Practical work 6: Vision Transformer (Vi-T) for digit sequence recognition in images

All along this classroom work we will train then test a Transformer encoder-decoder Network for the recognition of handwritten digit sequences in images. We will use new datasets for this purpose, namely:
"MNIST_5digits2D.pkl"
"MNIST_5digits2DHorizontal.pkl"
"MNIST_5digits2DVertical.pkl"

The squared images of size (120,120) pixels contain a single sequence of 5 digits organized Horizontally (class n° 10) or Vertically (class n° 11). The horizontal (resp. vertical) sequence happens at a random vertical position (resp. horizontal) in the image.

To account for the vertical or horizontal organization of the sequence two specific tokens have been introduced in the ground truth (class 10 and class 11). The Reading system will have to identify the reading order of the sequence by identifying a first reading order token as depicted in figure 1.

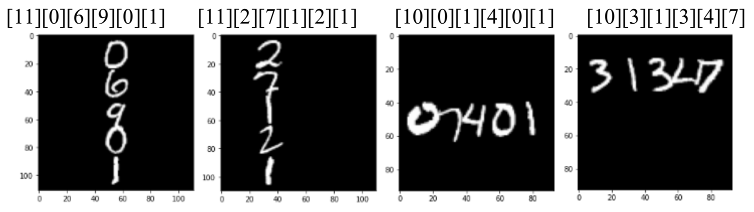

*Figure 1: examples of the example from the "*MNIST_5digits2D" dataset.

The MNIST_5digits2DHorizontal.pkl (resp. MNIST_5digits2DVertical.pkl) dataset contains horizontal (resp. vertical) sequences only, while the MNIST_5digits2D.pkl contains a mixture of horizontal and vertical sequences. Each dataset contains 12000 training images and 2000 testing images.

The Vision Transformer Architecture (ViT) architecture

The Encoder-Decoder architecture depicted on figure 2 below takes as input a sequence of image patches that are transformed into a sequence of feature vectors X. A 2D positional encoding is introduced to carry the original position of the path in the input image. Except the difference in pre-processing the input, the architecture remains similar to the one studied in the previous sessions, it is a standard transformer-based encoder-decoder architecture.

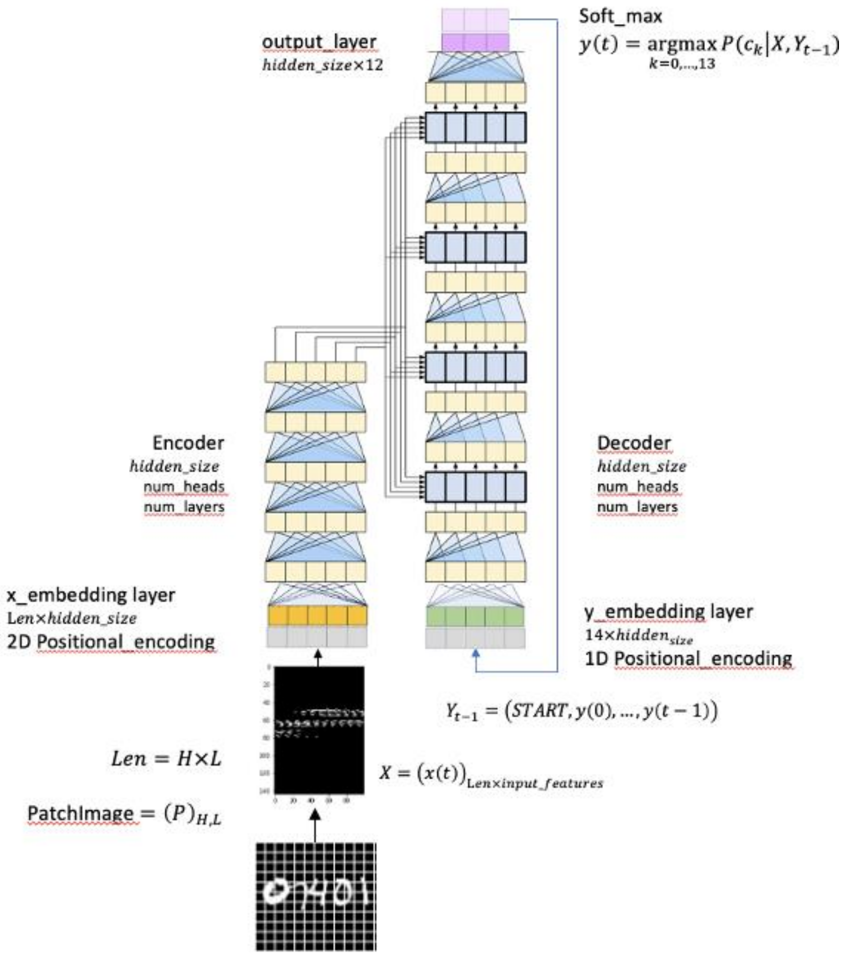

Figure 2: the ViT encoder - decoder architecture.

The $2D$ postional encoding is similar to the $1D$ positional encoding but it contains a concatenation x positional encoding and the y positional encoding. For a fixed dimension of the model ($hidden\_size$), the first half dimension is used for storing the y positional encoding, while the second half stores the x positional encoding. The following equations summarize the encoding.

$$\mathrm{PE}_{\mathrm{2D}}(x, y, 2k) = \sin(w_k \cdot y) \quad (1)$$
$$\mathrm{PE}_{\mathrm{2D}}(x, y, 2k+1) = \cos(w_k \cdot y) \quad (2)$$
$$\mathrm{PE}_{\mathrm{2D}}(x, y, d_{\mathrm{model}}/2 + 2k) = \sin(w_k \cdot x) \quad (3)$$
$$\mathrm{PE}_{\mathrm{2D}}(x, y, d_{\mathrm{model}}/2 + 2k + 1) = \cos(w_k \cdot x) \quad (4)$$
$$\forall k \in [0, d_{\mathrm{model}}/4] \quad (5)$$
$$(6)$$

The image patches are obtained using a squared sliding window of size ($w\_width \times w\_width$). The input image of size 120 120 are serialized into a fixed length of patches of length $(120 \,/\, w\_width)^2$. The patches of pixels are represented as real vectors of size $w_{width} \times w_{width} = input\_features$ as represented in figure 3 below.

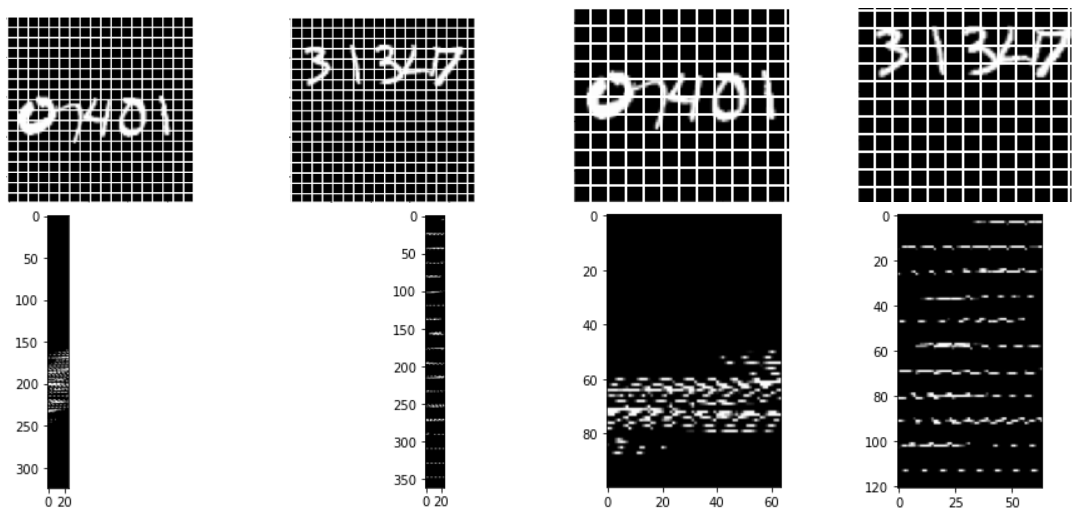

Figure 3: examples of 2D 5×5 sliding window (left) and 8×8 sliding window (roght), with their respective sequence of embedding.

Considering the fixed dimension of the input images (120×120), the embedding sequences, as well as the embedding vectors, have a fixed size depending on the size of the sliding window as summarized in table 1 below.
Make sure you configure the code with one of these entries !

w_width	input_features	length (= max_length^2)
5	25	$576 = 24 \times 24$
8	64	$225 = 15 \times 15$
10	100	$144 = 12 \times 12$
12	144	$100 = 10 \times 10$
15	225	$64 = 8 \times 8$

Table 1: input embedding size and sequence length with respect to sliding window size.

Experimentations

- Explore the possibility to make sequence recognition for the easy (Horizontal or vertical) and for the more complex (Horizontal and vertical) dataset with a ViT architecture.
- Explore the capability to train a hybrid CNN-Vit architecture.