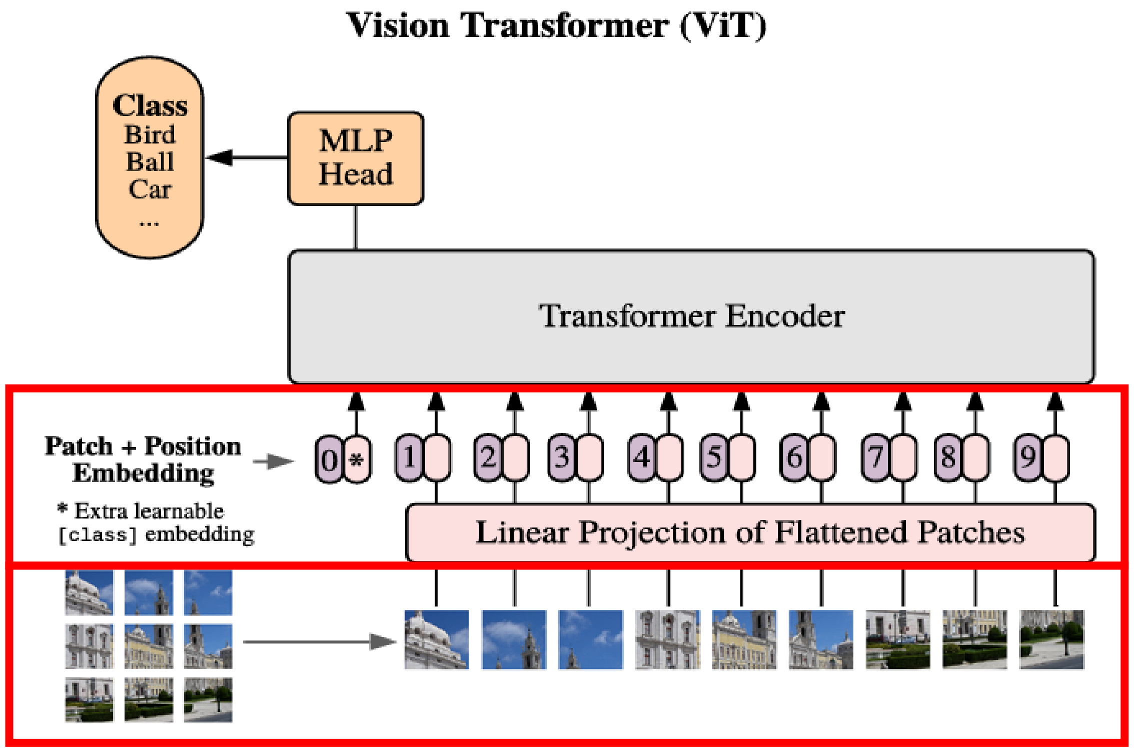
ViT Short Tutorial

James

Splitting images into patches



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

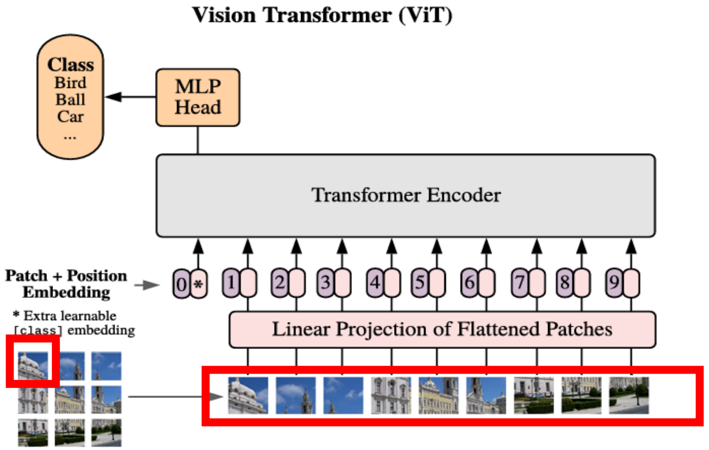
https://arxiv.org/abs/2010.11929

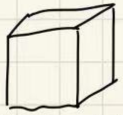
Q1: why do we divide it into small patches?

Main reason is that **image is too big for Transformers**.

Typical Transformers operates on a sequence of tokens of a maximum length 512. But a small image in CiFar10 has shape 32 * 32 *3, which after flattening gives 3072 length of tokens.

How to Split?





embed_dim = (patch_size^2) × in_channel

$= 16^2 \times 3 = 768$

in_channel = 3

img_size = 224

patch_size = 16

batch_size = 512.

① (512, 3, 224, 224)

nn.Conv2d(
in_channel = in_channel,
out_channel = embed_dim,
kernel_size = patch_size,
stride = patch_size)

output size $H' = \left\lfloor \frac{H + 2\times \text{padding} - \text{Kernel}}{\text{Stride}} \right\rfloor + 1$

$= \left\lfloor \frac{224 - 16}{16} \right\rfloor + 1$

$= 14$

② (512, 768, 14, 14)

num_patches = $H' \times H'$

$= 14 \times 14 = 196$

nn.flatten(2)

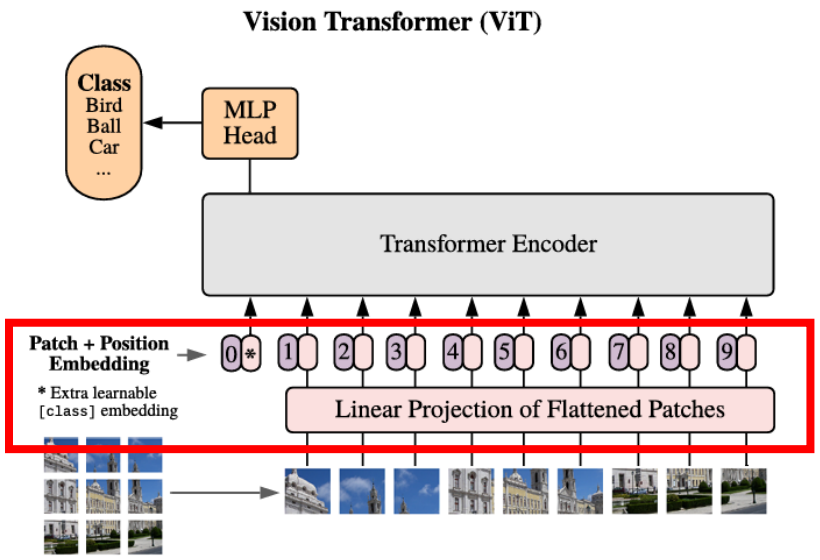
③ (512, 768, 196)

nn.permute(0,2,1)

④ (512, 196, 768)

From a (3,224,224) image
divide into 196 patches
of size 768 dimension

ViT-step2 cls_token and position embedding



Operation	Size
Input_size	(512, 3, 224, 224)
nn.Conv2d	(512, 768, 14, 14)
nn.Flatten(2)	(512, 768, 196)
nn.permute(0, 2, 1)	(512, 196, 768)
Torch.cat([cls_token, x], dim=1)	(512, 197, 768)
X = x + self.position_embedding	(512, 197, 768)
Transformer block	(512, 197, 768)
MLP-Head	(512, 10)

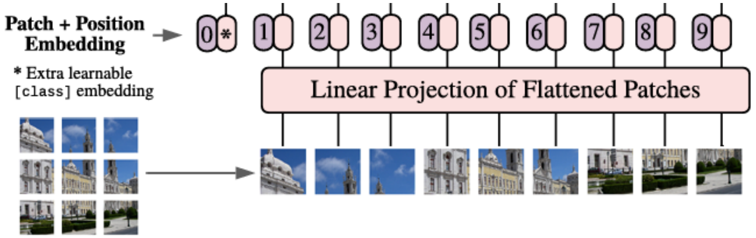
```
def forward(self, x):
    # initiation cls token, and match the batch size of x in the 0th dimension
    cls_token = self.cls_token.expand(x.shape[0], -1, -1)
    # after flatten we need to permute the dimension
    x = self.patcher(x).permute(0, 2, 1)
    # concat cls token
    x = torch.cat([x, cls_token], dim=1)
    # add position embedding
    x = x + self.position_embedding
    x = self.dropout(x)
    return x
```

Transformer Encoder v.s. CNN

Position Embedding

Source: *"Intriguing Properties of Vision Transformers" https://arxiv.org/abs/2105.10497*

```
self.position_embedding = nn.Parameter(torch.randn(size=(1, num_patches+1, embed_dim)), requires_grad=True)
```



Randomness Position Embedding **works**

Because **position_embedding assumes no prior information**. In other words, each patch position is learnt from training.

Thanks for watching