# RDFIA – TP1-e – Transformers

CHAHINE Marilyn 21517286

MACQUET Jean 3532408

## Summary

The practical focused on exploring Vision Transformers (ViTs). Unlike CNNs that focus on local features, ViTs allow every patch to attend to all others, enabling a richer understanding of the image structure. But this comes at a computational cost, as the attention mechanism scales quadratically with the number of patches. In the practical, I implemented all the core components of a ViT (patch embedding, single-head and multi-head self-attentions, the MLP, and the final ViT structure with positional encoding to preserve spatial information, and the [CLS] token used for aggregating global representations).

Through experimentation on the MNIST dataset, I analyzed how different hyperparameters—such as embedding dimension, patch size, and number of transformer blocks affect model performance. Increasing the embedding dimension and number of blocks improved accuracy, while patch size had little impact due to the simplicity of MNIST images. I also observed that pretraining plays a crucial role on ViT performance when the dataset is too small to train a model with high accuracy. This highlighted the importance of transfer learning when dealing with small datasets. Finally, I encountered practical issues when applying pretrained ViTs to small grayscale images, which emphasized how CNNs remain more flexible with input size.

## Questions

### Q3: Self-Attention

**What is the main feature of self-attention, especially compared to its convolutional counterpart? What is its main challenge in terms of computation/memory?**

The main feature of self-attention is its ability to find relationships between different elements of the input sequence, regardless of the distance between them, while convolutions capture only local relationships. Self-attention allows every token (here, every image patch) to attend to every other token.

Its main challenge in terms of computation/memory is its quadratic complexity with respect to the number of tokens. The attention matrix has size $N \times N$, making it inefficient for large inputs.

**Equations:** The single-head equations are:

$$Q = XW_Q, \quad K = XW_K, \quad V = XW_V$$

The main equation is:

$$\text{Attention}(Q, K, V) = \text{SoftMax}\left(\frac{QK^T}{\sqrt{embedDim}}\right)V$$

where $W_Q$, $W_K$, and $W_V$ are the weights for the queries, keys, and values, and $embedDim$ is the embedding dimension.

## Q4: Multi-Head Self-Attention

**Equations:** For multiple heads, the equations are:

$$Q_i = XW_Q^i, \quad K_i = XW_K^i, \quad V_i = XW_V^i$$

$$\text{head}_i = \text{SoftMax}\left(\frac{Q_i K_i^T}{\sqrt{embedDim}}\right)V_i$$

where $h$ is the number of heads.

Finally, we need to concatenate the results for each head, and perform a final linear projection using $W_O$, the output projection matrix.

## Q6: Full ViT Model

**Class Token:** The `[CLS]` token is a learnable embedding prepended to the sequence of patch embeddings. After passing through the transformer encoder, the final hidden state corresponding to this token is used as the aggregate image representation for classification. Each image has one `[CLS]` token, allowing the model to summarize global information across all its patches.

**Positional Embeddings (PE):** Since transformers do not model the spatial structure of the input, positional embeddings are added to each patch embedding to encode the patch locations. Without positional information, the self-attention mechanism would treat the input as an unordered set of tokens. In text, PEs preserve the structure of sentences; in images, they preserve the location of the patches.

## Experimental Analysis

**Test different hyperparameters and explain how they affect performance — particularly `embed_dim`, `patch_size`, and `nb_blocks`.**

Table 1: Effect of different hyperparameters on ViT performance

| Embed_dim | Patch_size | Nb_blocks | Accuracy (%) |
|---|---|---|---|
| 8 | 7 | 2 | 92.13 |
| 16 | 7 | 2 | 95.05 |
| 32 | 7 | 2 | 96.41 |
| 32 | 14 (4 patches) | 2 | 97.06 |
| 32 | 28 (whole image) | 2 | 96.52 |
| 32 | 7 | 4 | 97.01 |
| 32 | 7 | 8 | 97.32 |

**Comment and discuss the final performance that you get. How to improve it?**

For the embedding dimension, we expect the performance to be better when the size of the embeddings increase since the input will be represented in more detail. We can see that with the test accuracy increasing gradually from 92.13%, to 95.05%, to 96.41% when increasing the embedding dimension from 8, to 16, to 32.

For the patch size, we expect the accuracy to be better when the patch size is smaller since it would mean that the image is studied under more spatial detail. But in this case, for patch sizes going from 7, 14, and 28 for 28x28 pixels images, we didn't see much difference in performance (96.41%, 97.06%, and 96.52% respectively), which could be due to the image already being small, so the patch size doesn't affect performance since the model already performs well using big patches. That could also be due to the simplicity of the MNIST images.

The number of blocks determines how many times the input will pass through the attention block. So, more blocks should lead to a more complex and abstract representation of the input. We thus expect the performance to be better as the number of blocks increase. And that is indeed what we see as we have performances of 96.41%, 97.01%, and 97.32% for 2, 4, and 8 number of blocks respectively.

**What is the complexity of the transformer in terms of number of tokens? How can you improve it?**

If the number of tokens is N, the time and space complexity of the transformer is O(N2) as the attention matrix is of size NxN (Where, for images, N corresponds to the number of patches). To improve this complexity, we can decrease the number of tokens by increasing the patch size. We could also

## Q8: Larger Transformers

**a) Load the model using the timm library without pretrained weights. Try to apply it directly on a tensor with the same MNIST images resolution. What it is the problem and why we have it? Explain if we have also such problem with CNNs.**

When we load a Vision Transformer (ViT) model from the `timm` library without pretrained weights and try to apply it directly to a tensor matching the MNIST image resolution (28×28 pixels), we encounter a problem because the ViT architecture expects input images of size 224×224 pixels. Since the model divides the image into patches, the embedding won't work properly.

The loaded ViTs are also designed for RGB images (3-channel inputs), while MNIST images are grayscale (1-channel). This issue arises from the architectural design of ViT models, since they rely on positional embeddings defined for a specific input size. CNNs are more flexible with input size because convolutional layers operate locally, allowing them to handle variable image sizes through receptive fields that adapt to input dimensions.

**d) Comment the final results and provide some ideas on how to make transformer work on small datasets. You can take inspiration from some recent work.**

| **Model** | **Test Accuracy** |
|---|---|
| No Pre-Training | 76.95% |
| DINO Pre-Training | 92.52% |
| ImageNet Pre-Training | 88.77% |

**Discussion:** Transformers require large amounts of data to learn useful representations, so the model that was only trained on a small dataset (no pretraining) doesn't perform well (accuracy = 76.95%).

Using ViTs pretrained on larger datasets improves accuracy significantly (DINO = 92.52%, ImageNet = 88.77%). To make transformers work better on small datasets, we can:

- Use pretrained weights from large-scale datasets (e.g., ImageNet, DINO, SIGLIP) and fine-tune these models on the small dataset.

- Apply self-supervised learning or strong data augmentation to increase data diversity.