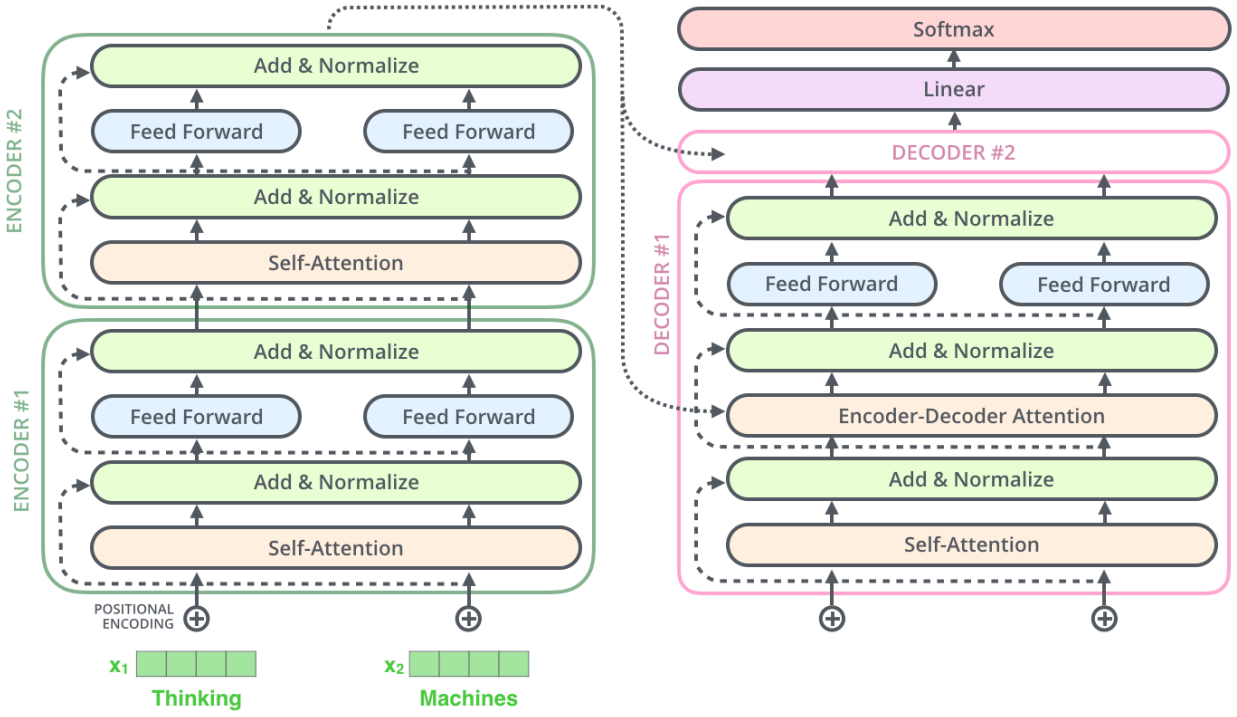
SegFormer Model - a Transformer based framework for semantic segmentation.

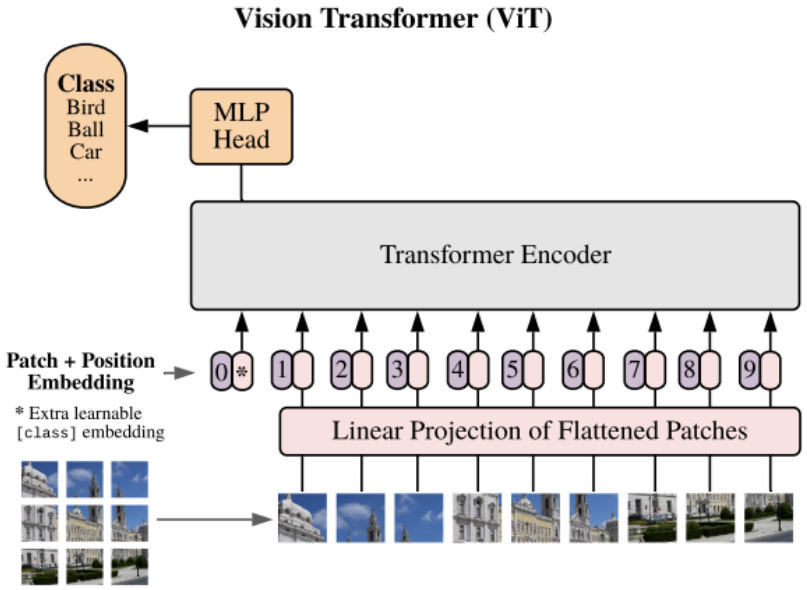
Nguyen Hoang-Phuong

Content:

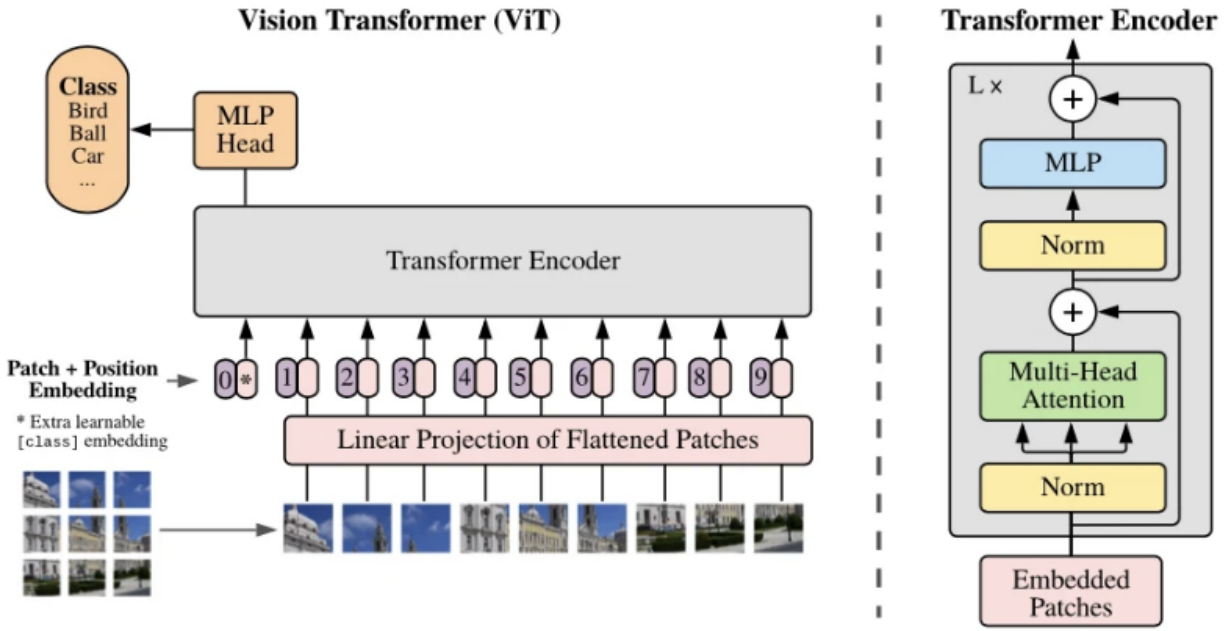
- Vision Transformer
- Segformer Model:
  - Encoder
  - Decoder

Transformer: From NLP to CV



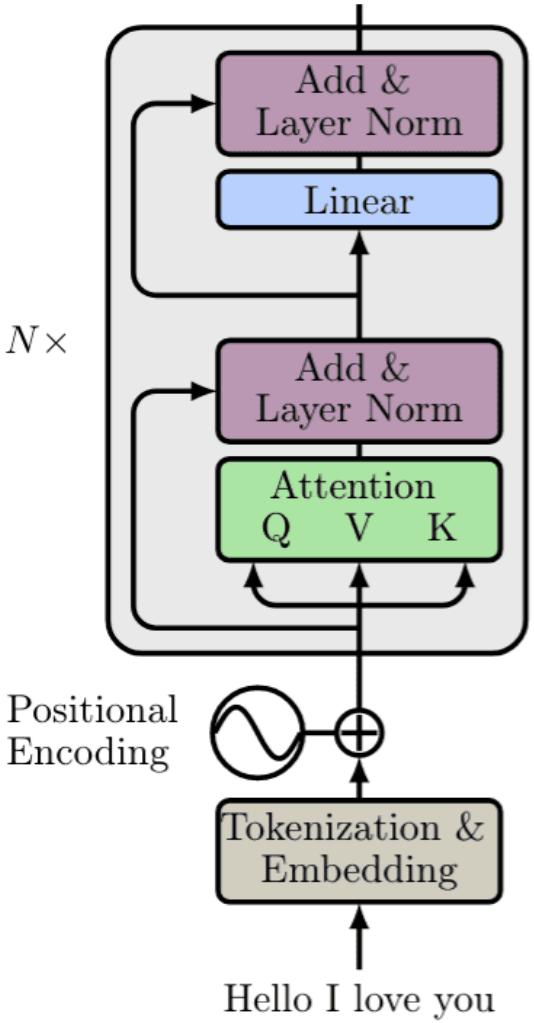


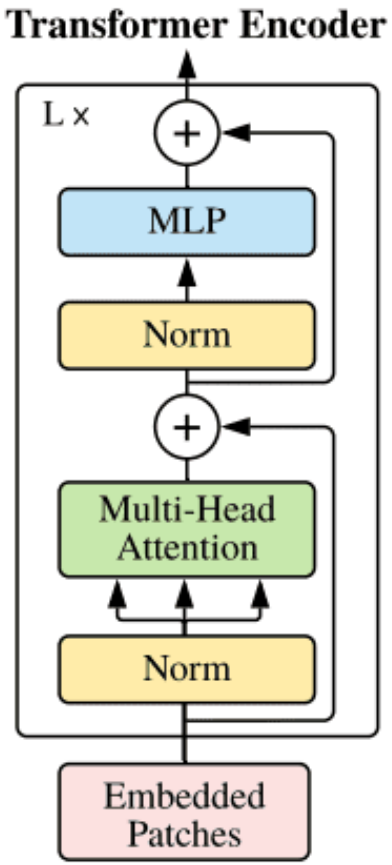
Vision Transformer



Vision Transformer ViT Architecture – Source

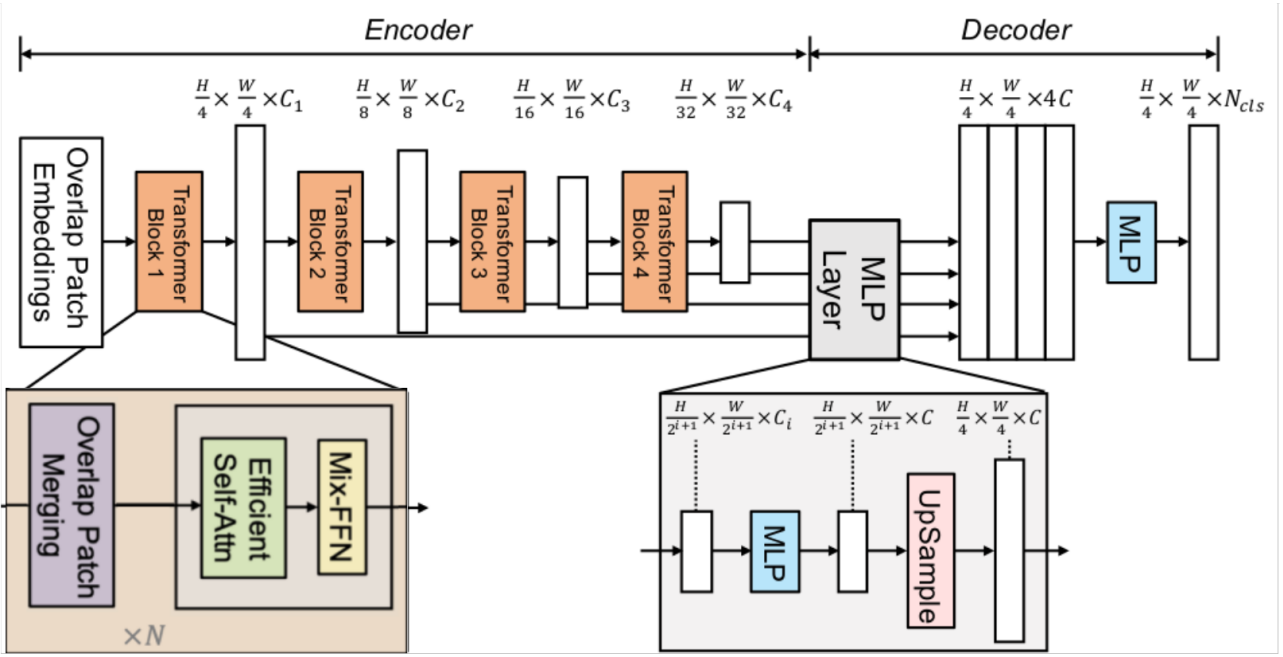
- Split an image into patches
- Flatten the patches
- Linear embeddings from the flattened patches
- Add positional embeddings
- Feed the sequence into transformer encoder
- Using multi layer perceptron MLP to get classification head


Add &
Layer Norm
Linear
N×
Add &
Layer Norm
Attention
Q V K
Positional
Encoding
Tokenization &
Embedding
Hello I love you


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

Vision Transformer vs CNN

Property	CNNs	Vision Transformer
Shift invariance	Mostly	No
Permutation invariance	No	Mostly (at patch level)
Spatially local processing	Yes 3x3, 5x5 conv	Partially (only patching)
Parameter sharing	Yes	Yes
With increased depth	Yes	No



Segformer Model

SegFormer has the following notable points:

- Encoder: Mix Transformer (MiT) that extracts coarse and fine features
- Decoder: MLP network to directly fuse the multi-level features of the encoder part and predicts the semantic segmentation mask

Encoder

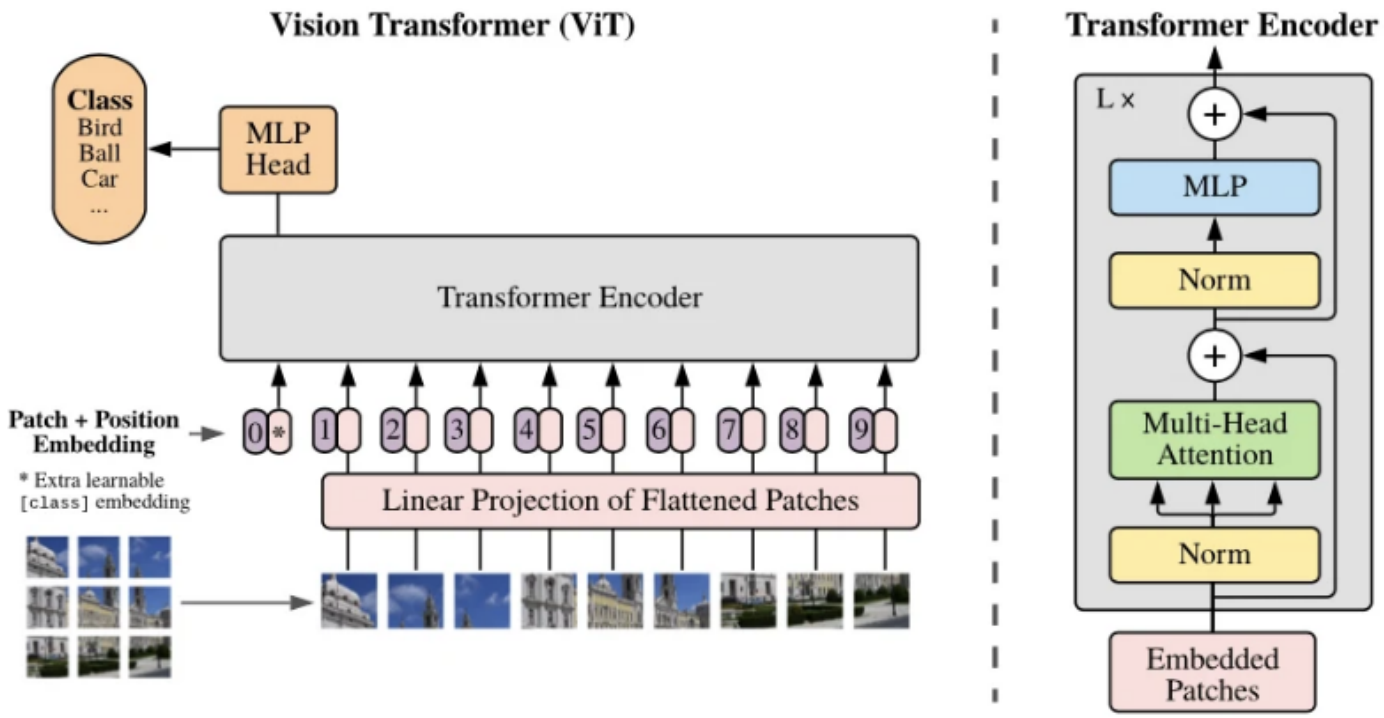
Hierarchical Feature Representation

- ViT that can only generate a single-resolution feature (one head)
- MiT generate multi-level features (MiT have 4 heads)

Overlapped Patch Merging

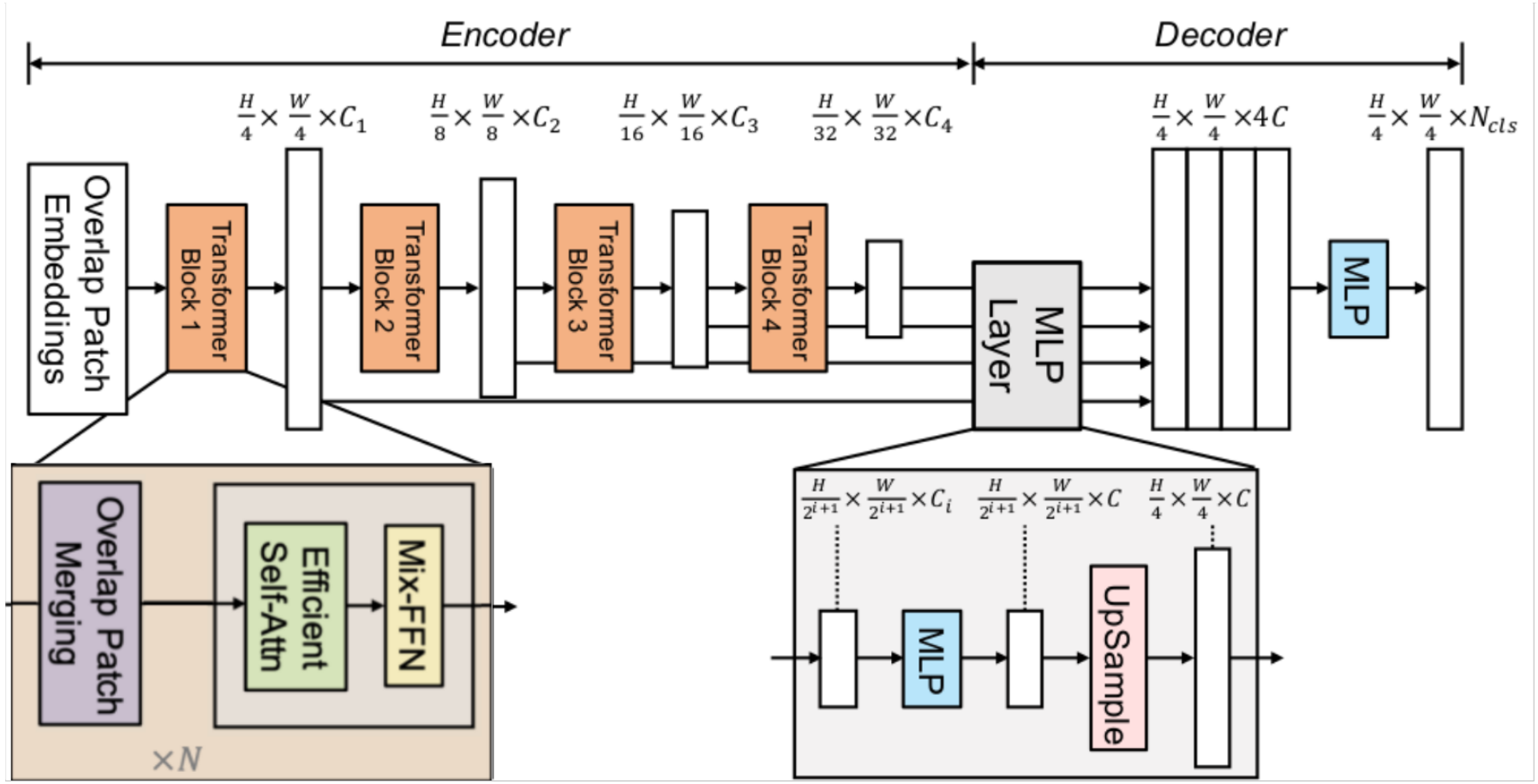
- In ViT, a image input is splitted into **partition** patches.
- Mix Transformer, a image input is splitted into **overlapping** patches.

ViT model

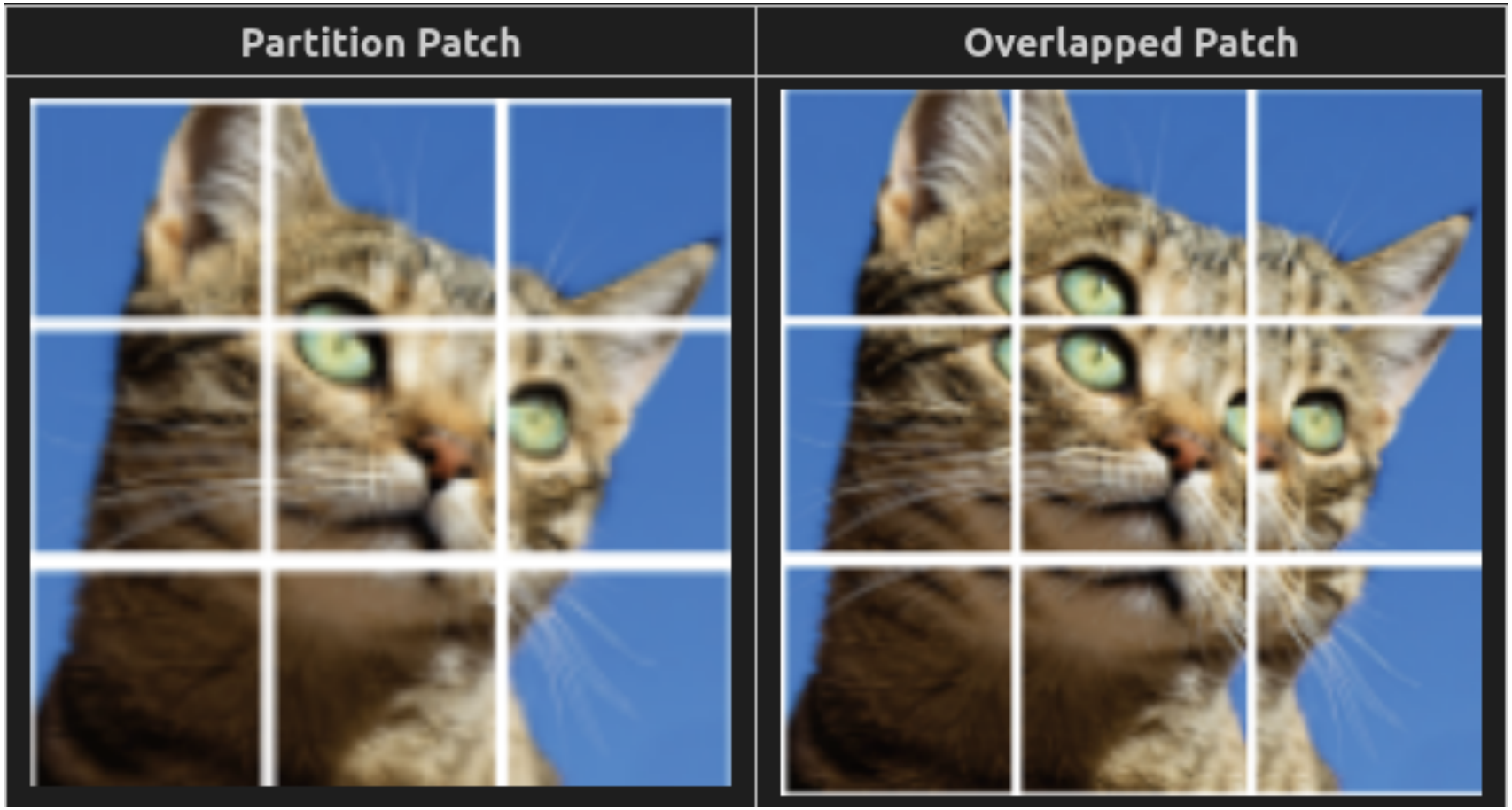


Vision Transformer ViT Architecture – Source

Segformer model



Segformer model

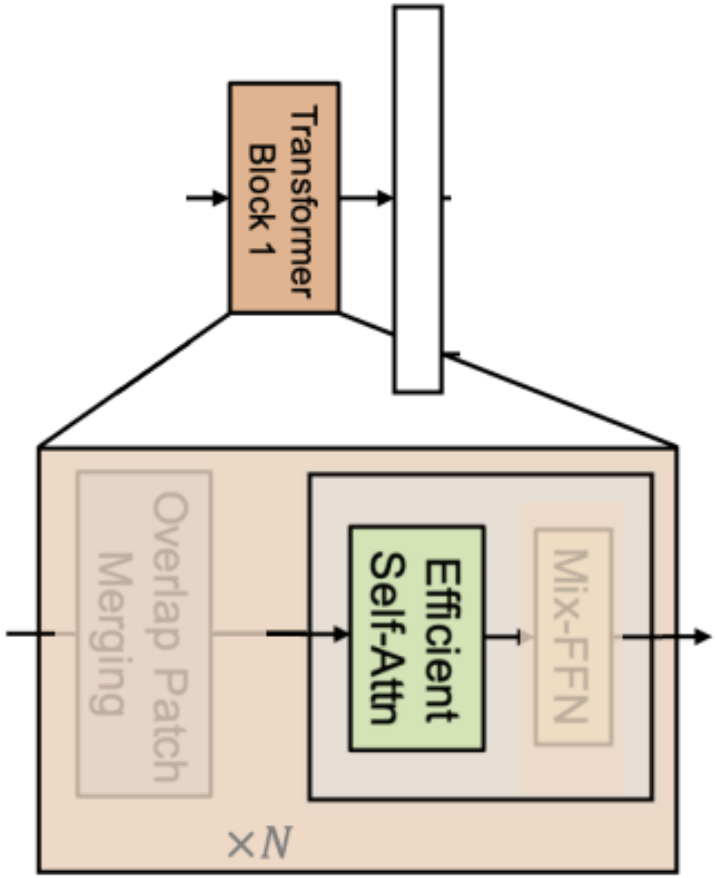


Encoder

- With the overlapping patches, the MiT are using a CNN layer. That helps the model learn better the local feature.

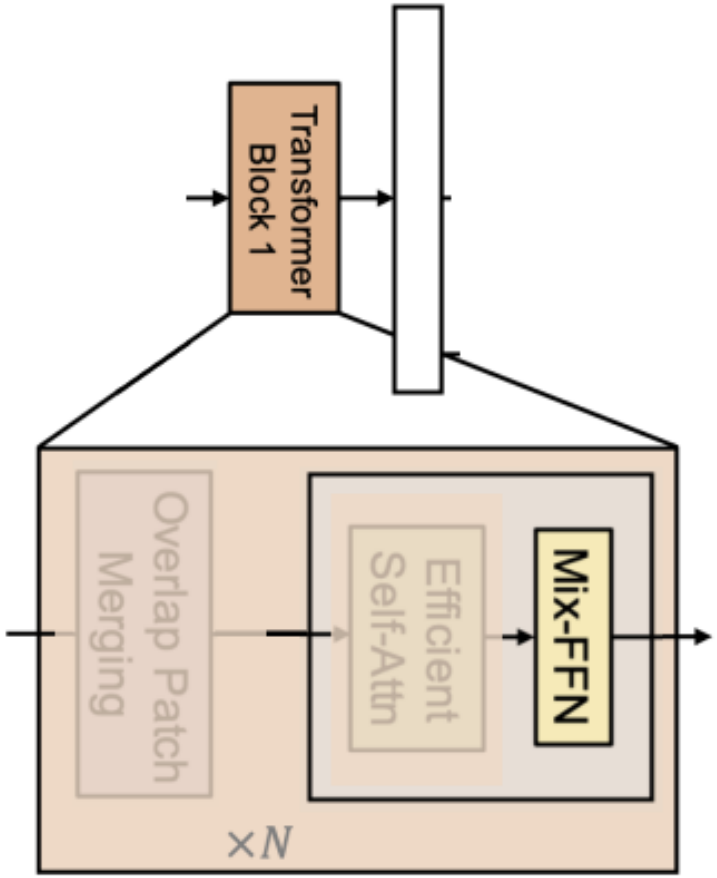
Is that why we call Mix Transformer?

- Multi-level features idea is one of the most important ideas for the semantic segmentation model.



Encoder

- ViT uses Multi Head Attn (Computation of $O((H \times W)^2))$
- MiT uses Efficient Self Attn (Computation of $O((H \times W)^2 / R))$



Encoder

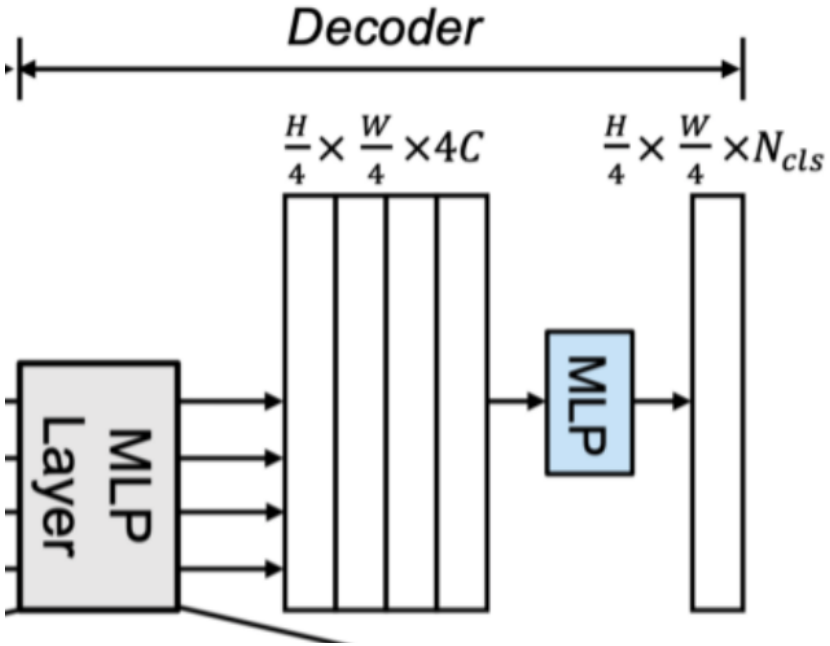
- ViT uses MLP and GELU activation function.
- MiT uses

$$x_{out} = MLP(GELU(CONV_{3\times3}(MLP(x_{in})))) + x_{in}$$

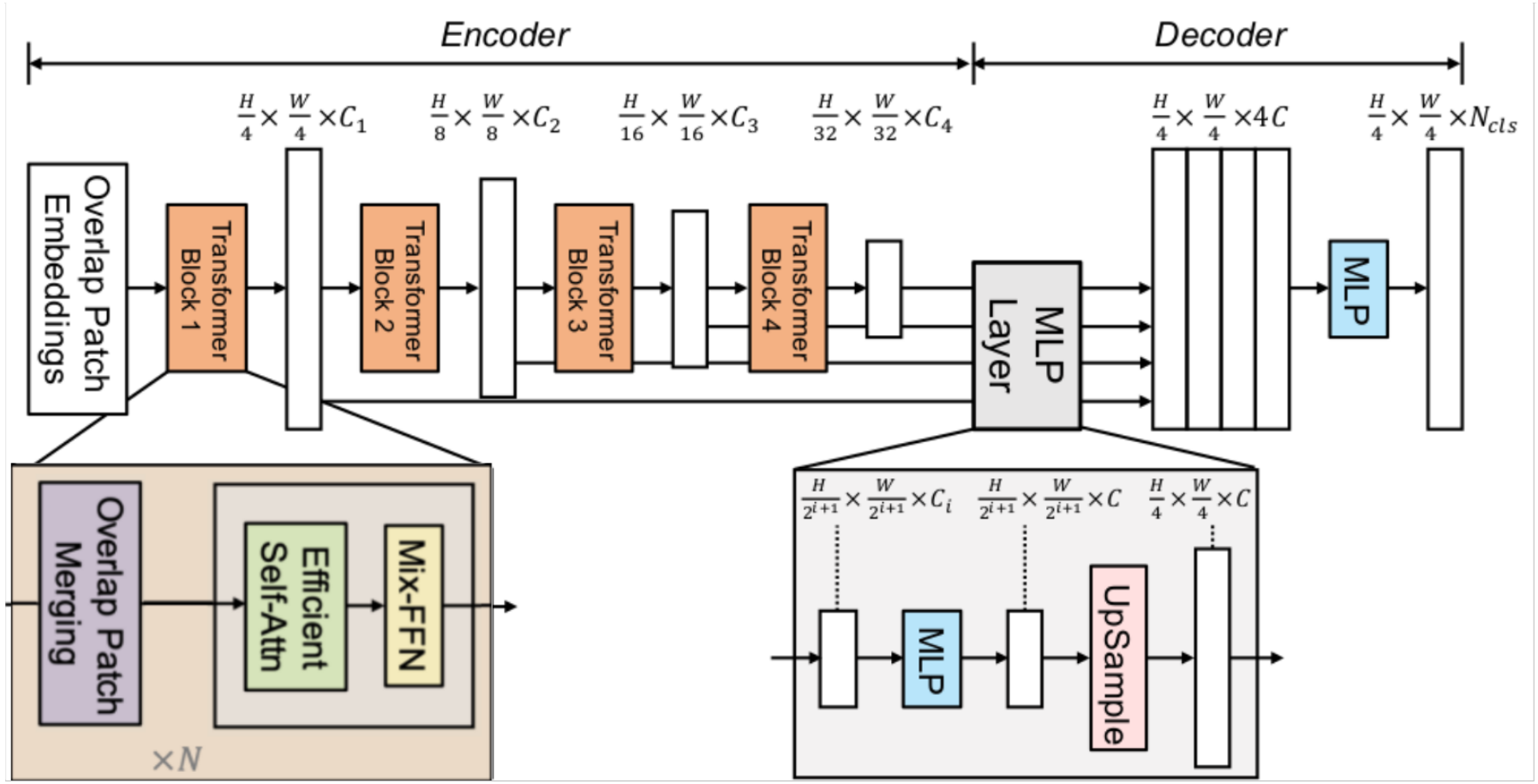
- MLP = nn.Linear(dim, num_classes) (or Dense Layer)
- One again, MiT uses a CNN.

Decoder

Because Encoder is quite good, then to avoid the overfitting, we use a MLPs to fuse the features of the encoder and predict the semantic segmentation mask.



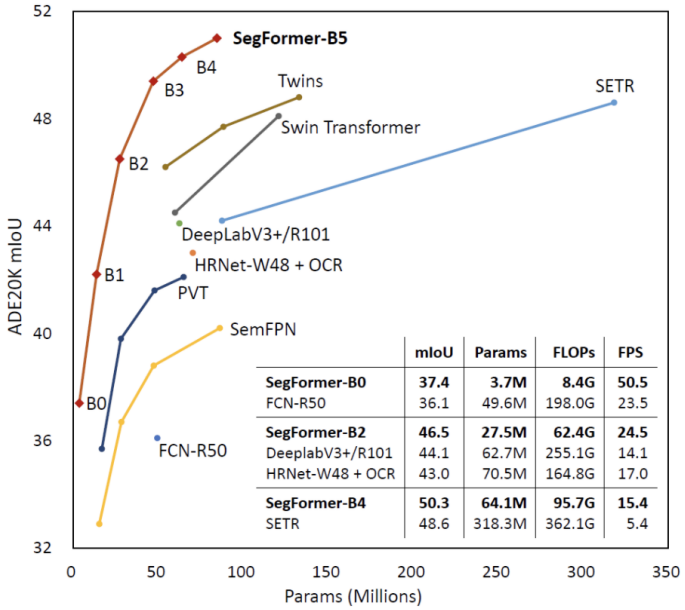
Decoder


Encoder
Decoder
$\frac{H}{4}\times\frac{W}{4}\times C_1$
$\frac{H}{8}\times\frac{W}{8}\times C_2$
$\frac{H}{16}\times\frac{W}{16}\times C_3$
$\frac{H}{32}\times\frac{W}{32}\times C_4$
$\frac{H}{4}\times\frac{W}{4}\times 4C$
$\frac{H}{4}\times\frac{W}{4}\times N_{cls}$
Overlap Patch Embeddings
Transformer Block 1
Transformer Block 2
Transformer Block 3
Transformer Block 4
MLP Layer
MLP
Overlap Patch Merging
Efficient Self-Attn
Mix-FFN
$\times N$
$\frac{H}{2^{i+1}}\times\frac{W}{2^{i+1}}\times C_i$
$\frac{H}{2^{i+1}}\times\frac{W}{2^{i+1}}\times C$
$\frac{H}{4}\times\frac{W}{4}\times C$
MLP
UpSample

A MLP of decoder part

```
class SegFormerDecoderBlock(nn.Sequential):
    def __init__(self, in_channels: int, out_channels: int, scale_factor: int = 2):
        super().__init__(
            nn.UpsamplingBilinear2d(scale_factor=scale_factor),
            nn.Conv2d(in_channels, out_channels, kernel_size=1),
        )
```

performance of Segformer



	mIoU	Params	FLOPs	FPS
SegFormer-B0	**37.4**	**3.7M**	**8.4G**	**50.5**
FCN-R50	36.1	49.6M	198.0G	23.5
SegFormer-B2	**46.5**	**27.5M**	**62.4G**	**24.5**
DeeplabV3+/R101	44.1	62.7M	255.1G	14.1
HRNet-W48 + OCR	43.0	70.5M	164.8G	17.0
SegFormer-B4	**50.3**	**64.1M**	**95.7G**	**15.4**
SETR	48.6	318.3M	362.1G	5.4

Figure 1: Performance of SegFormer-B0 to SegFormer-B5.

Segformer works well on a large dataset.

References

- SegFormer: A Transformer-based Framework for Semantic Segmentation
- Training the Segformer model by using Pytorch Lightning and HuggingFace
- Usage of TensorFlow based SegFormer in hugs transformers