
DIVE INTO
VISION
TRANSFORMERS
Tianyi

Self-attention

https://arxiv.org/abs/1706.03762

q: query (to match others)

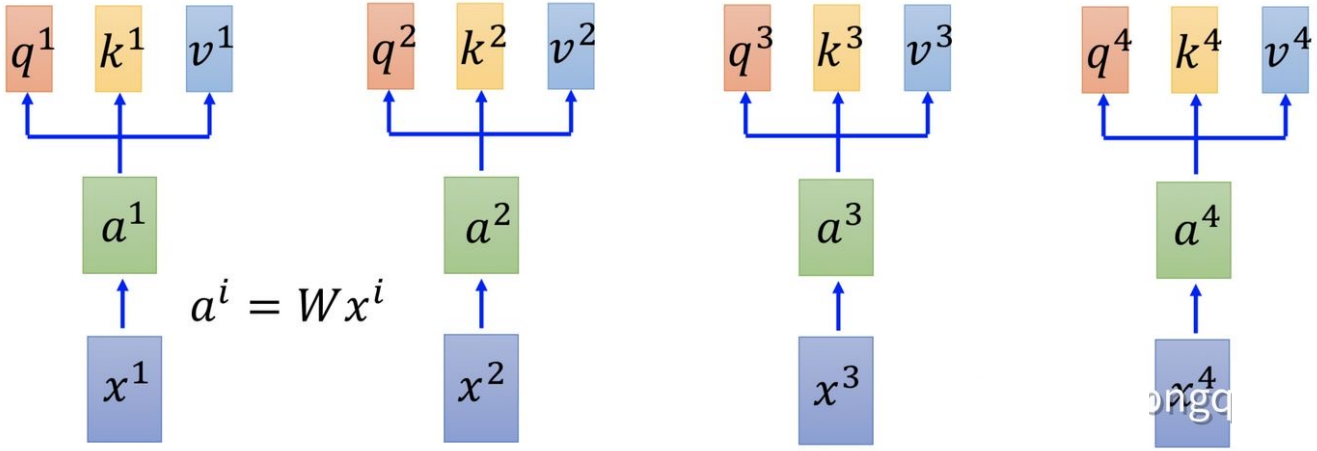
$$q^i = W^q a^i$$

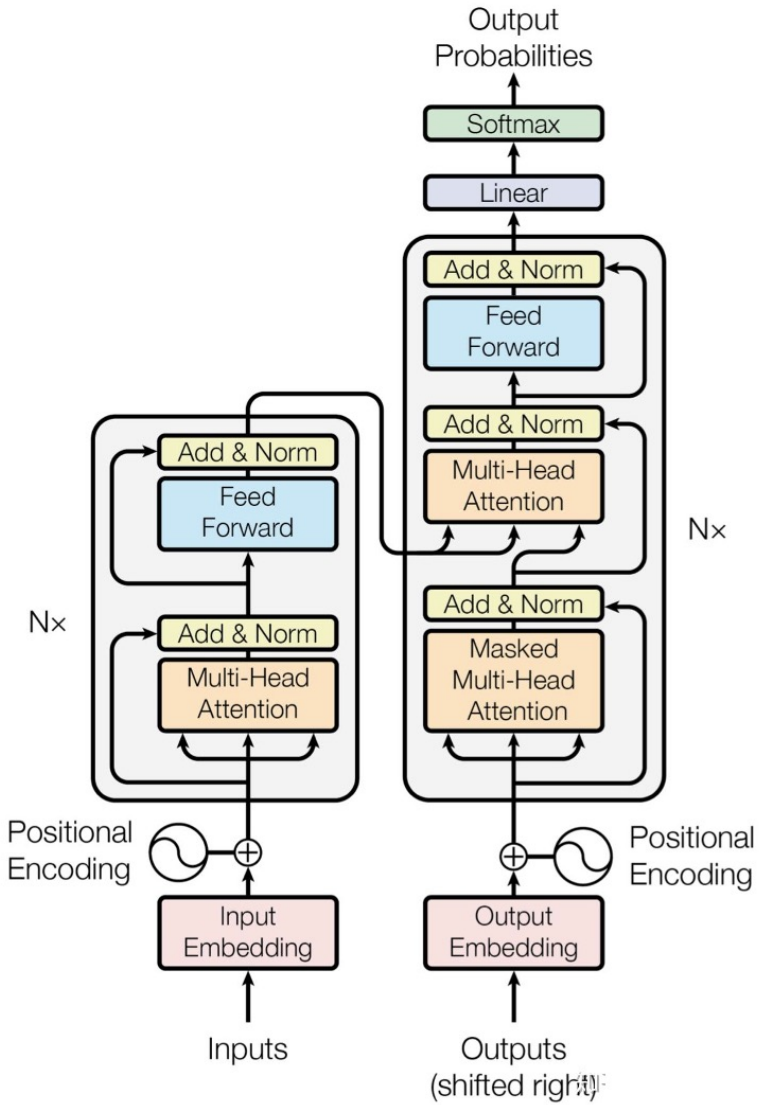
k: key (to be matched)

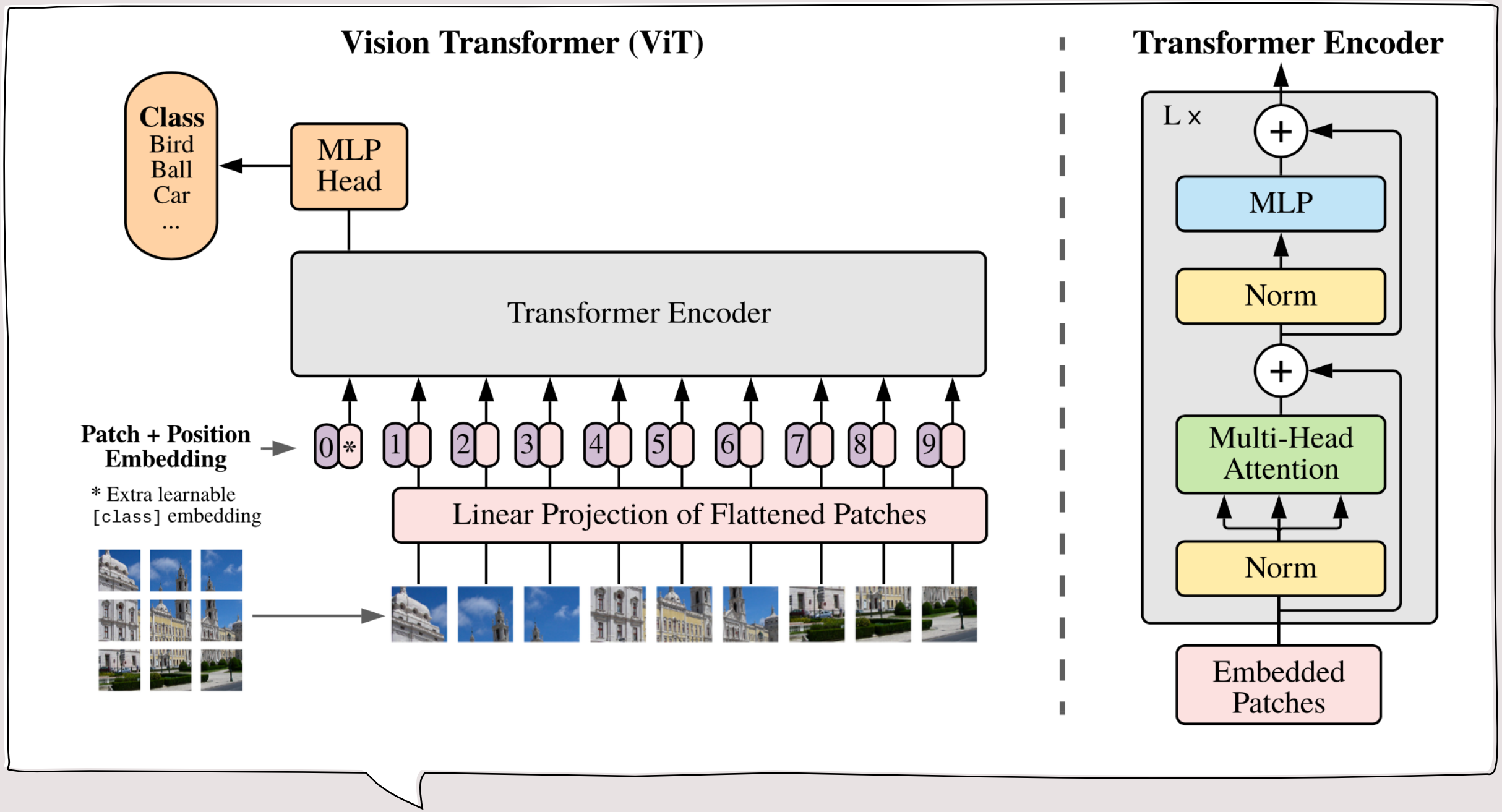
$$k^i = W^k a^i$$

v: information to be extracted

$$v^i = W^v a^i$$

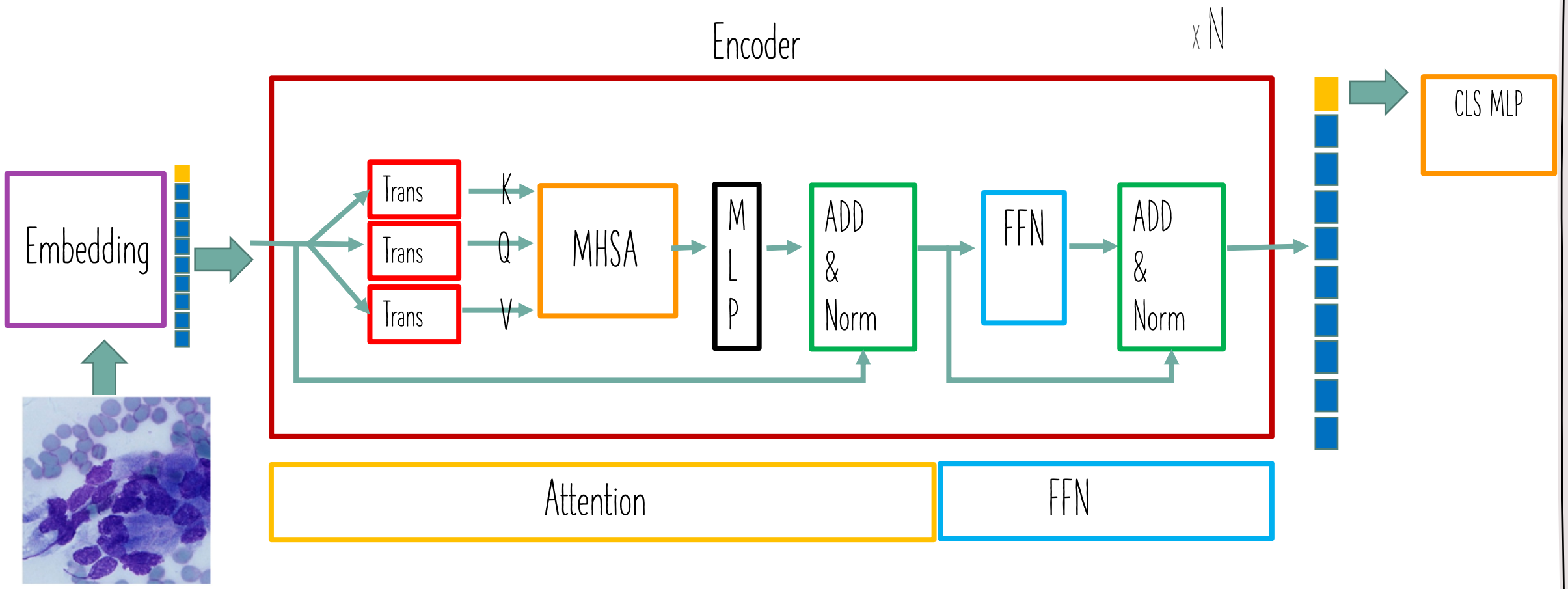





Vision Transformer (ViT)
Class
Bird
Ball
Car
...
MLP
Head
Transformer Encoder
Patch + Position
Embedding
* Extra learnable
[class] embedding
0*
1
2
3
4
5
6
7
8
9
Linear Projection of Flattened Patches
Transformer Encoder
L x
MLP
Norm
Multi-Head
Attention
Norm
Embedded
Patches

ViT Vision Transformer

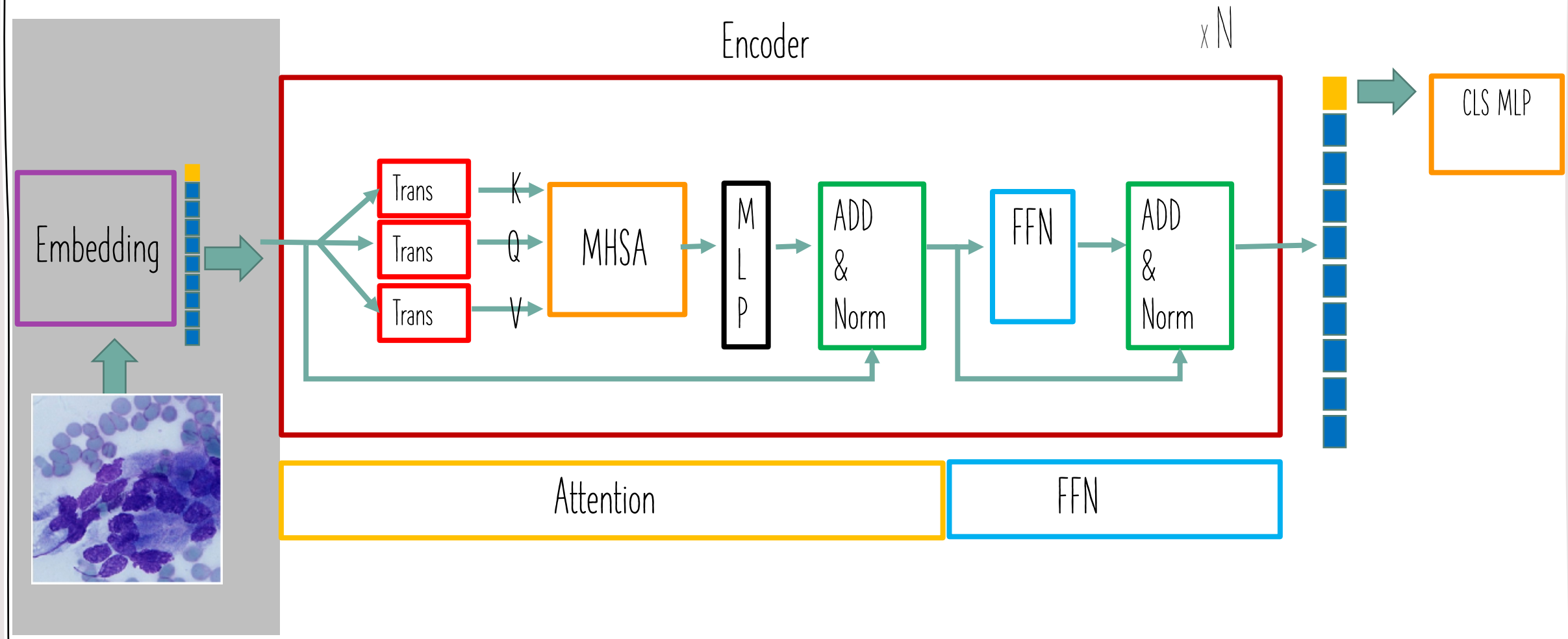
AN IMAGE IS WORTH 16X16 WORDS: TRANSFORMERS FOR IMAGE RECOGNITION AT SCALE
Arxiv 2010.11929



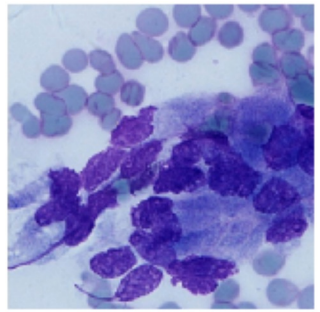

ConvNet
ViT
MLP Mixer

ViT Vision Transformer

AN IMAGE IS WORTH 16X16 WORDS: TRANSFORMERS FOR IMAGE RECOGNITION AT SCALE
Arxiv 2010.11929



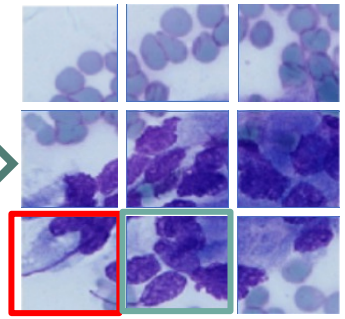
Patch embedding



(Batch, 3, 224, 224)

CNN projection

Use CNN to split patches



(Batch ,
Embed_dim,
Patch_num_col,
Patch_num_row)

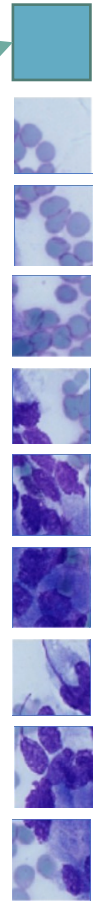
Kernal size = Stride
Input Chanel = 3
Output Chanel = Embed_dim

ReArrange

(Batch,
patch_num,
Embed_dim)
(8, 196, 768)

CLS token
(1,768) Zeros

Concatenate



(Batch,
patch_num +1,
Embed_dim)
(8, 197, 768)

(patch_num +1
Embed_dim)

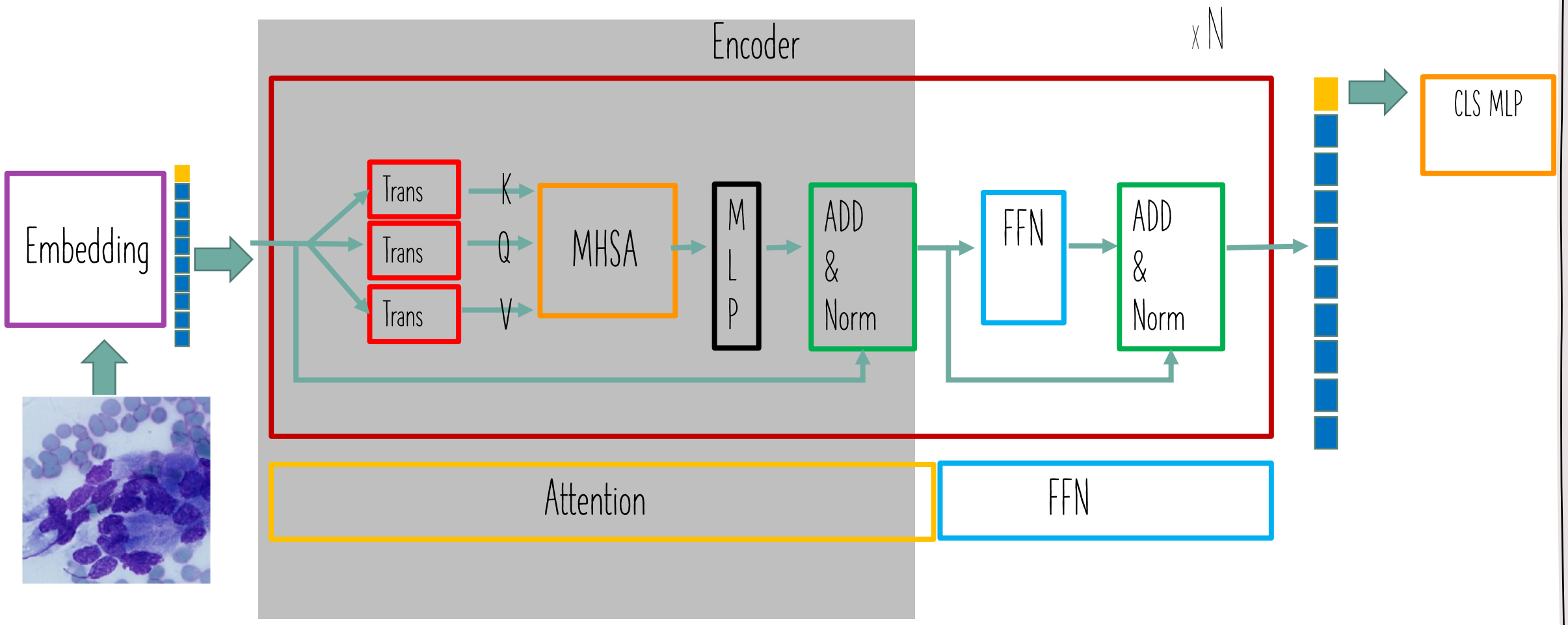
+ positional encoding

zeros :
learnable
posional
encoding

(Batch,
patch_num +1,
Embed_dim)
(8, 197, 768)

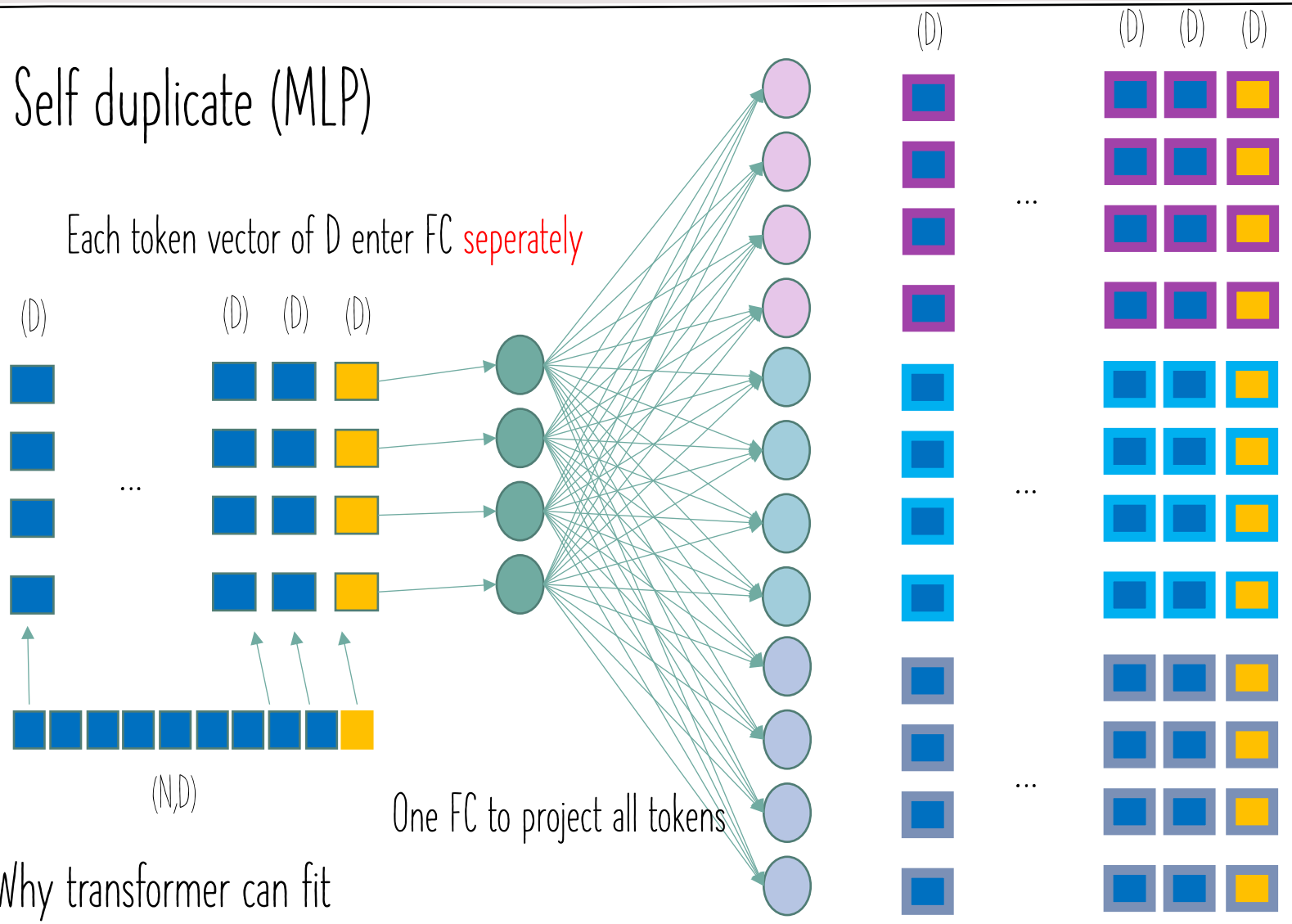
ViT Vision Transformer

AN IMAGE IS WORTH 16X16 WORDS: TRANSFORMERS FOR IMAGE RECOGNITION AT SCALE
Arxiv 2010.11929



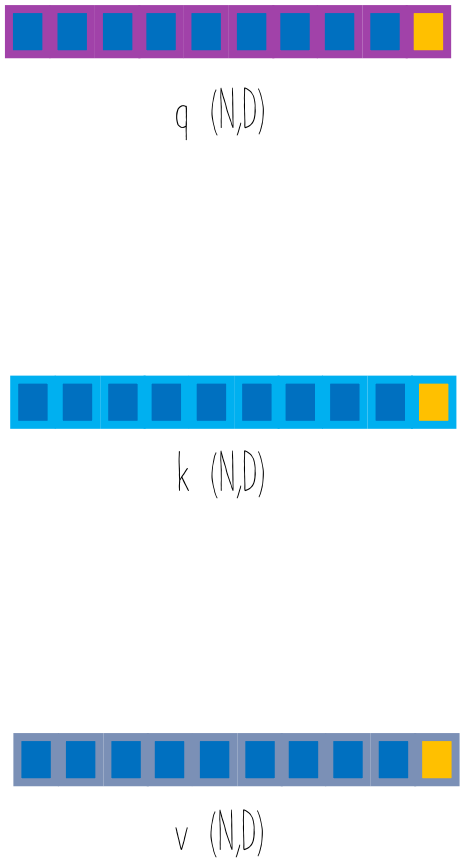
Self duplicate (MLP)

Each token vector of D enter FC seperately



Why transformer can fit different length ?

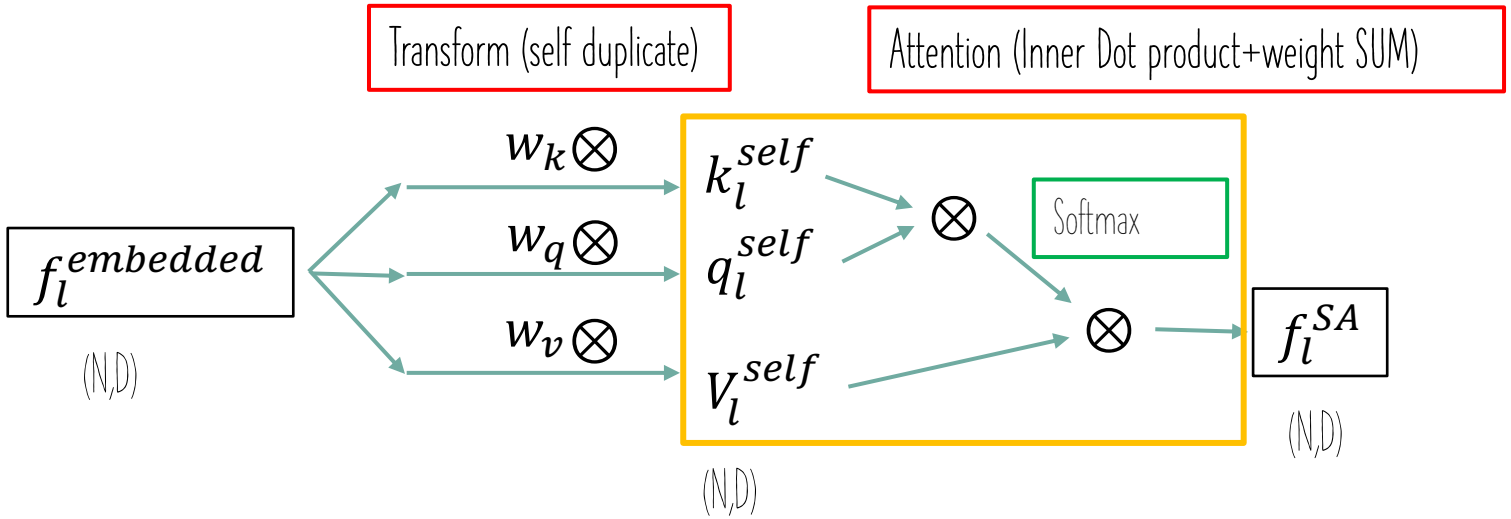
Self-duplicate projection obtain each q,k,v token togather



What is q,k,v ?

Tokens projeted to different representation latent space

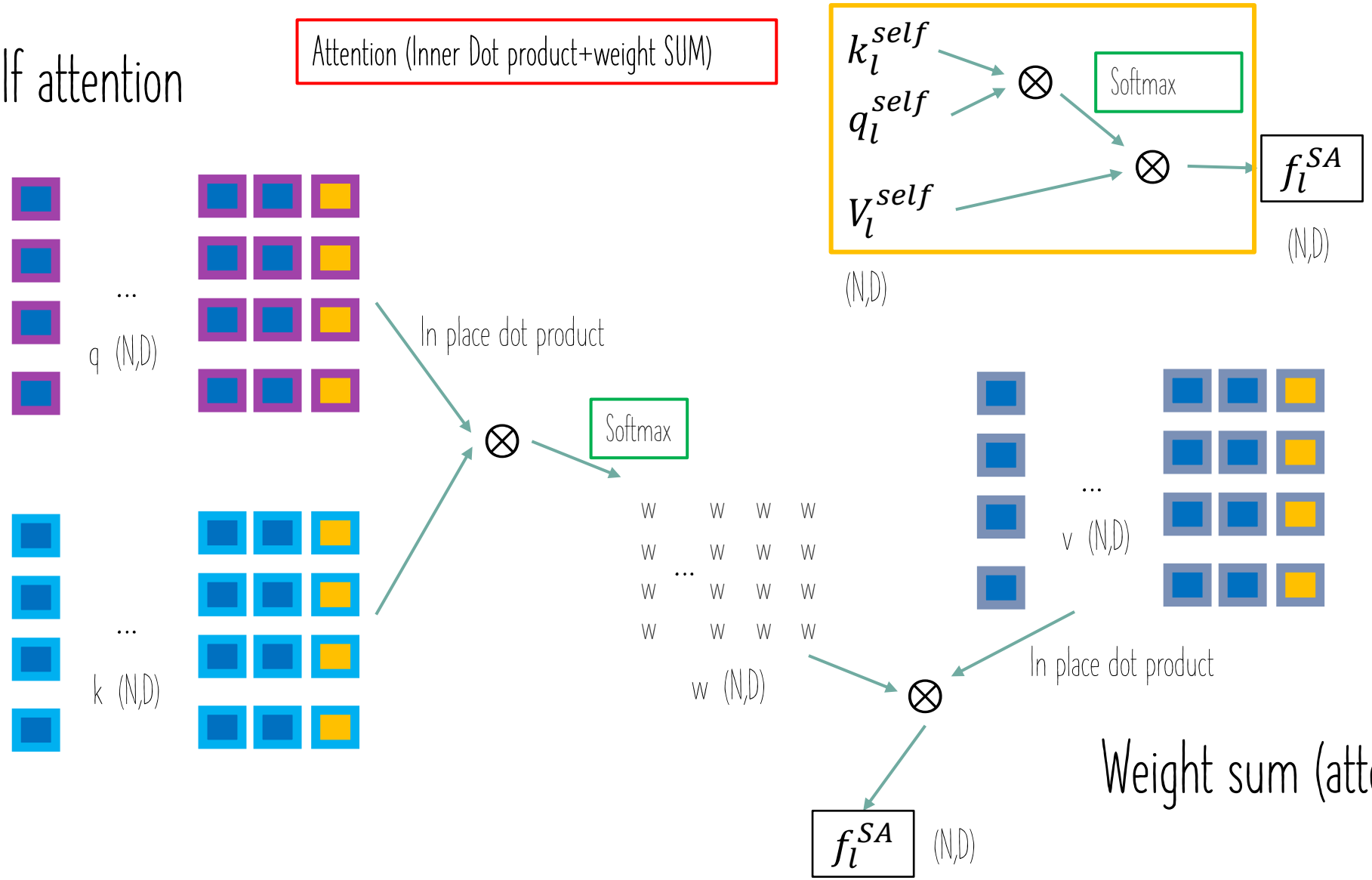
SA self attention



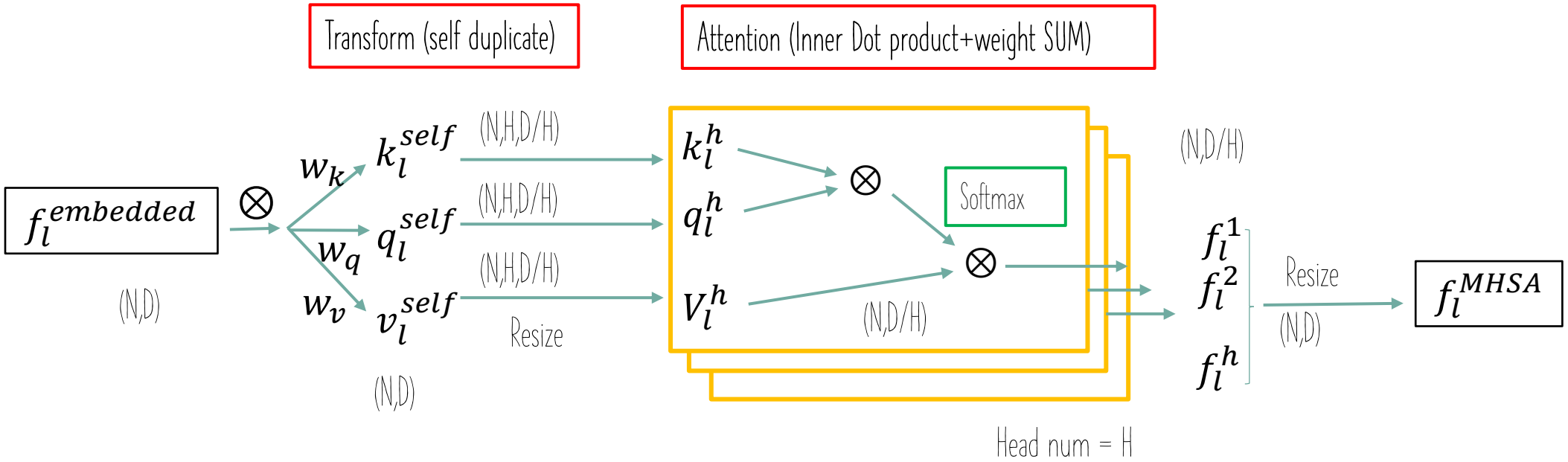
For a given embedded patch $f_l^{embedded}$, $f_l^{SA} = SoftMax(q_l^{self^T} \cdot k_l^{self})^T \cdot V_l^{self}$,
where $q_l^{self} = w_q \cdot f_l^{embedded}$, $k_l^{self} = w_k \cdot f_l^{embedded}$, $V_l^{self} = w_v \cdot f_l^{embedded}$.

SA self attention

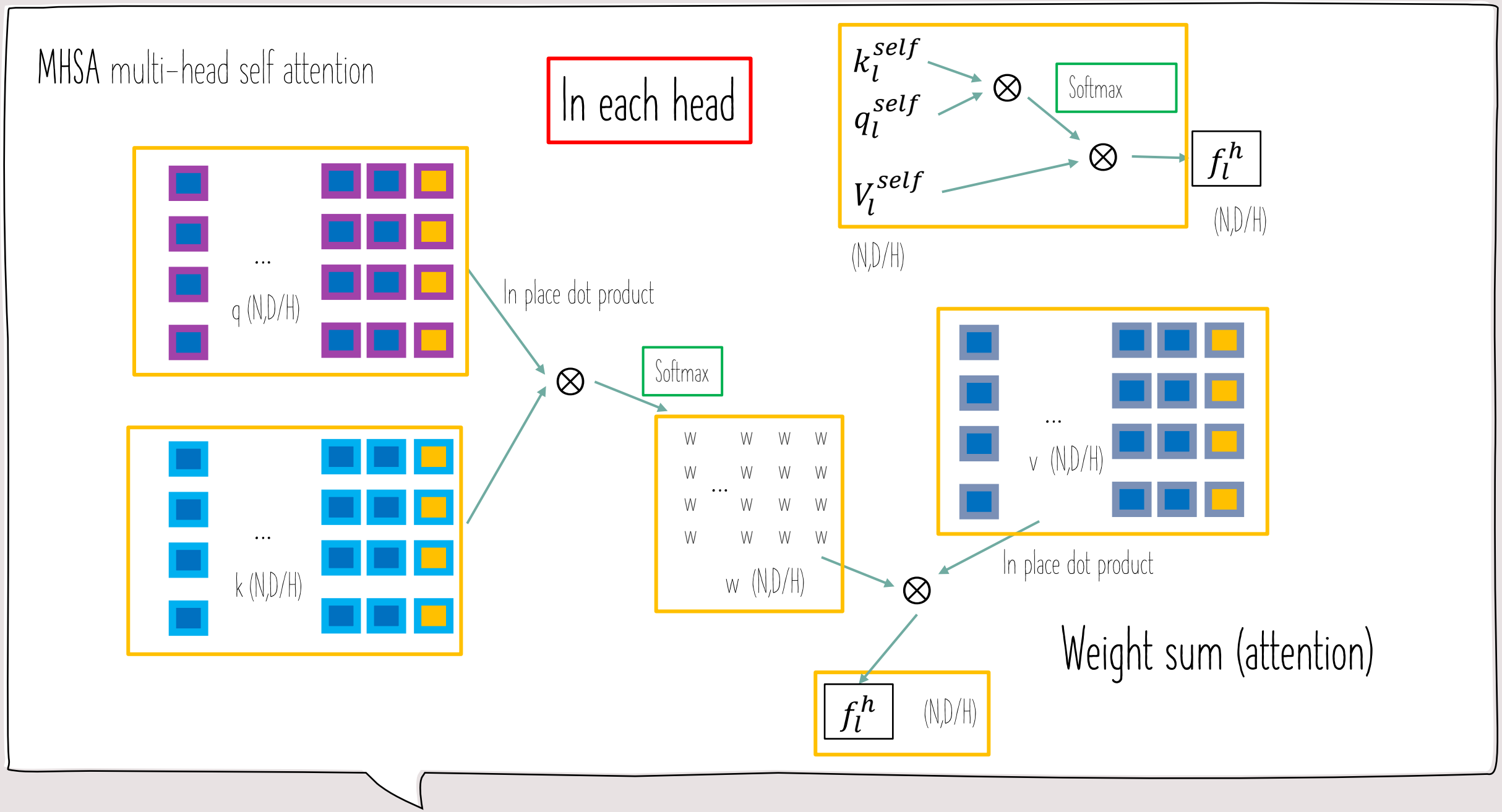
Attention (Inner Dot product+weight SUM)



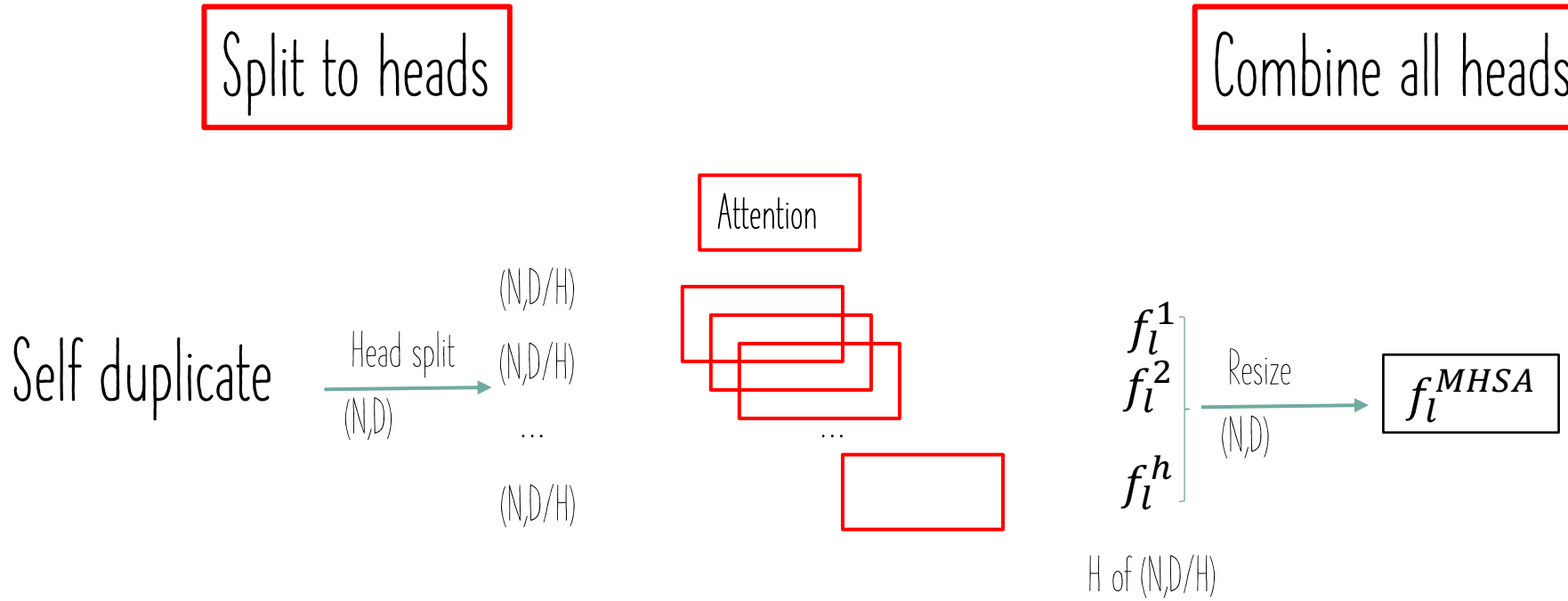
MHSA multi-head self attention



For a given embedded patch in each head $f_l^{embedded}$, $f_l^{MHSA} = SoftMax(q_l^{self^T} \cdot k_l^{self})^T \cdot V_l^{self}$,
where $q_l^{self} = w_q \cdot f_l^{embedded}$, $k_l^{self} = w_k \cdot f_l^{embedded}$, $V_l^{self} = w_v \cdot f_l^{embedded}$.

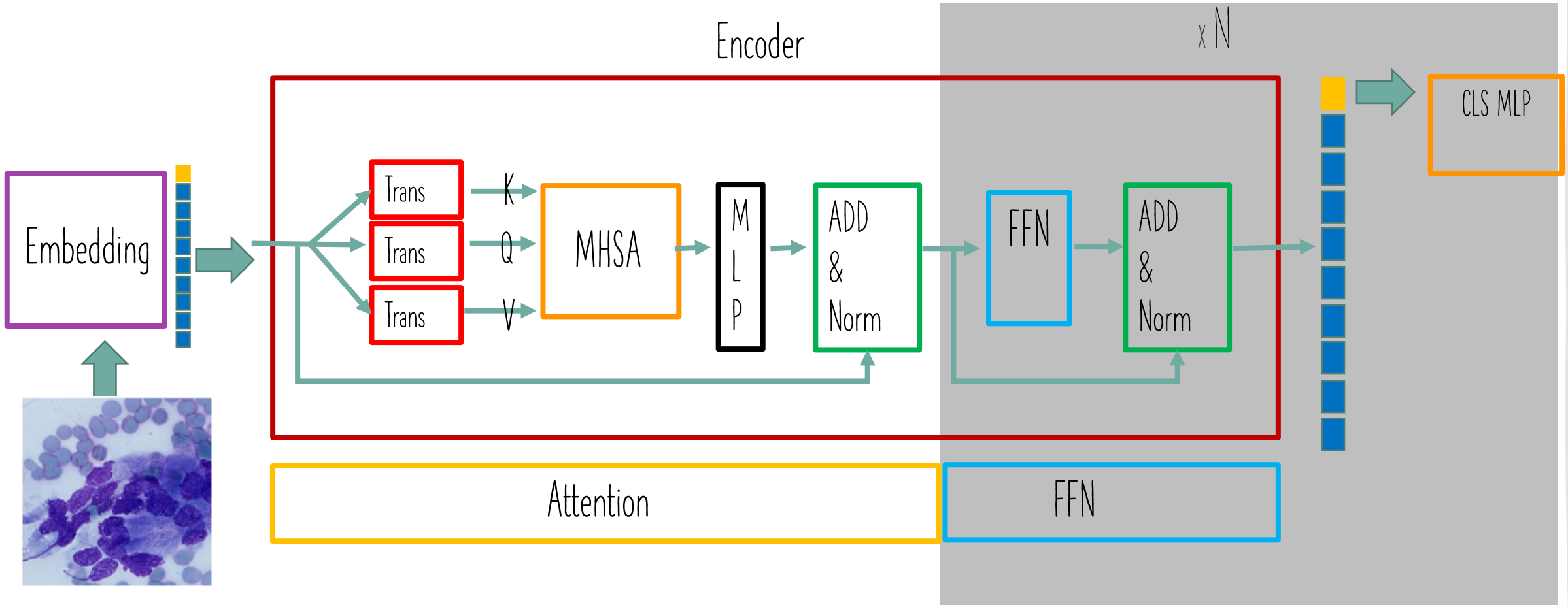

MHSA multi-head self attention
In each head
k_l^{self}
q_l^{self}
V_l^{self}
Softmax
f_l^h
(N,D/H)
(N,D/H)
In place dot product
q (N,D/H)
k (N,D/H)
Softmax
w (N,D/H)
v (N,D/H)
In place dot product
Weight sum (attention)
f_l^h
(N,D/H)

MHSA multi-head self attention



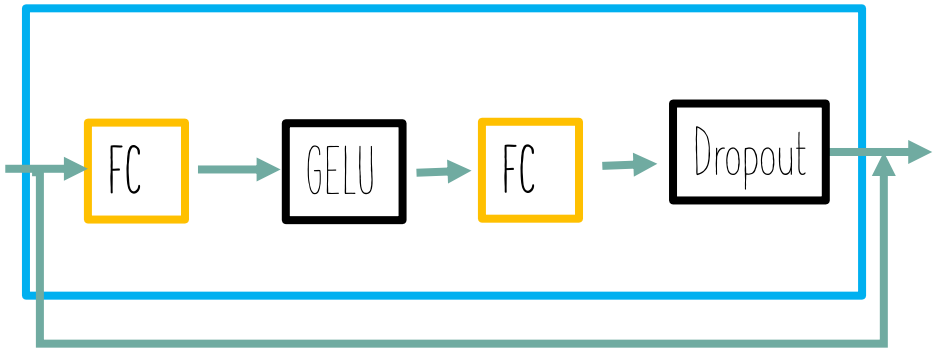
ViT Vision Transformer

AN IMAGE IS WORTH 16X16 WORDS: TRANSFORMERS FOR IMAGE RECOGNITION AT SCALE
Arxiv 2010.11929



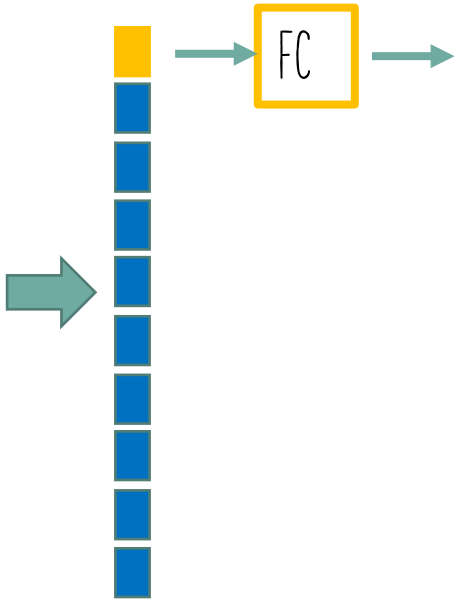
FFN

(Batch,
patch_num,
Embed_dim)
(8, 196, 768)



(Batch,
patch_num,
Embed_dim)
(8, 196, 768)

CLS MLP

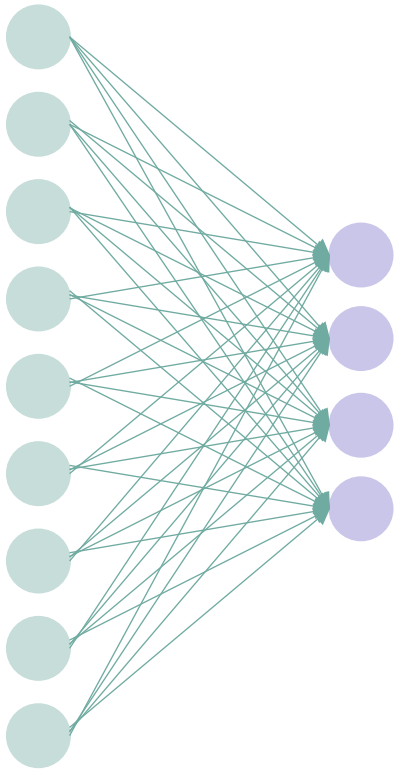


Confidence on each class

(B, CLS numbers)

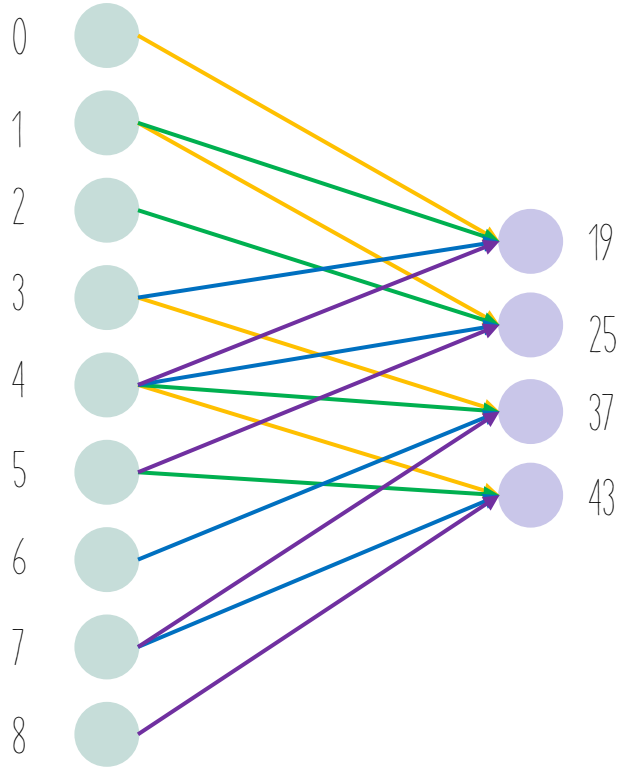

The projection issue

What is the neurons of CNN looks like?

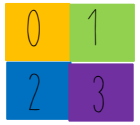


9 * 4 = 36

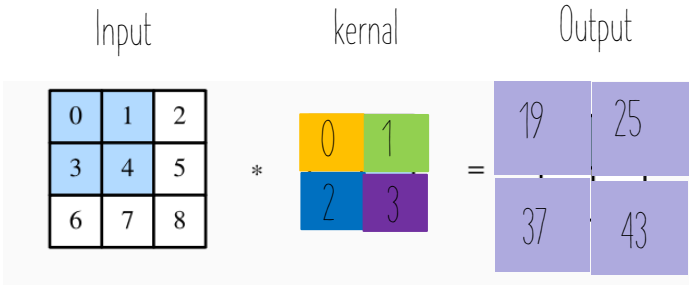
$$F(X) = \sigma(A\,X + B)$$



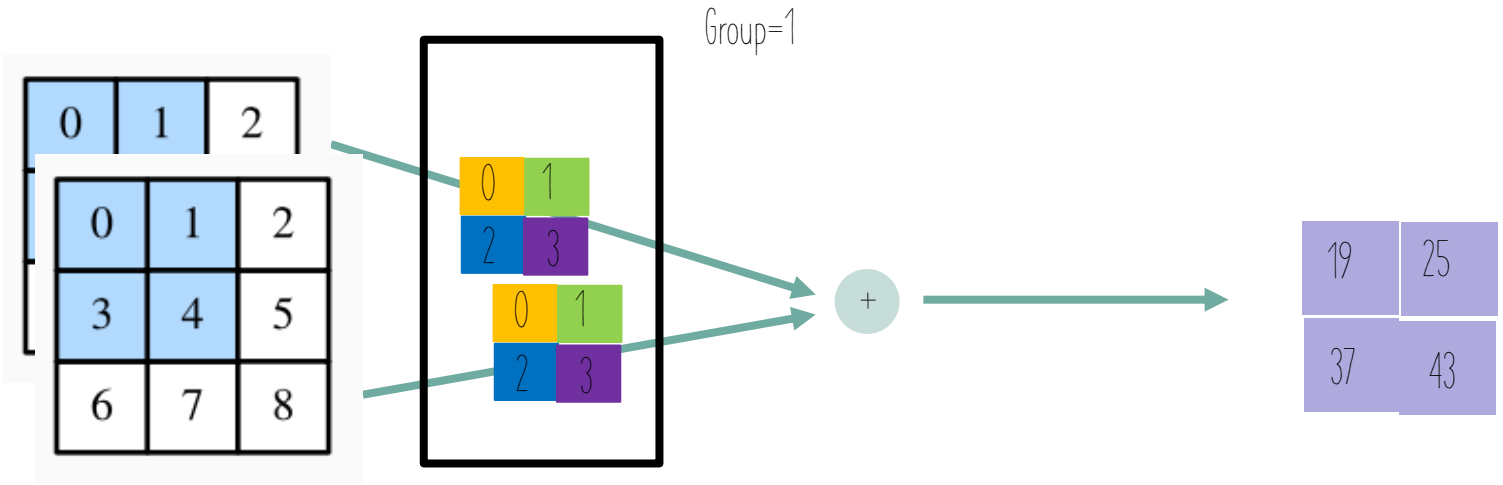
4 * 4 = 16



The number of real parameters is only 4



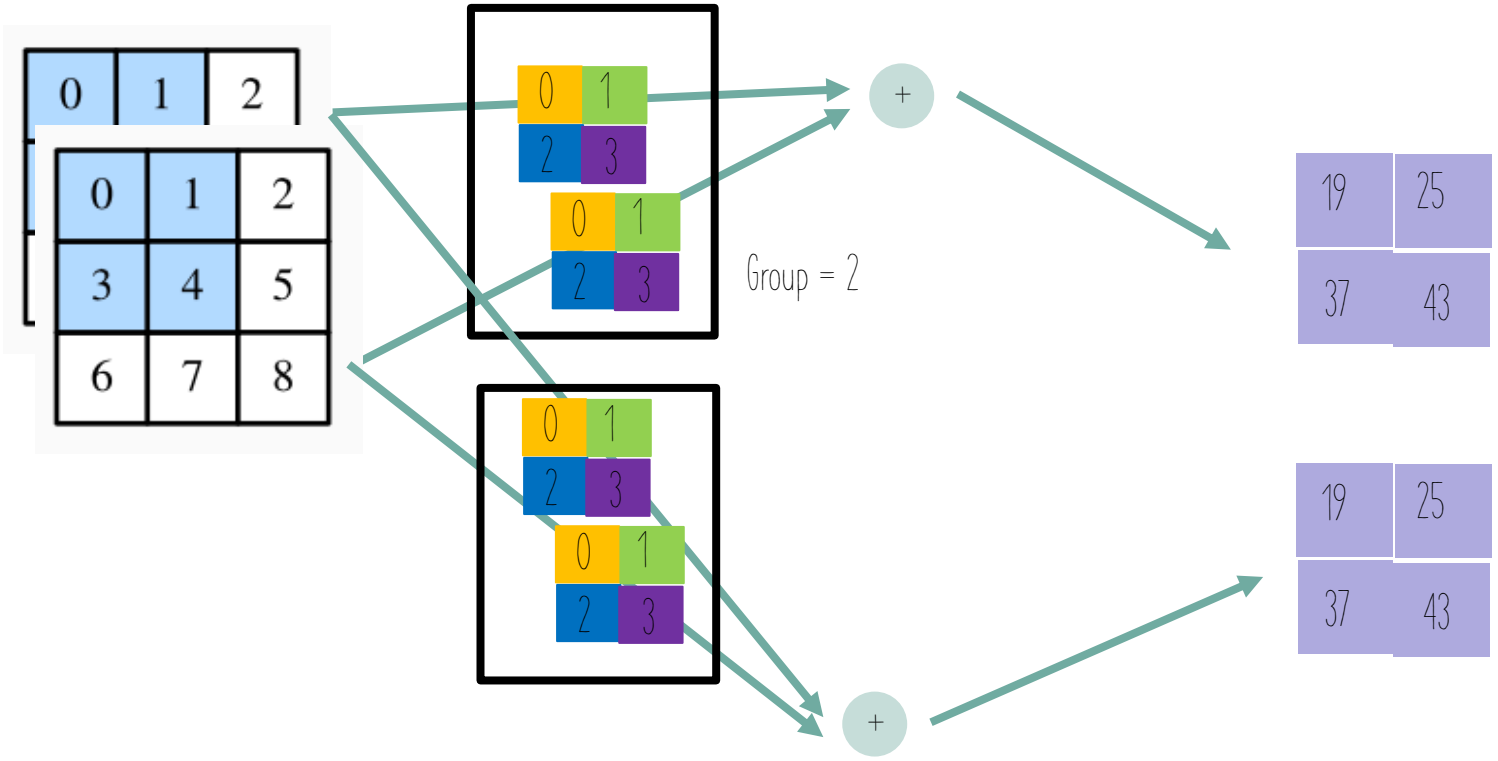
What is CNN's channel?



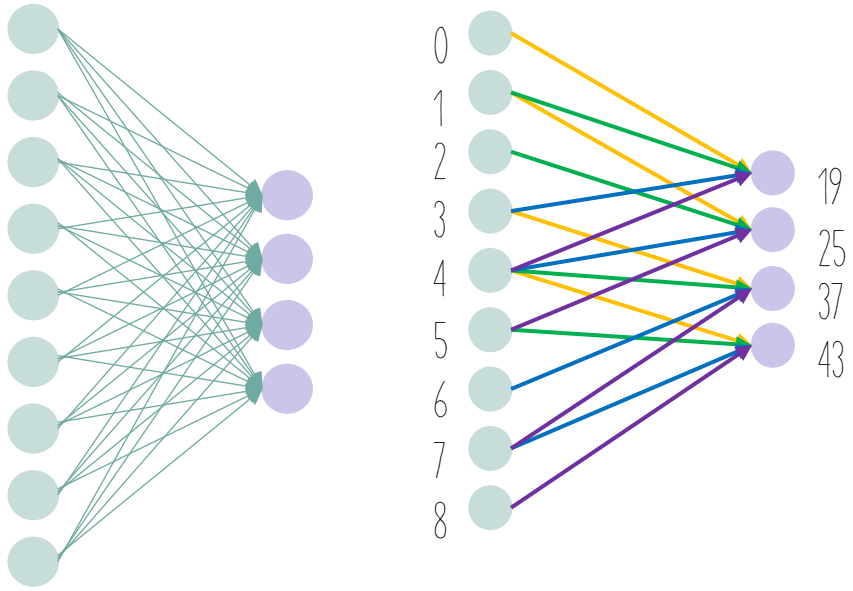
Input channel = Kernal channel

Output channel = Kernal Group

What is CNN's channel?



```python
import torch
import torch.nn as nn

# FC
layer = nn.Linear(9, 4)
img = torch.randn(1, 9)

preds = layer(img)  # (1, class_number)
print('test model output:  ', preds.shape)

img = torch.randn(1, 10)
preds = layer(img)  # (1, class_number)
print('test model output:  ', preds.shape)
# RuntimeError: mat1 and mat2 shapes cannot be multiplied (1x10 and 9x4)
```



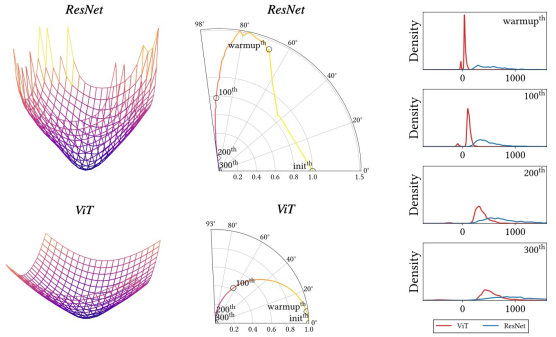

```python
import torch
import torch.nn as nn

layer = nn.Conv2d(3, 768, kernel_size=(16,16),
stride=(16,16))

edge_size = 384
img = torch.randn(1, 3, edge_size, edge_size)
preds = layer(img)  # (1, class_number)
print('test model output:  ', preds.shape)

edge_size = 224
img = torch.randn(1, 3, edge_size, edge_size)
preds = layer(img)  # (1, class_number)
print('test model output:  ', preds.shape)
```

POINTS TO DECLEAR

1. Who's learning the most of the hard works ?
2. Who's the heavyest ?
3. Why its soooo stirring ?
4. A slight view of the future ? (structure, strategy, ...)

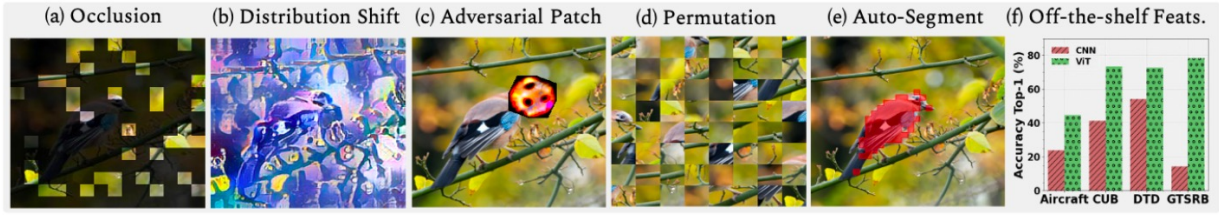
Hard Math

HOW DO VISION TRANSFORMERS WORK?
Arxiv 2202.06709



Inspiring Experiments

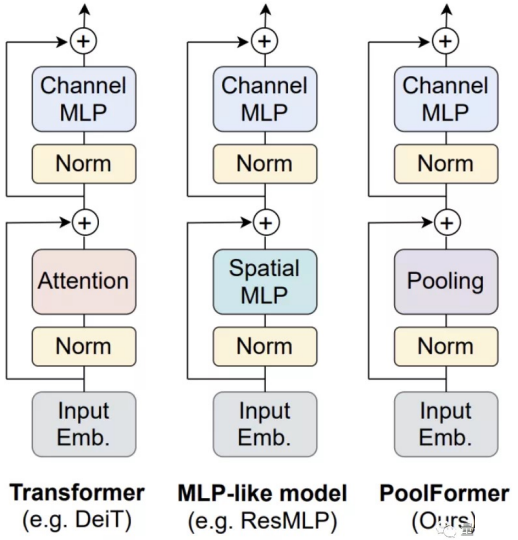
NeurPS2021 Intriguing Properties of Vision Transformers
Arxiv 2105.10497



Inspiring Model play

MetaFormer is Actually What You Need for Vision
Poolformer Arxiv 2111.11418

Patches Are All You Need
https://openreview.net/pdf?id=TVHS5Y4dNvM



Under review as a conference paper at ICLR 2022

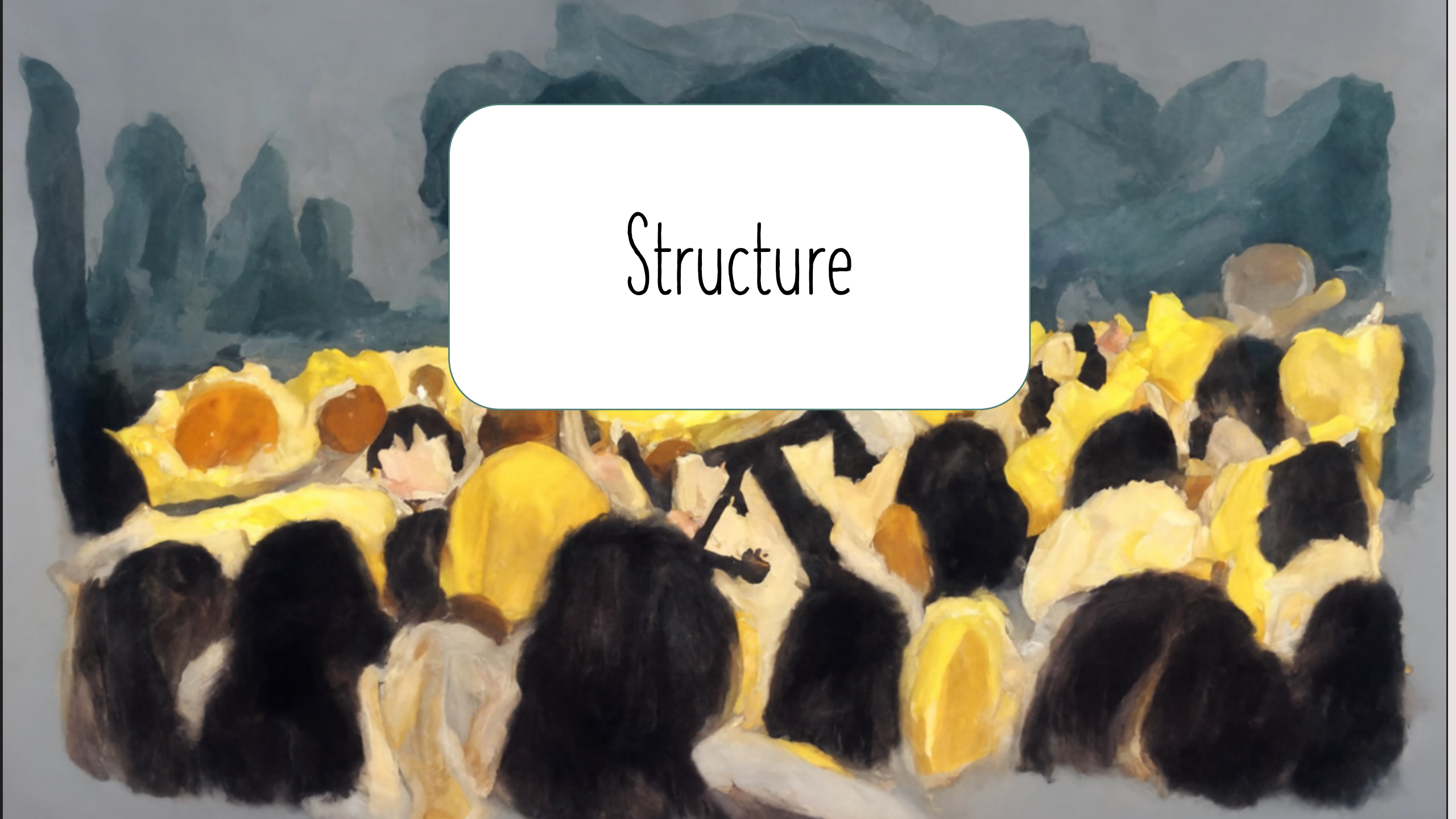
~~CONVOLUTIONS~~ ~~ATTENTION~~ ~~MLPS~~
PATCHES ARE ALL YOU NEED? 🤷

Anonymous authors
Paper under double-blind review

Model structure

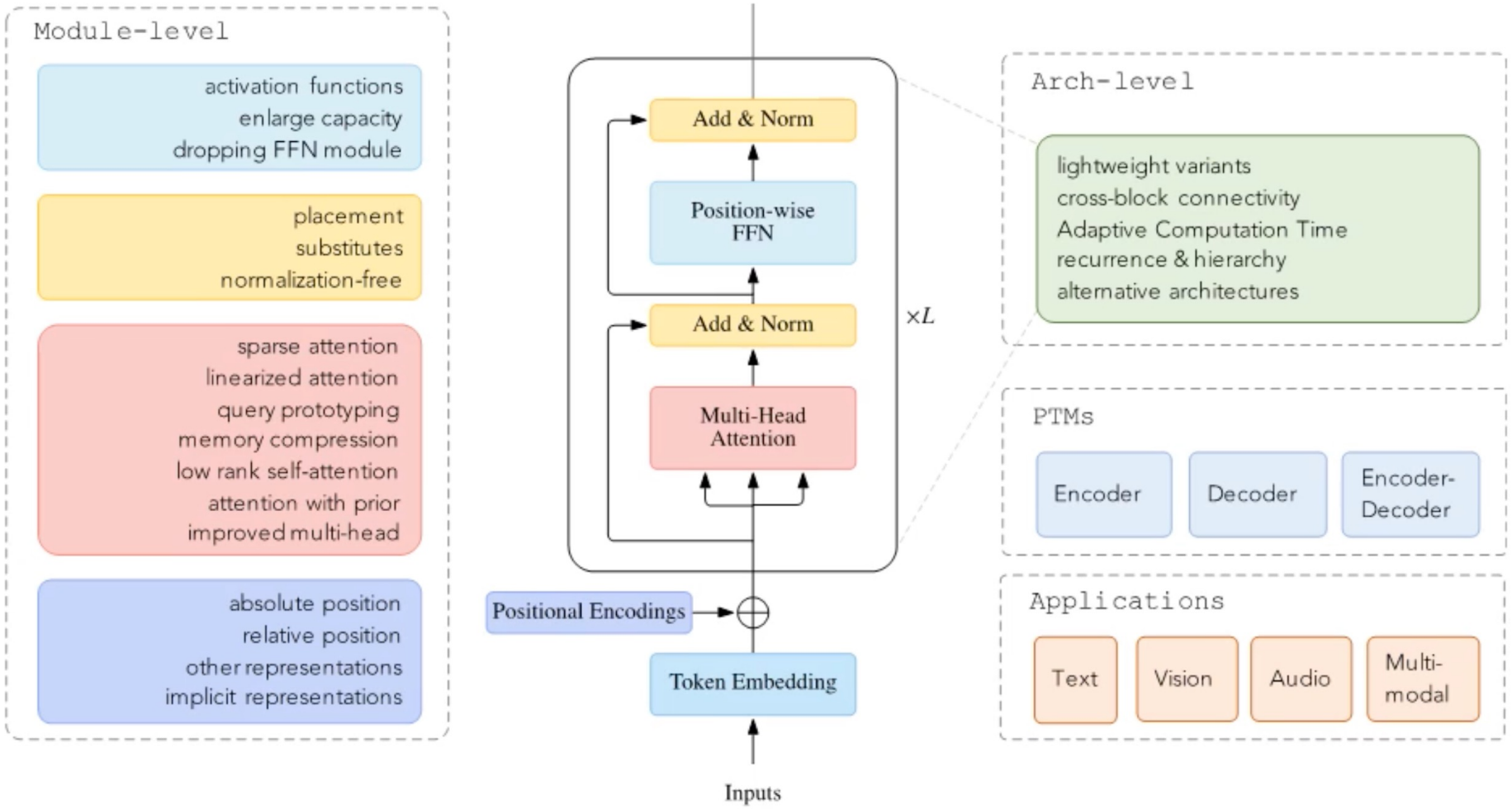
Learning strategy

Initiallization


Structure

A taxonomy of X-formers (Transformer variants)

arXiv 2106.04554



MSHT

arXiv:2112.13513

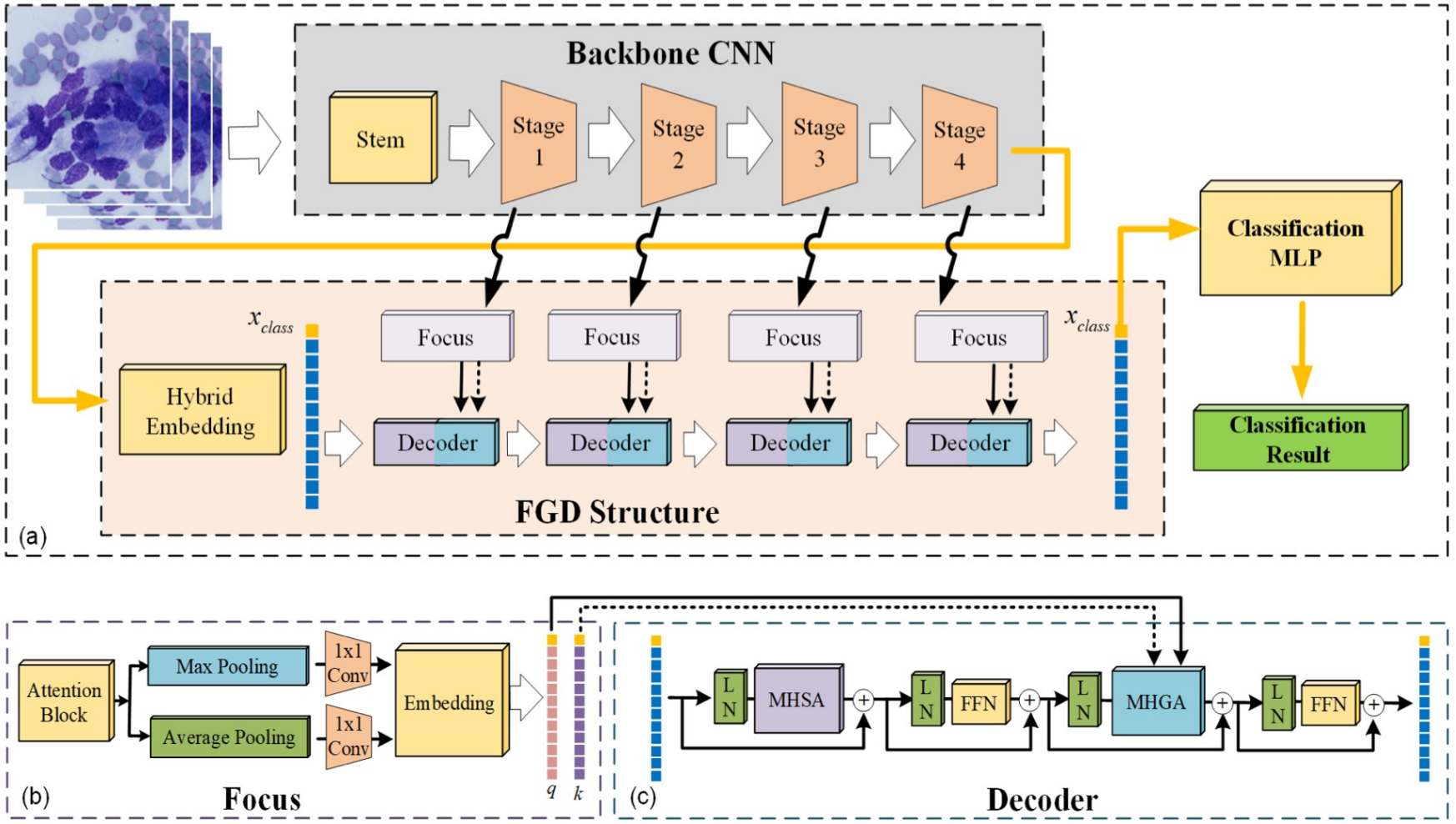
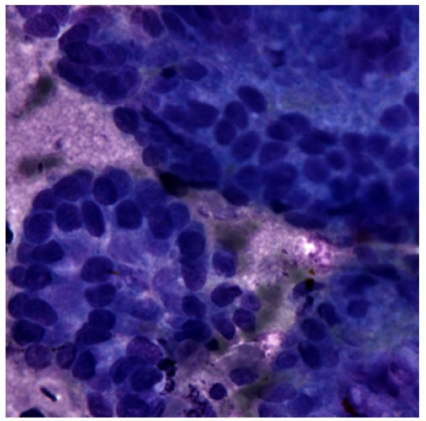


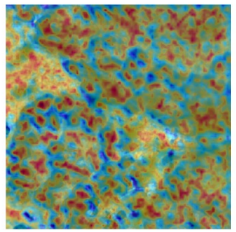
Fig. 1. The architecture of the proposed Multi-stage Hybrid Transformer (MSHT) model for the classification of ROSE images of pancreatic cancer. (a) The architecture of MSHT. (b) The focus block of the FGD structure (c) The decoder of the FGD structure. MHSA denotes multi-head self-attention, MHGA denotes multi-head guided-attention, LN denotes layer norm block, FFN denotes the feed-forward network, and MLP denotes multi-layer perceptron.

Tianyi Zhang, Yunlu Feng, Yu Zhao, Guangda Fan, Aiming Yang, Shangqin Lyu, Peng Zhang, Fan Song, Chenbin Ma, Yangyang Sun, Youdan Feng, and Guanglei Zhang*, "MSHT: Multi-stage hybrid transformer for the ROSE image analysis of pancreatic cancer arXiv:2112.13513 . (JBHI under review)

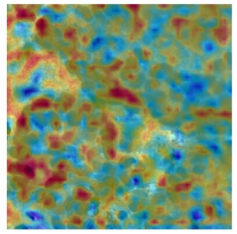
Original Image
Negative



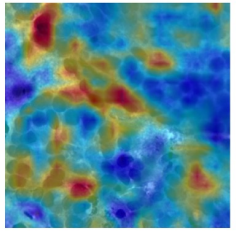
Encoder 1



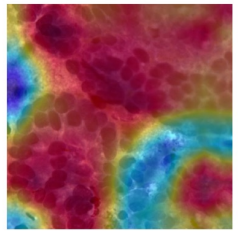
Encoder 2



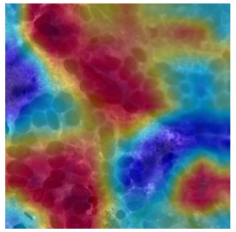
Encoder 3



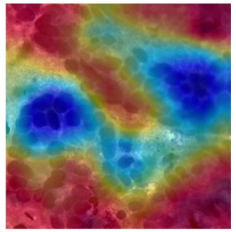
Encoder 4



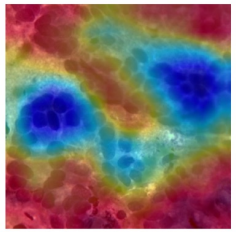
Decoder 1



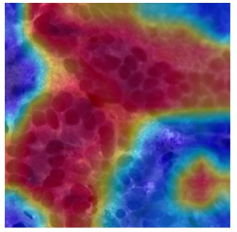
Decoder 2



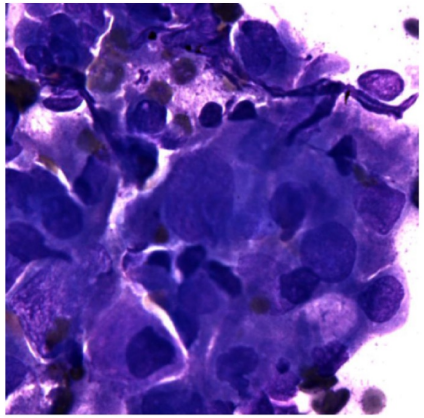
Decoder 3



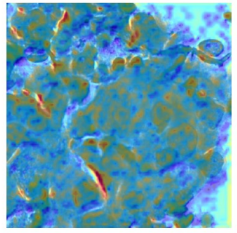
Decoder 4



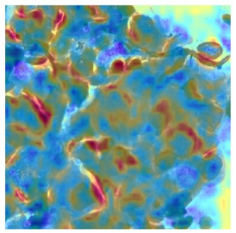
Original Image
Positive



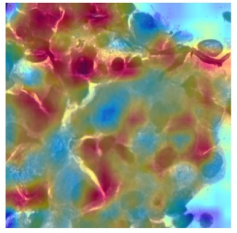
Encoder 1



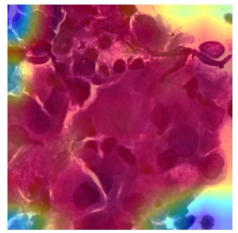
Encoder 2



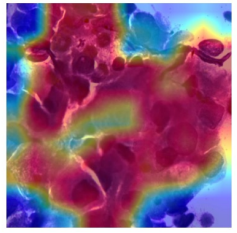
Encoder 3



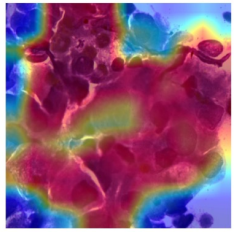
Encoder 4



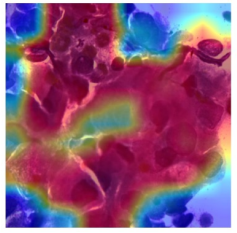
Decoder 1



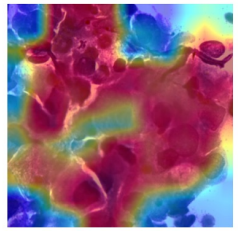
Decoder 2



Decoder 3

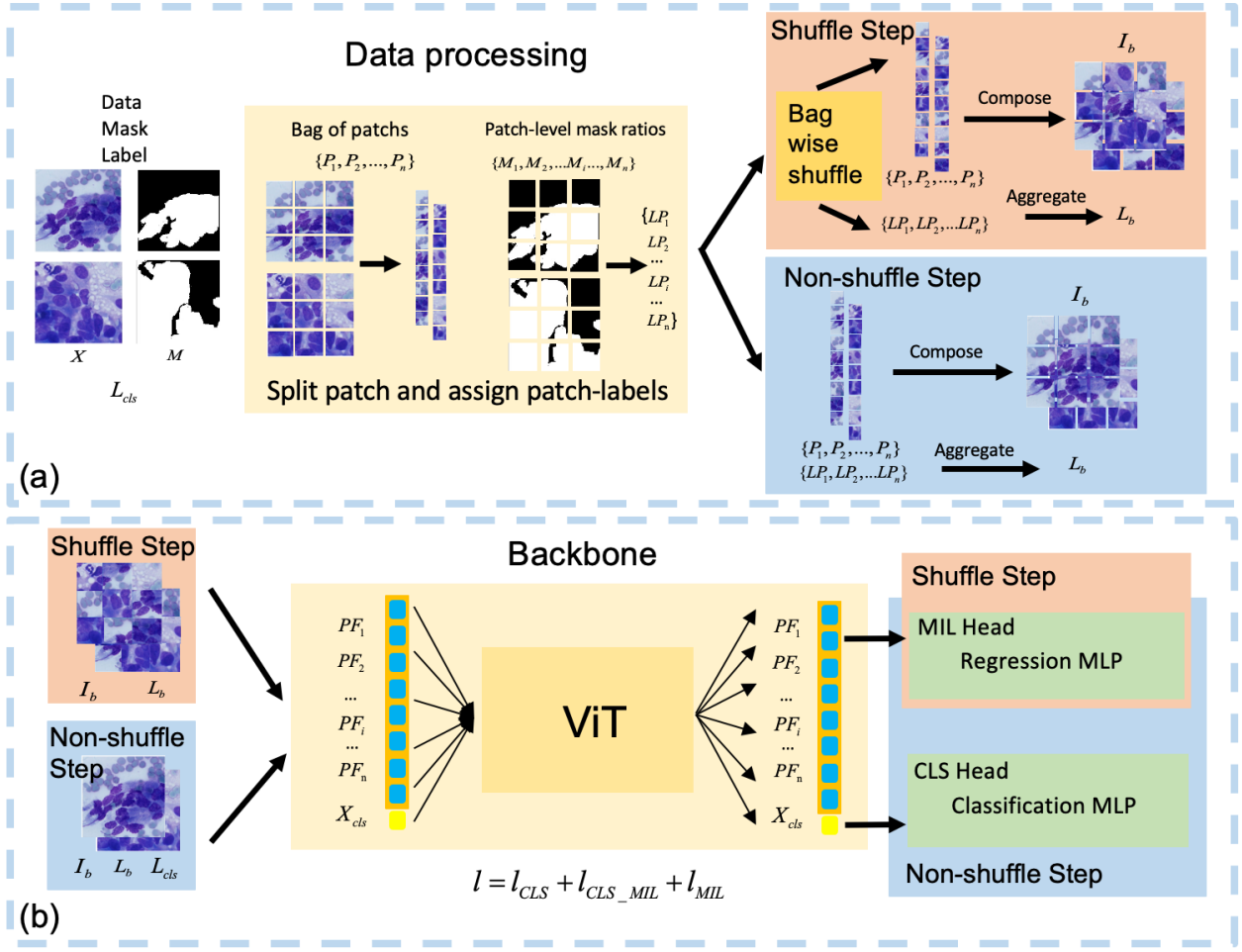


Decoder 4




Strategy

SI-ViT/MIL-SI



Key Points

1. Fast and with only limited alterations
2. Instance re-grouping strategy
3. A soft-label with background
4. Alternative training strategy to balance CLS and MIL tasks
5. Two head design
6. Simple Loss design

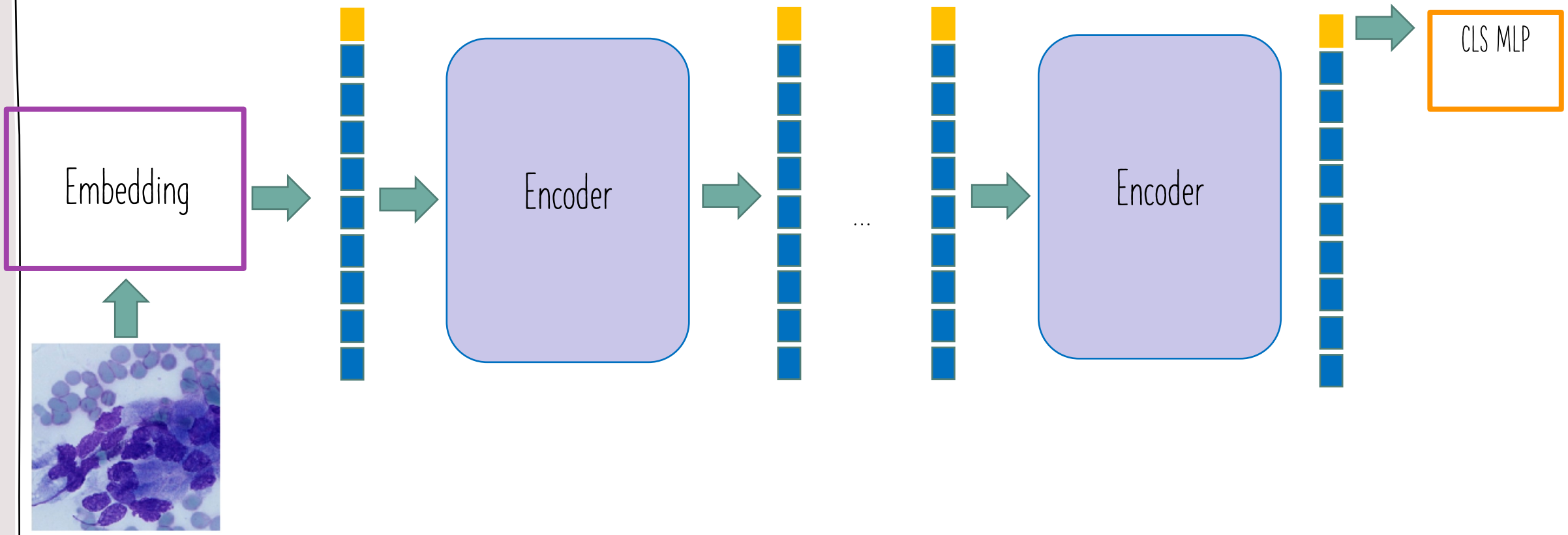
Shuffle step to understand the differences of the patches and understand what each patch are, across the samples.

Non shuffle step to understand the relations of the patches in the original image.

Tianyi Zhang, Youdan Feng, Yunlu Feng, and Guanglei Zhang*, "Pancreatic cancer ROSE image classification based on multiple instance learning with shuffle instances," arXiv:2206.03080, 2022.

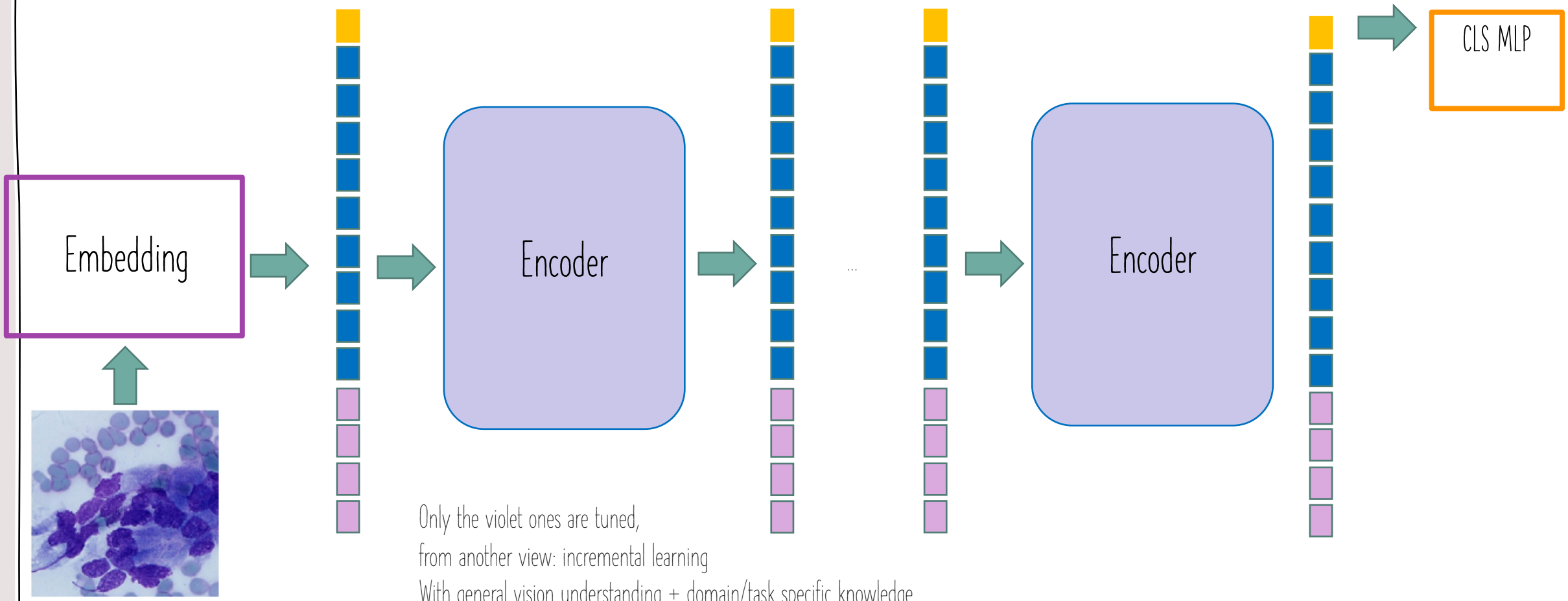
ViT

AN IMAGE IS WORTH 16X16 WORDS: TRANSFORMERS FOR IMAGE RECOGNITION AT SCALE
arxiv 2010.11929



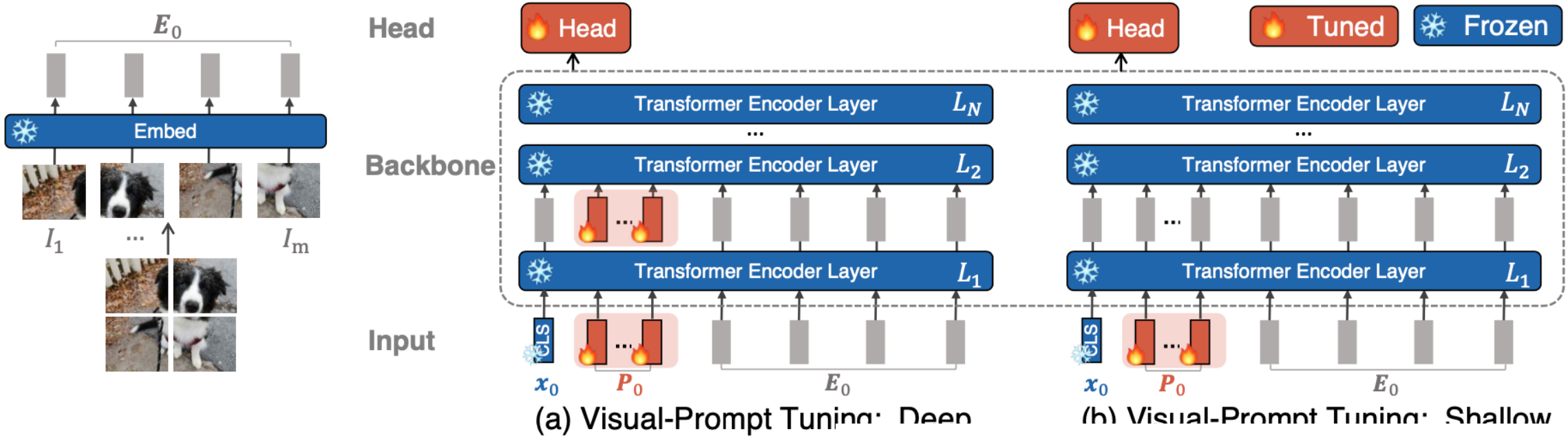
VPT

Jia, M., Tang, L., Chen, B.C., Cardie, C., Belongie, S., Hariharan, B. and Lim, S.N., 2022. Visual prompt tuning. *arXiv preprint arXiv:2203.12119*.



Only the violet ones are tuned,
from another view: incremental learning
With general vision understanding + domain/task specific knowledge

VPT: use the Prompt tokens in the tuning process



$$[\mathbf{x}_i, \_, \mathbf{E}_i] = L_i([\mathbf{x}_{i-1}, \mathbf{P}_{i-1}, \mathbf{E}_{i-1}])$$
$$\mathbf{y} = \texttt{Head}(\mathbf{x}_N) \quad .$$

$$[\mathbf{x}_1, \mathbf{Z}_1, \mathbf{E}_1] = L_1([\mathbf{x}_0, \mathbf{P}, \mathbf{E}_0])$$
$$[\mathbf{x}_i, \mathbf{Z}_i, \mathbf{E}_i] = L_i([\mathbf{x}_{i-1}, \mathbf{Z}_{i-1}, \mathbf{E}_{i-1}])$$
$$\mathbf{y} = \texttt{Head}(\mathbf{x}_N) \quad ,$$

My version of code: https://github.com/sagizty/VPT


Thanks
Tianyi
PPT: https://github.com/sagizty/Insight/blob/main/Dive%20into%20Transformer.pdf
Code: https://github.com/sagizty/Insight/blob/main/ZTY_Torch/ViT.py