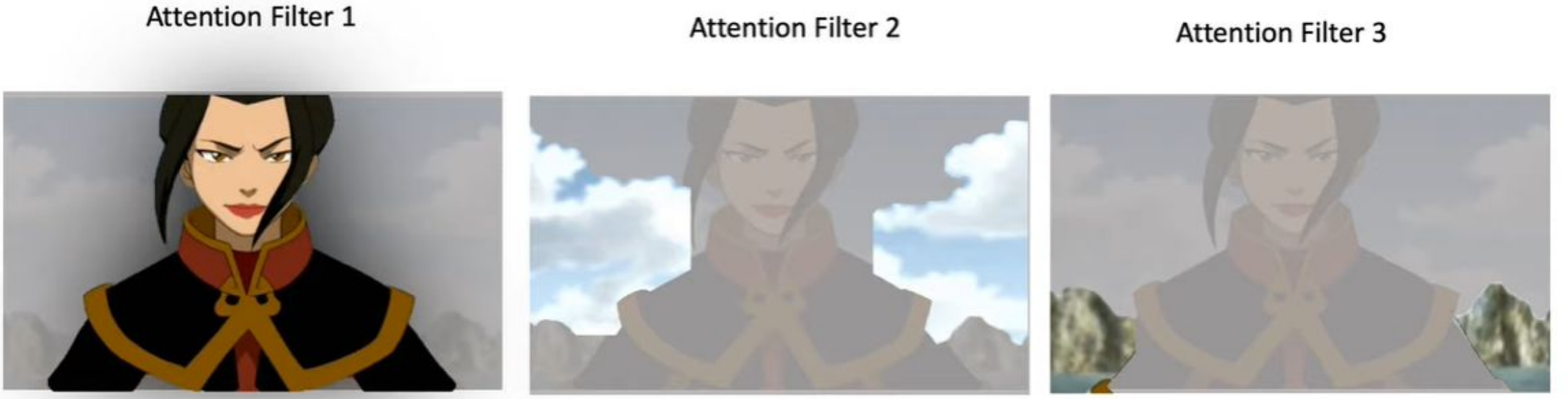
*

=

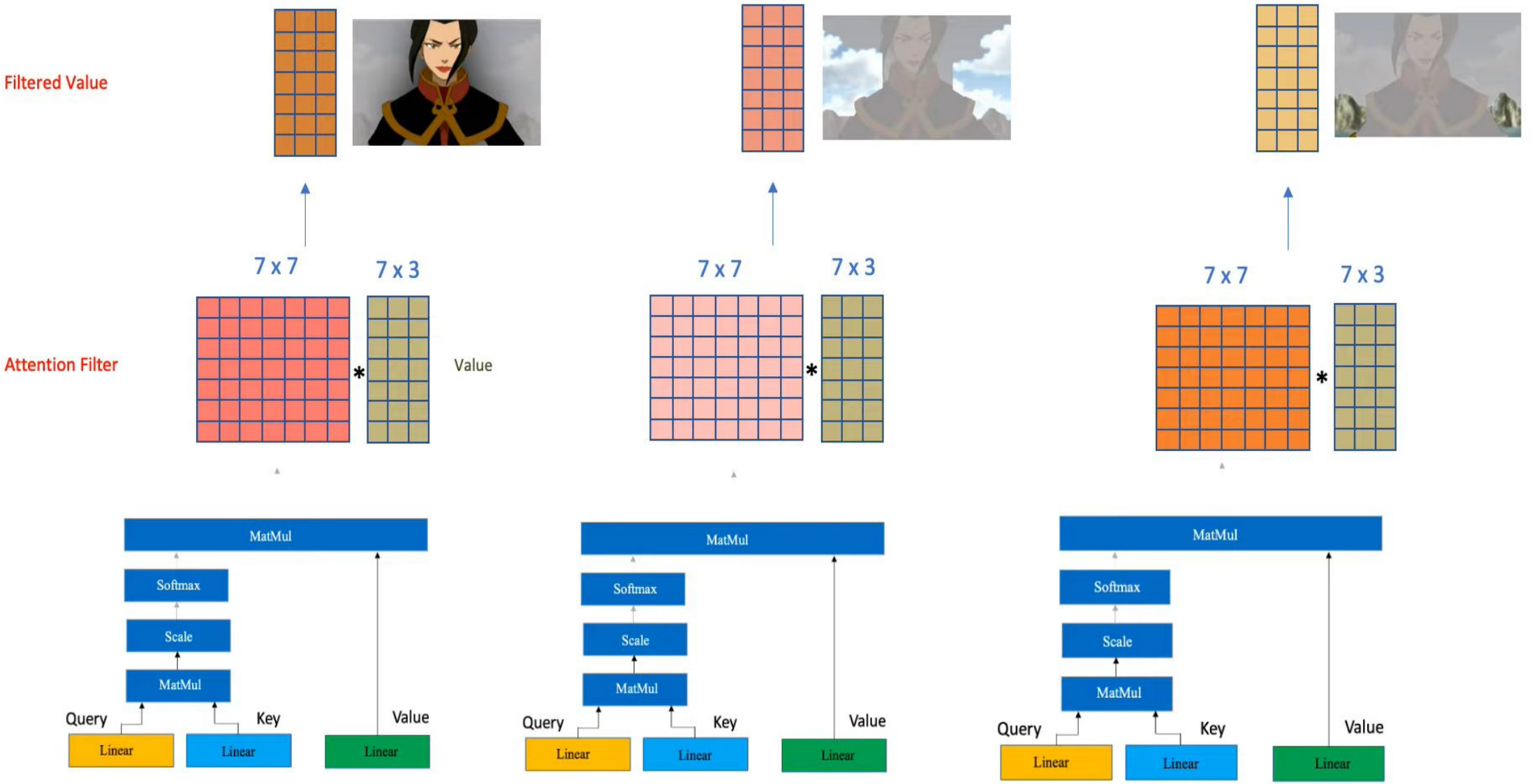


$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{Q.K^T}{\sqrt{d_k}}\right).V$$

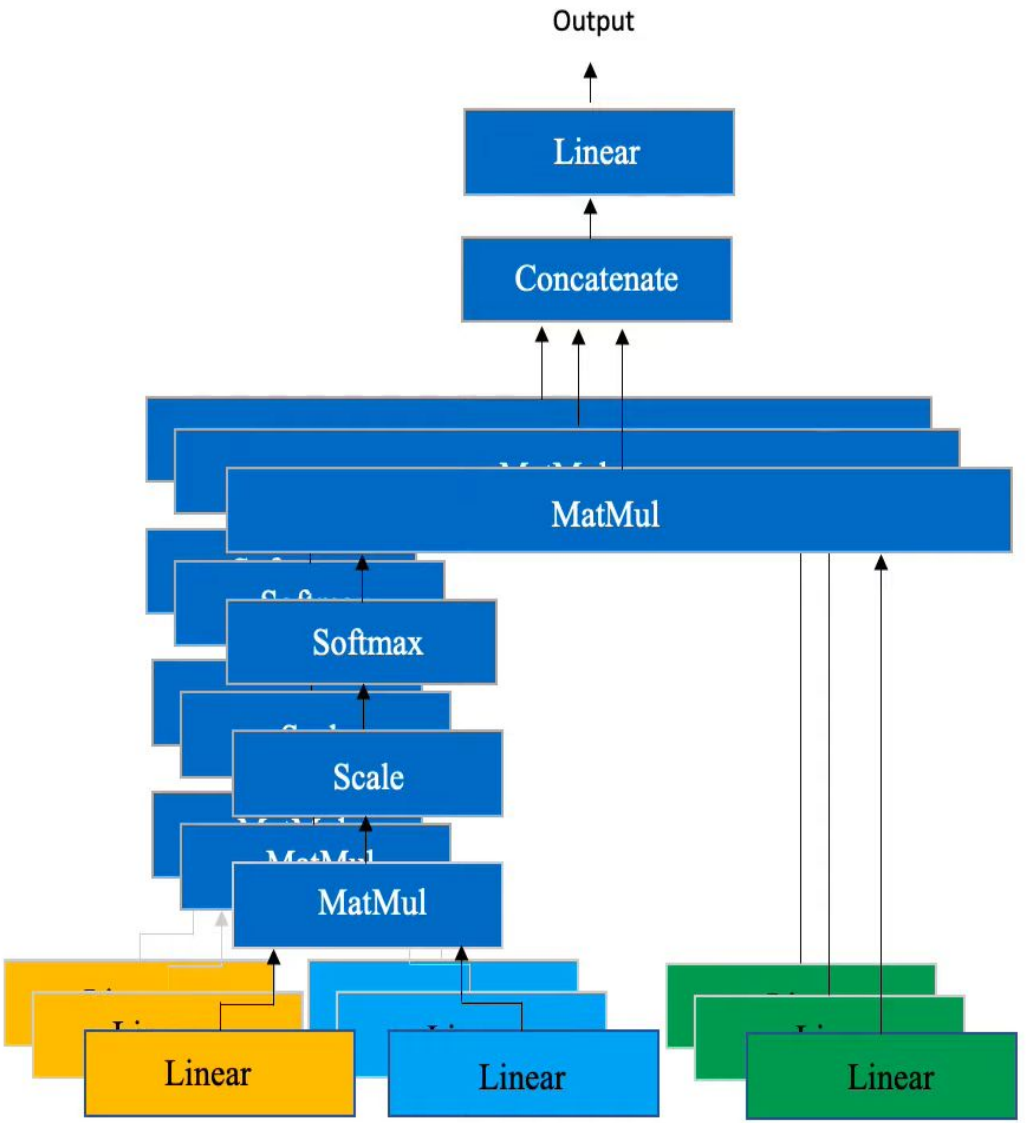
Intuition for Multi-head Attention

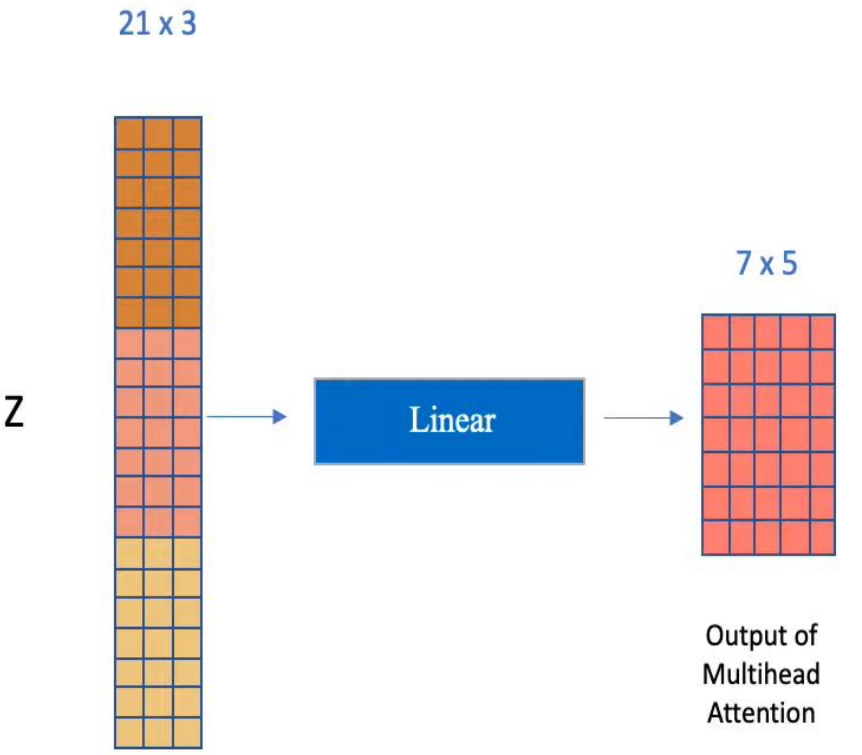


Multi-Head Attention



Multi-Head Attention





Training of Transformer Model

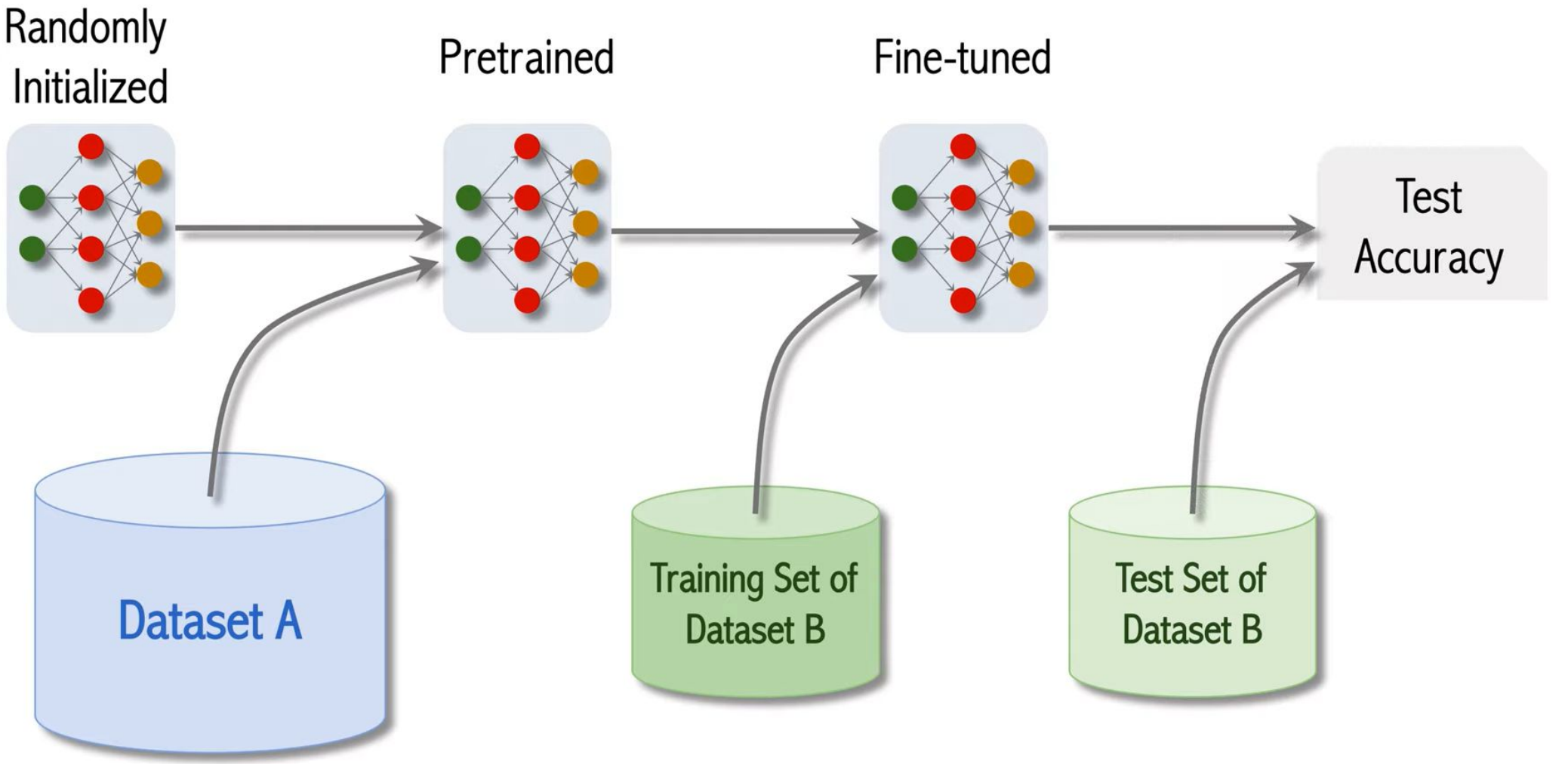
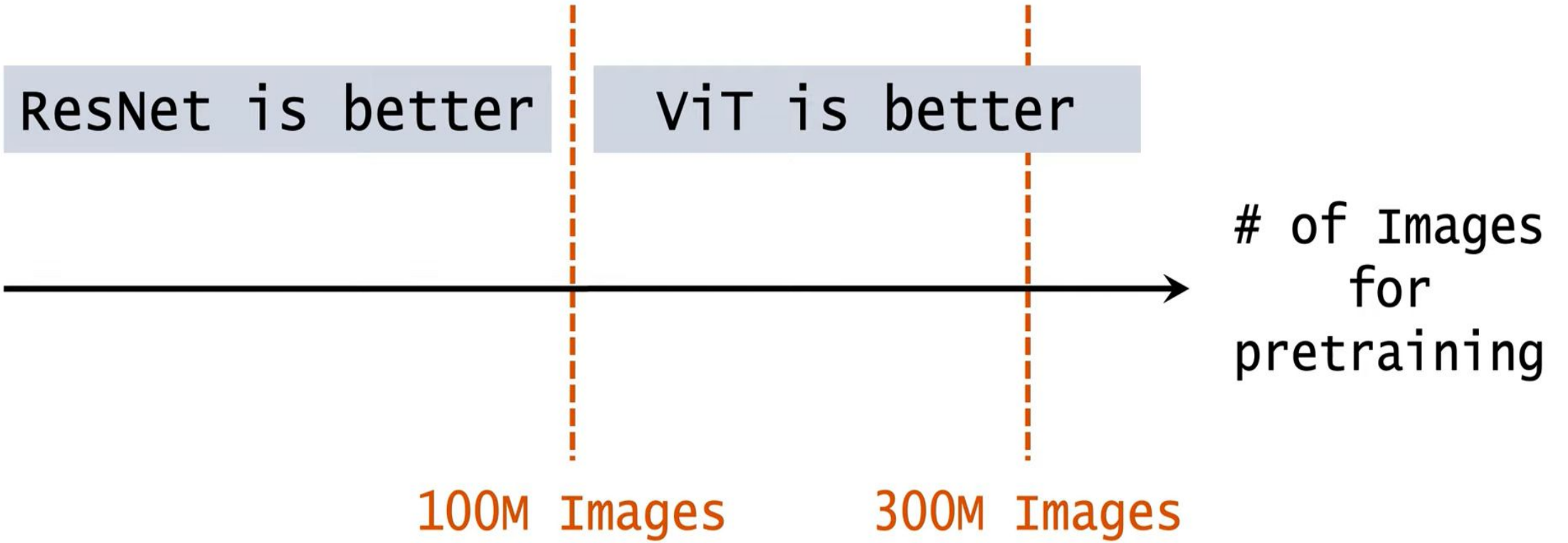


Image Classification Accuracies

- 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.

Image Classification Accuracies

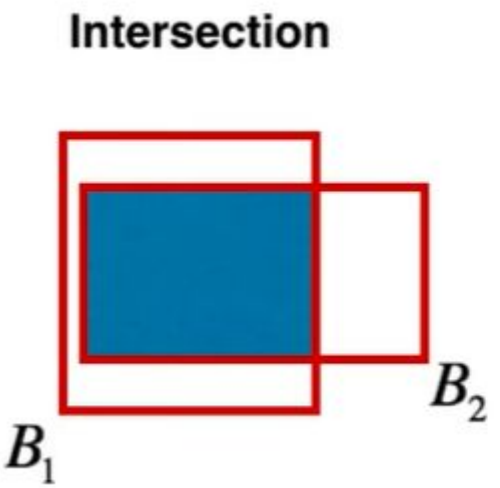


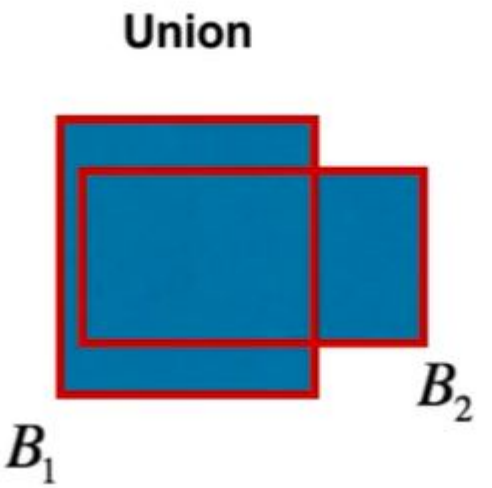
Architectures used for object - localization

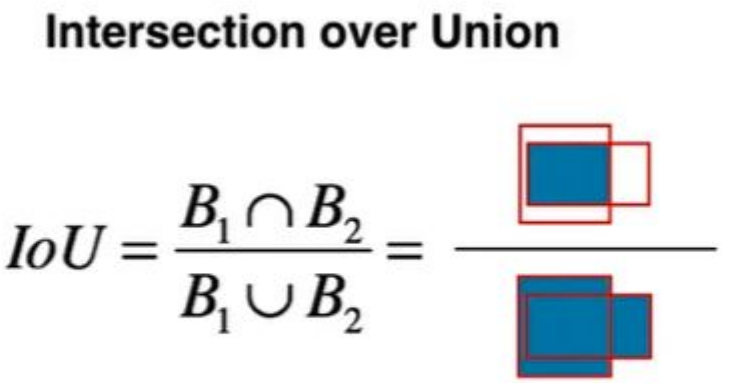
- R-CNN
- Fast R-CNN
- Faster R-CNN
- YOLO
- SSD

IOU

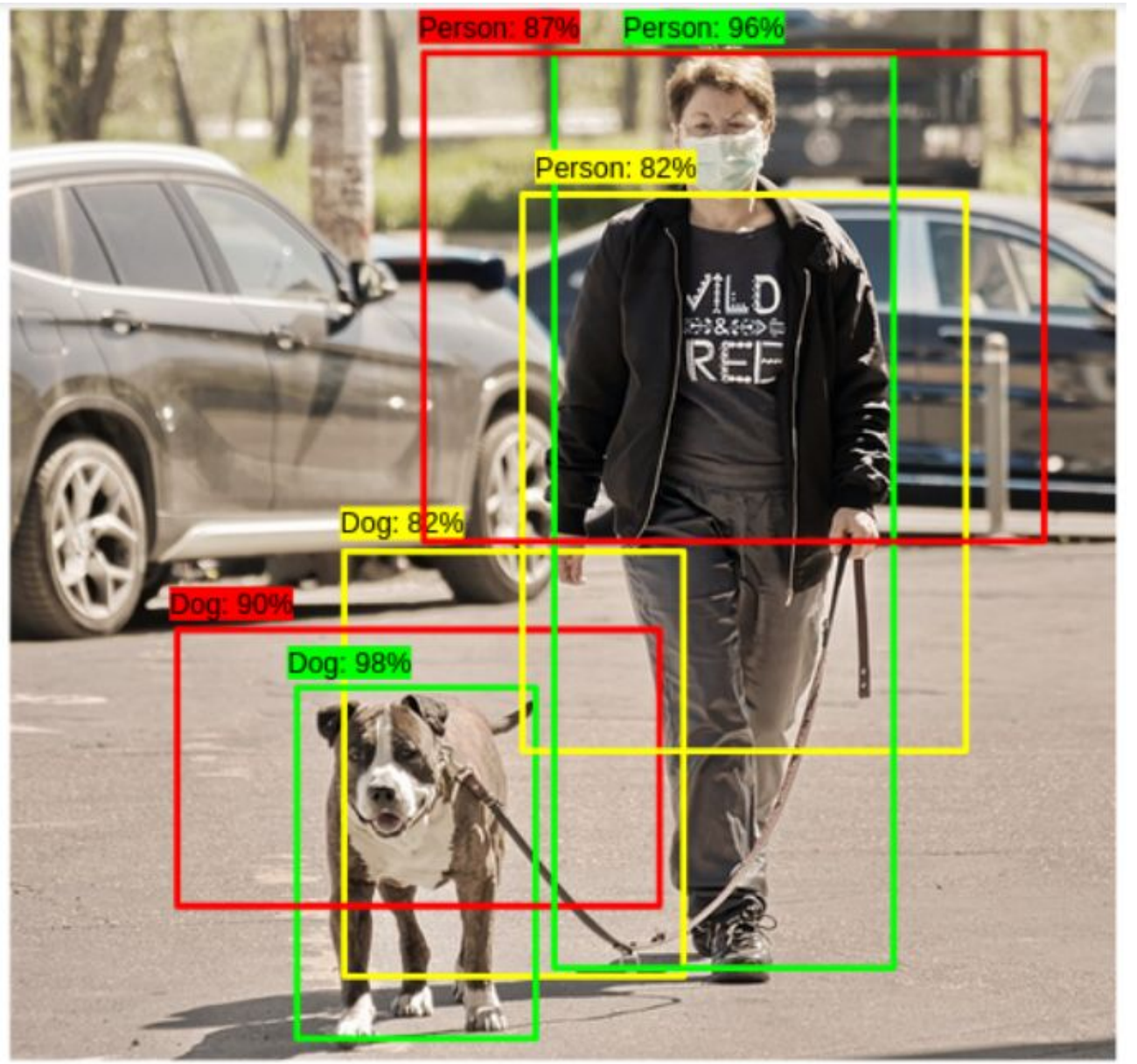
- iou = 1 (best fit), 0 (worst fit)



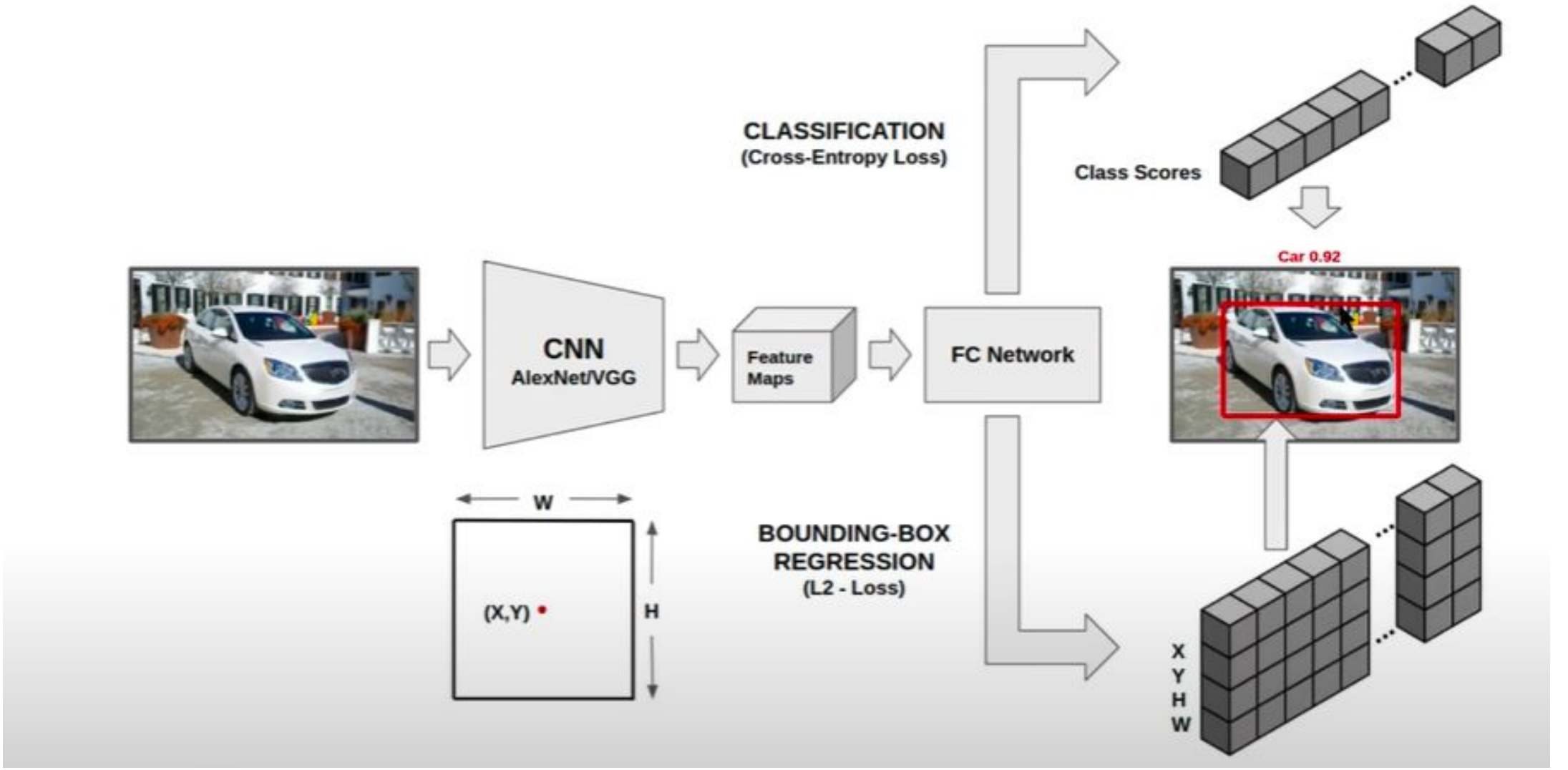




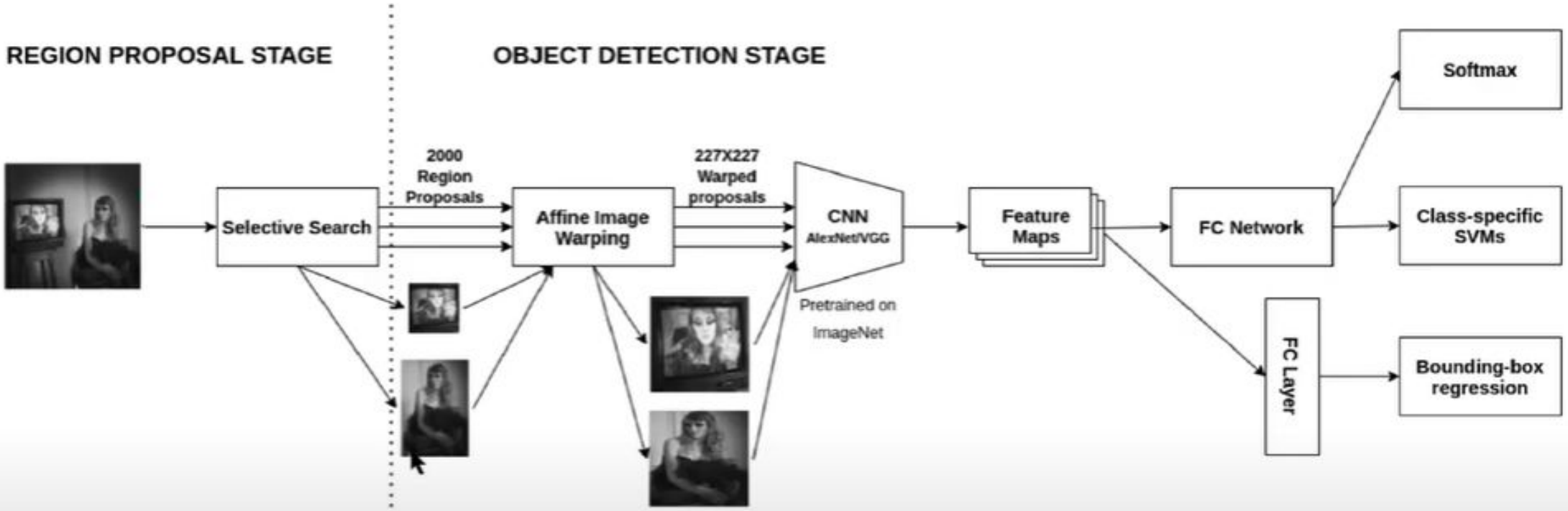
Non Maximum Suppression



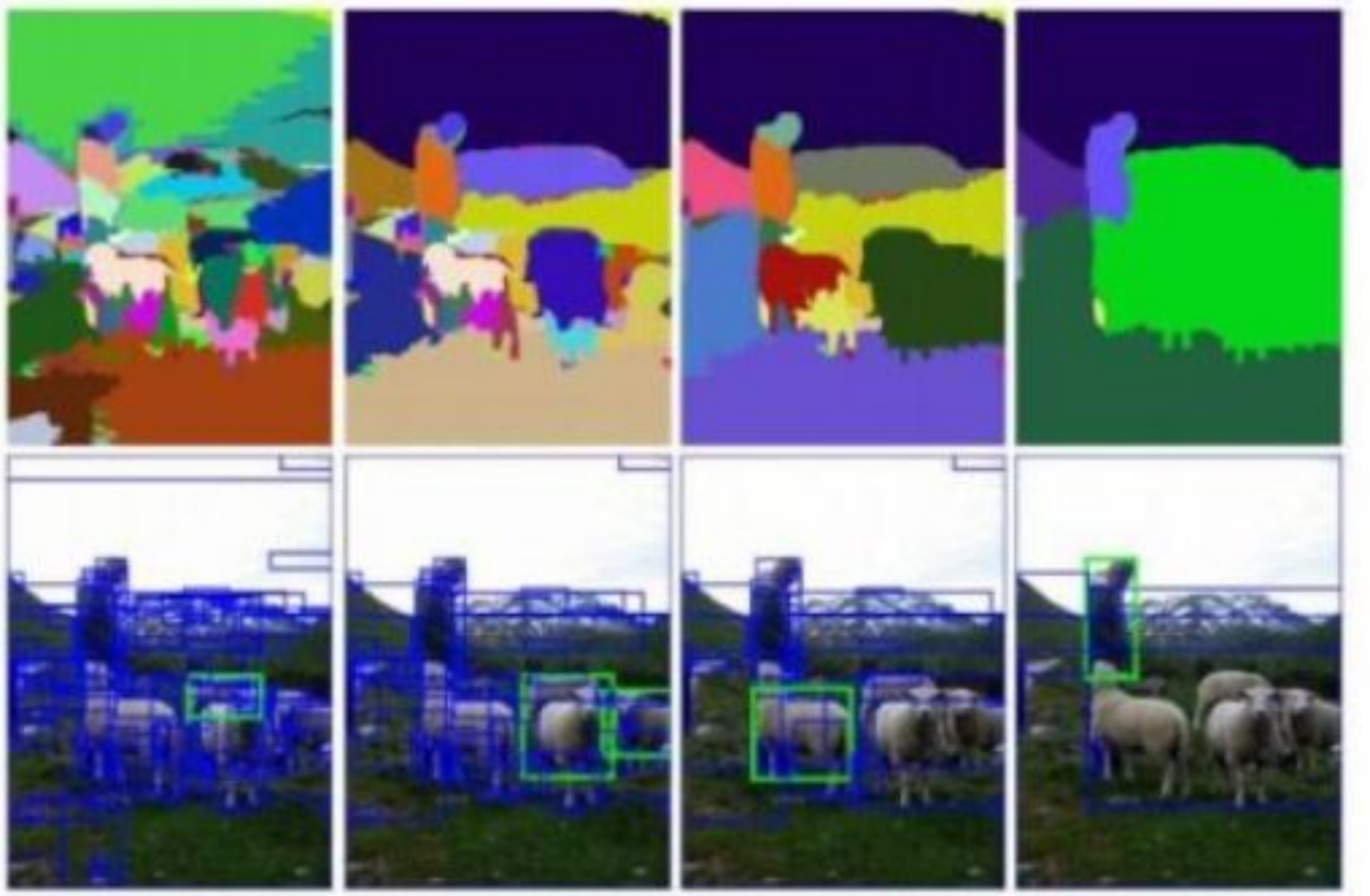
Object detection using CNN's and Bounding box regression



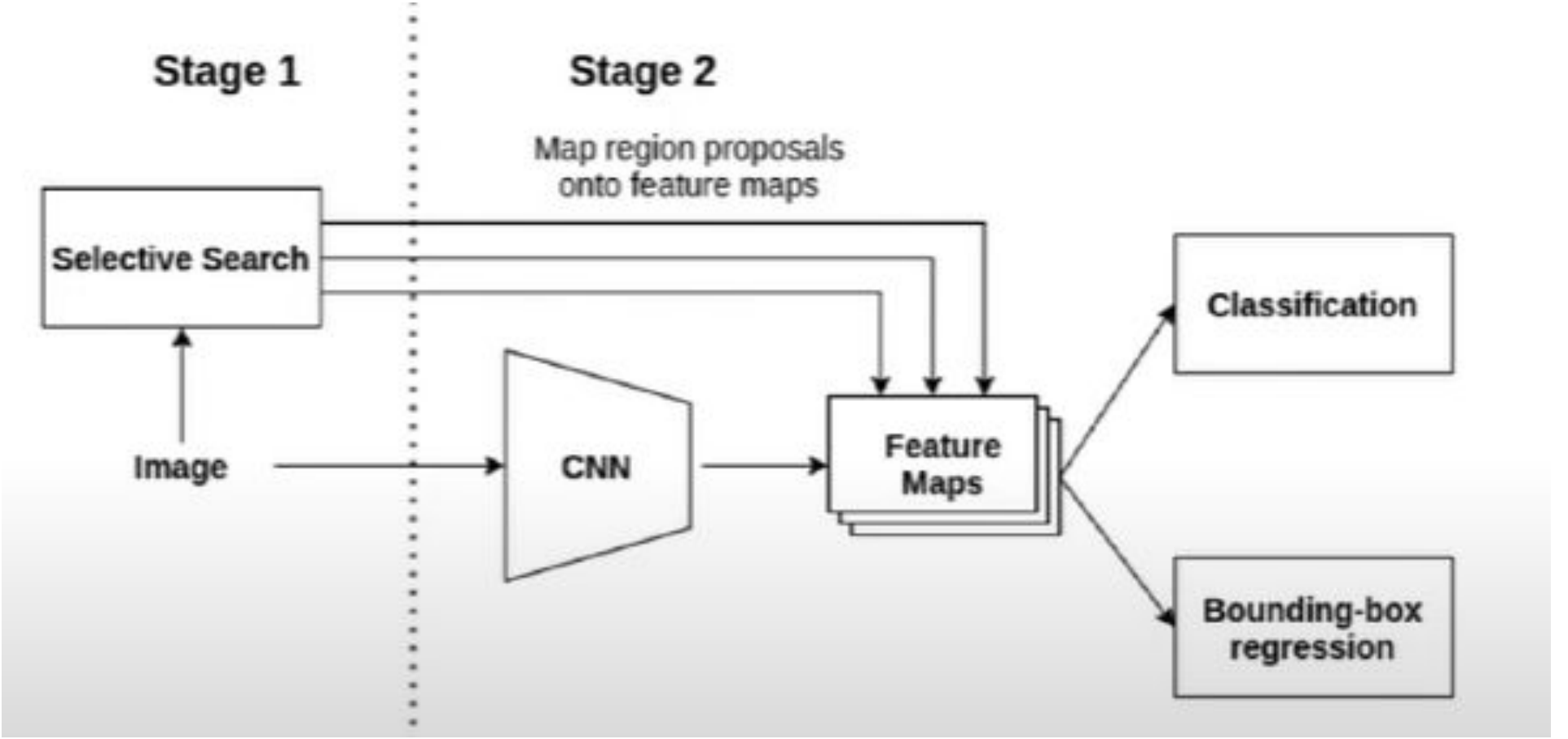
Object detection methods



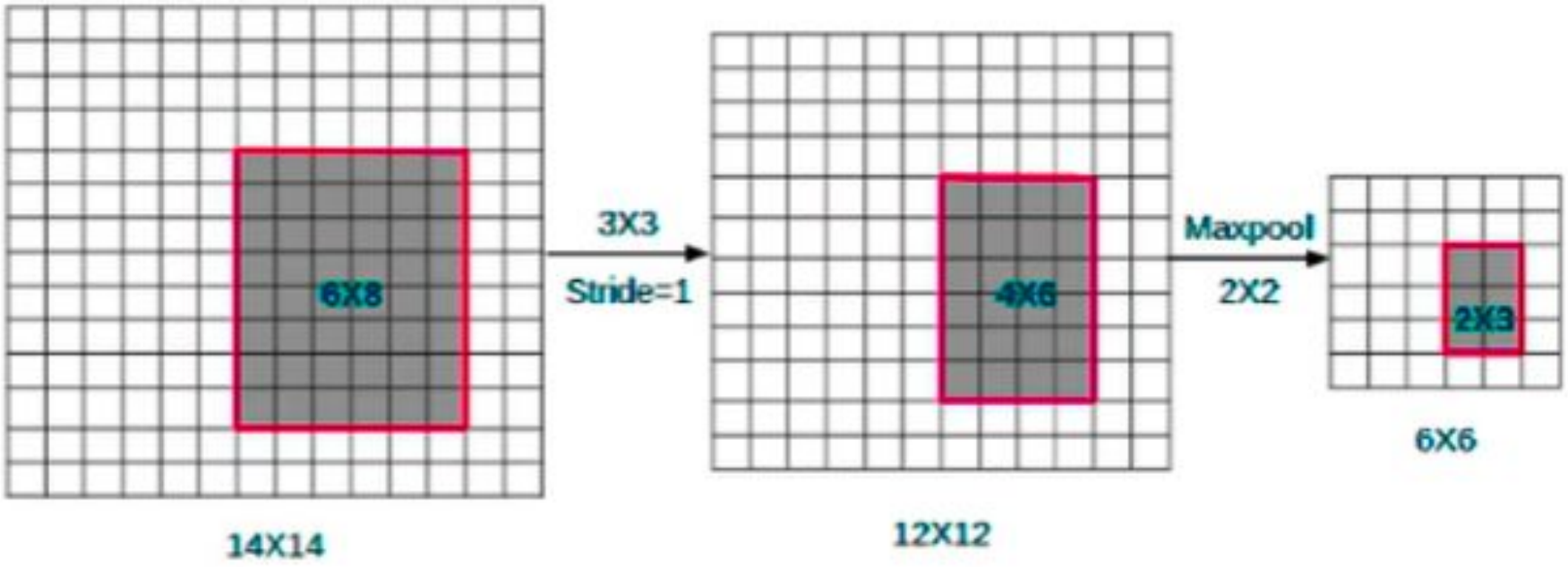
Selective search

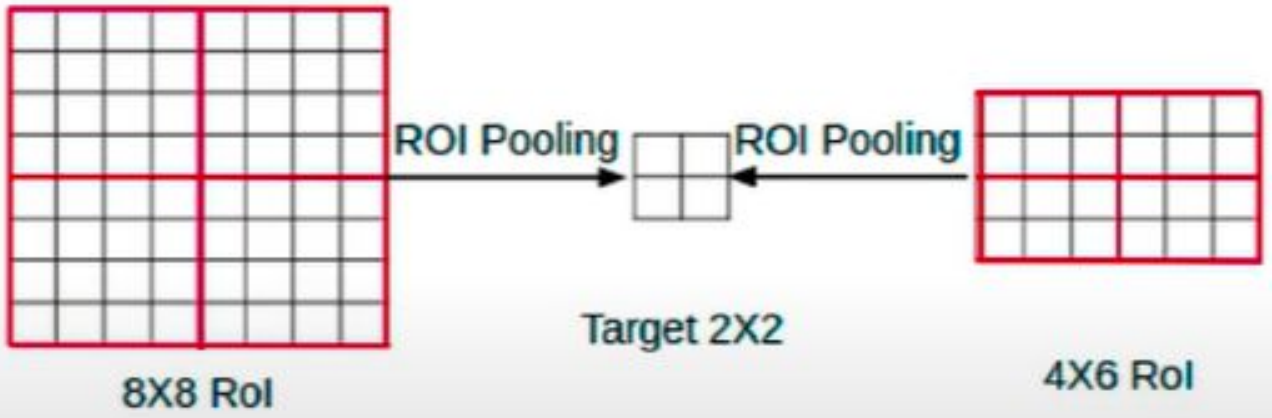


Fast - RCNN

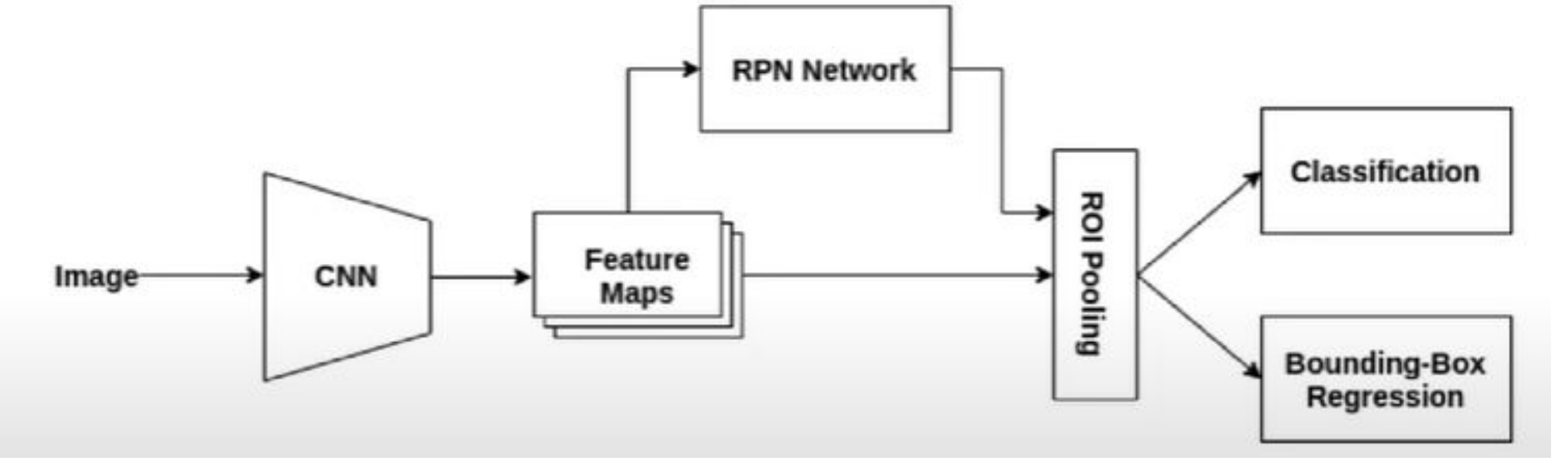

Stage 1
Stage 2
Map region proposals
onto feature maps
Selective Search
Image
CNN
Feature
Maps
Classification
Bounding-box
regression

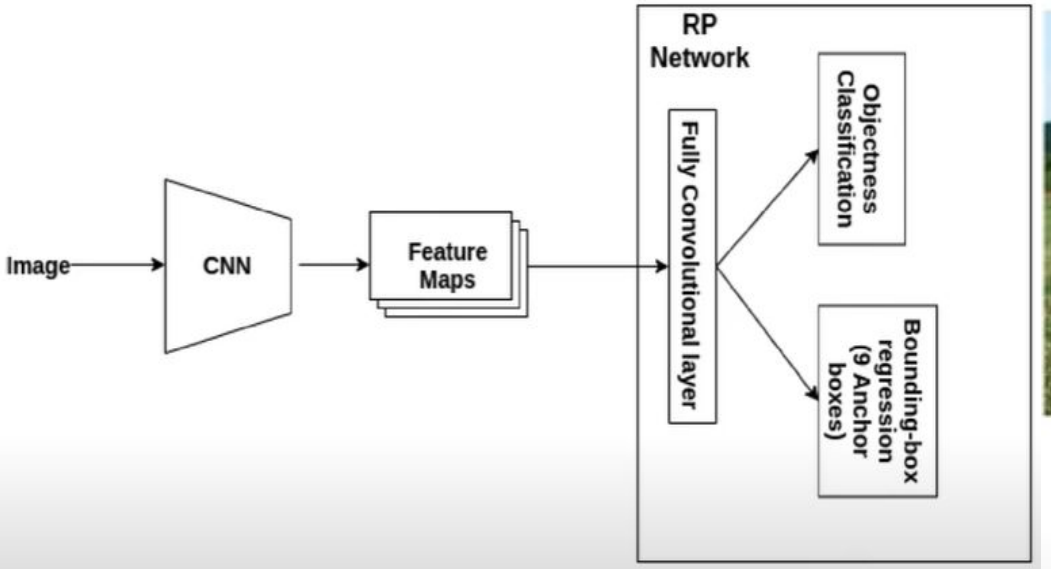
ROI projection and pooling





Faster - RCNN





Input Image
(e.g. 3 x 640 x 480)

CNN

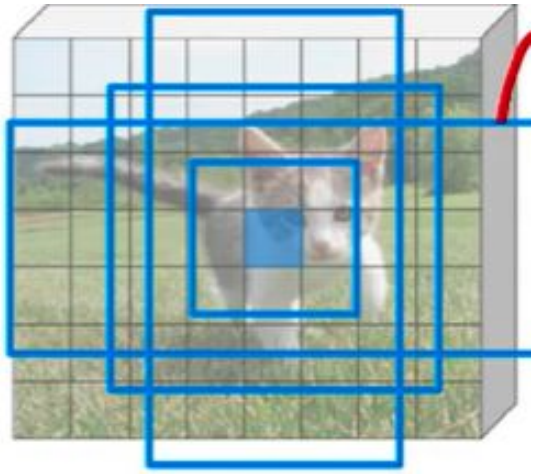
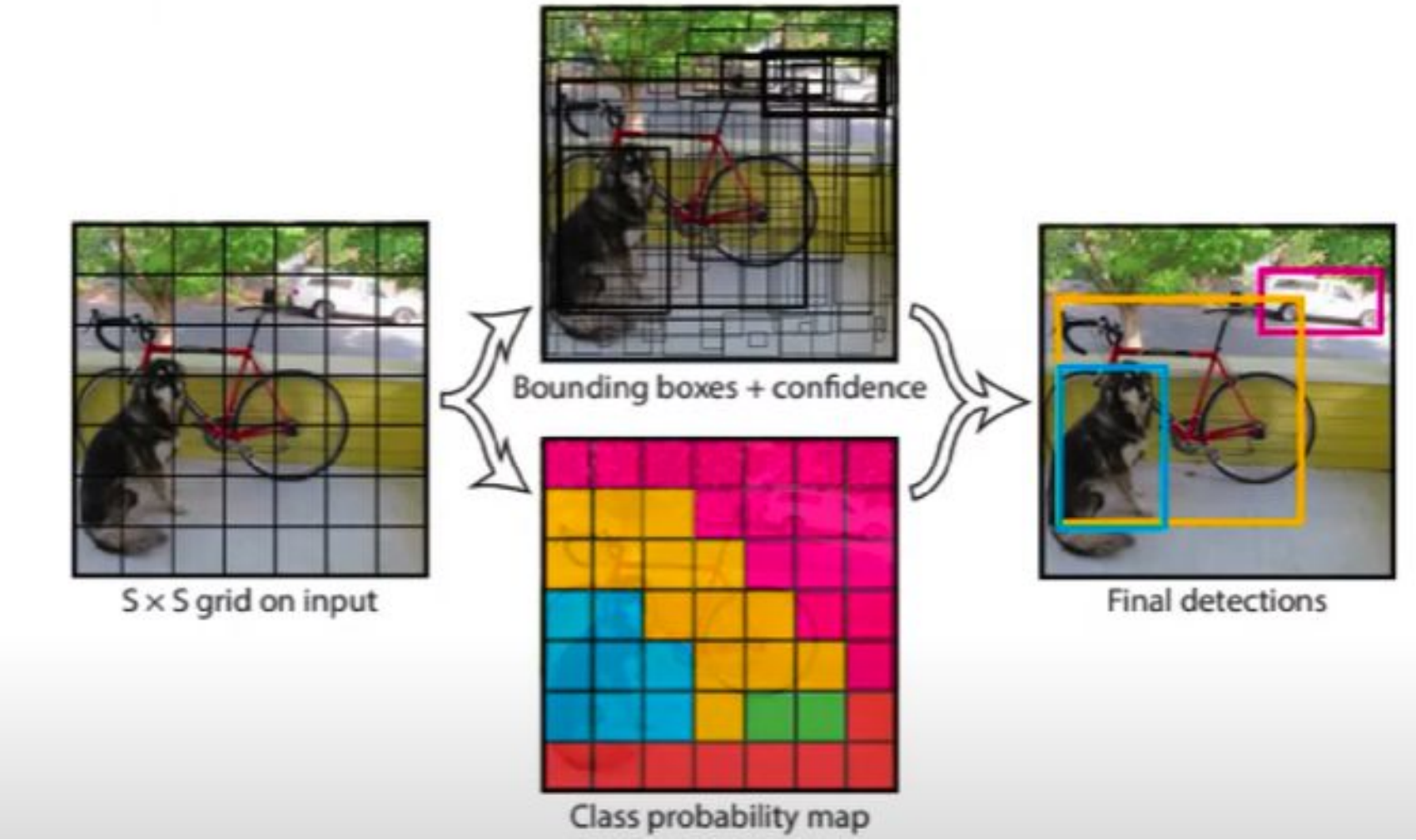
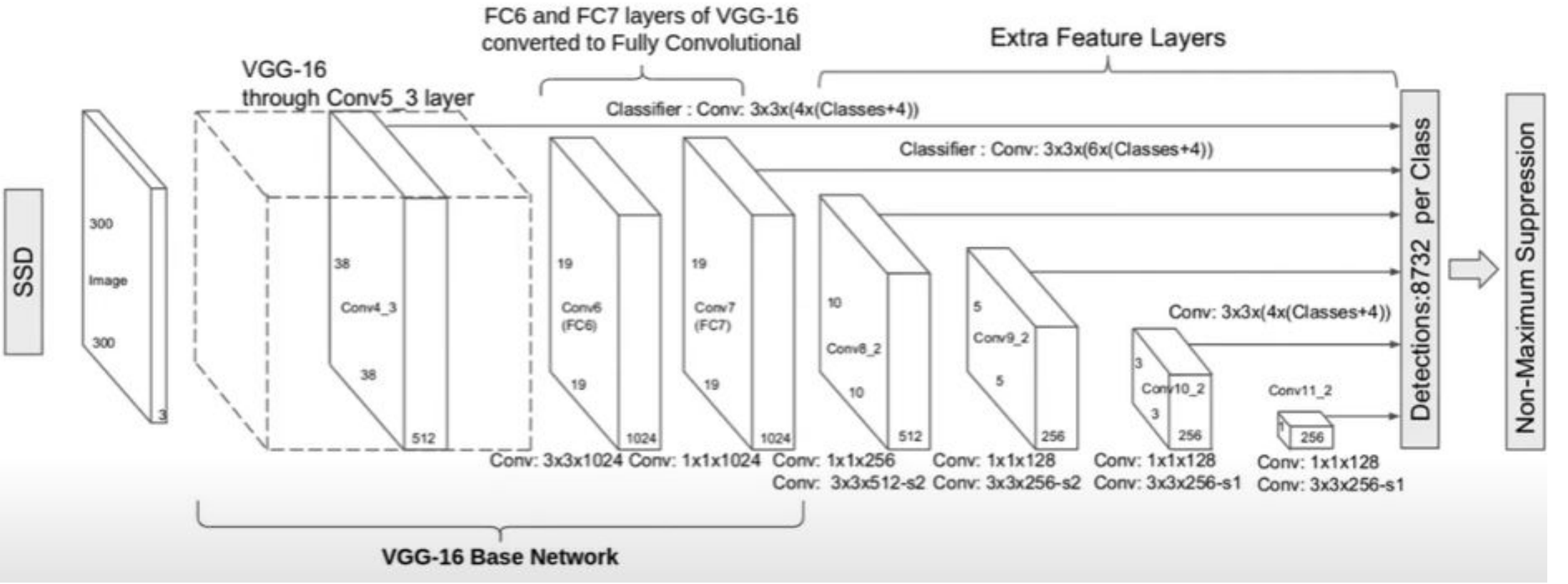


Image features
(e.g. 512 x 20 x 15)

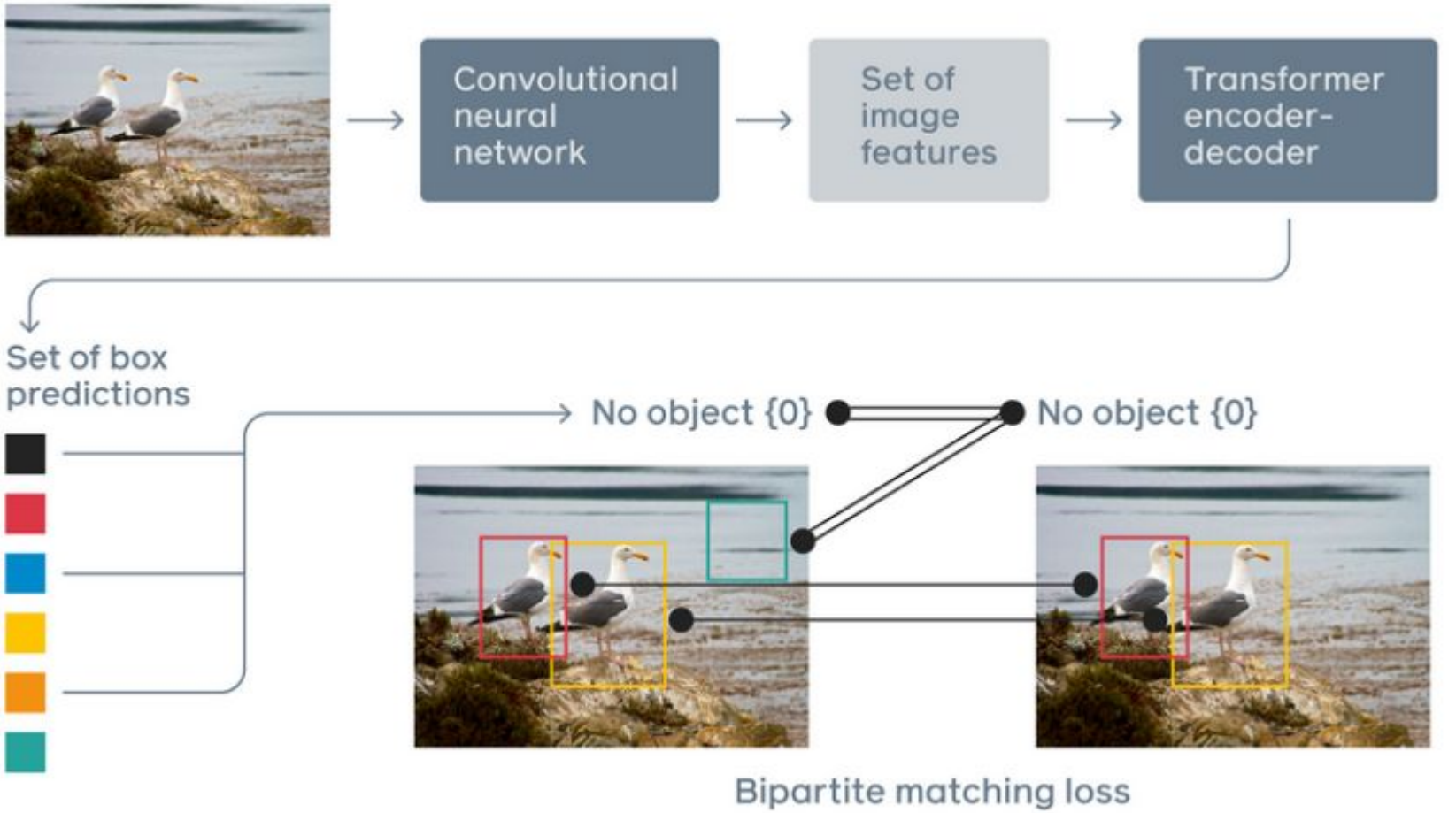
YOLO

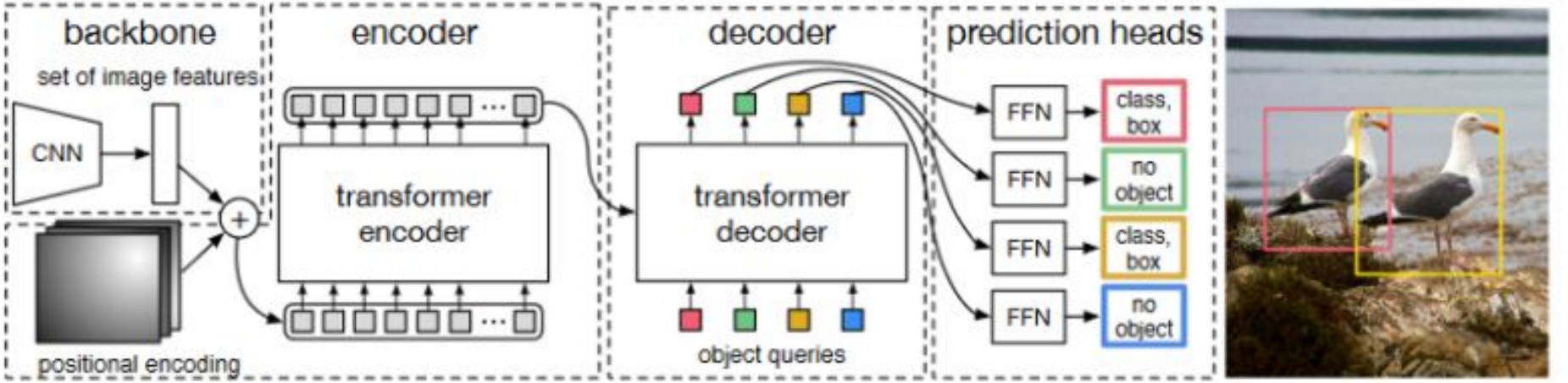

Bounding boxes + confidence
S × S grid on input
Final detections
Class probability map

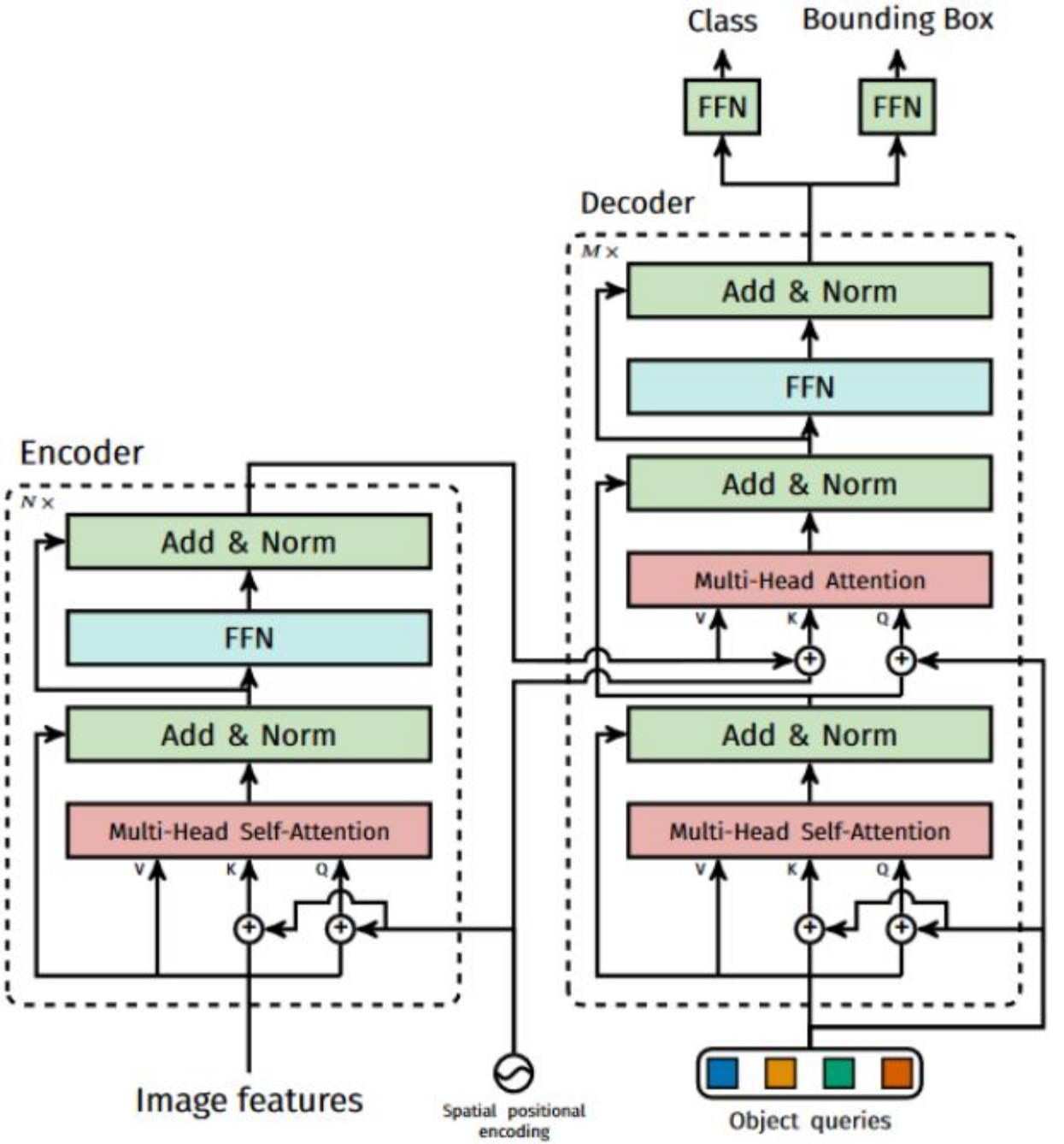
SSD


SSD
300
Image
300
3
VGG-16
through Conv5_3 layer
38
Conv4_3
38
512
FC6 and FC7 layers of VGG-16
converted to Fully Convolutional
19
Conv6
(FC6)
19
1024
19
Conv7
(FC7)
19
1024
Extra Feature Layers
Classifier : Conv: 3x3x(4x(Classes+4))
Classifier : Conv: 3x3x(6x(Classes+4))
10
Conv8_2
10
512
5
Conv9_2
5
256
Conv: 3x3x(4x(Classes+4))
3
Conv10_2
3
256
Conv11_2
1
256
Detections:8732 per Class
Non-Maximum Suppression
Conv: 3x3x1024 Conv: 1x1x1024 Conv: 1x1x256
Conv: 3x3x512-s2
Conv: 1x1x128
Conv: 3x3x256-s2
Conv: 1x1x128
Conv: 3x3x256-s1
Conv: 1x1x128
Conv: 3x3x256-s1
VGG-16 Base Network

DETR (DEtection TRansformer)




backbone
set of image features
CNN
positional encoding
encoder
transformer encoder
decoder
transformer decoder
object queries
prediction heads
FFN
FFN
FFN
FFN
class, box
no object
class, box
no object

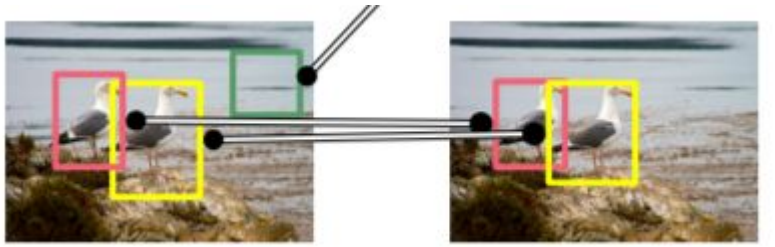

Class
Bounding Box
FFN
FFN
Decoder
M×
Add & Norm
FFN
Add & Norm
Multi-Head Attention
V
K
Q
Add & Norm
Multi-Head Self-Attention
V
K
Q
Encoder
N×
Add & Norm
FFN
Add & Norm
Multi-Head Self-Attention
V
K
Q
Image features
Spatial positional encoding
Object queries

Loss function

- Calculate the best match of predictions with respect to given ground truths using a graph technique with a cost function

$$\hat{\sigma} = \arg\min_{\sigma \in \mathfrak{S}_N} \sum_{i}^{N} \mathcal{L}_{\text{match}}(y_i, \hat{y}_{\sigma(i)}),$$

$$\mathcal{L}_{\text{Hungarian}}(y, \hat{y}) = \sum_{i=1}^{N} \left[-\log \hat{p}_{\hat{\sigma}(i)}(c_i) + \mathbb{1}_{\{c_i \neq \varnothing\}} \mathcal{L}_{\text{box}}(b_i, \hat{b}_{\hat{\sigma}}(i)) \right]$$



THANK YOU