OVO Computer Vision - The Vision Transformer

Raphaël Romand-Ferroni
CentraleSupélec
raphael.romandferroni@student-cs.fr

Abstract

*The transformer architecture, introducing the idea of self-attention, has revolutionized the field of Natural Language Processing (NLP). Its usage is now common for solving many different NLP tasks. It was only a matter of time for this architecture to be adapted for the Computer Vision field. It is finally at ICLR 2021, that Dosovitskiy et Al. introduced in their paper: An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale [1] a Transformer architecture designed for Computer Vision tasks: **The Vision Transformer** or **ViT**.*

1. Introduction

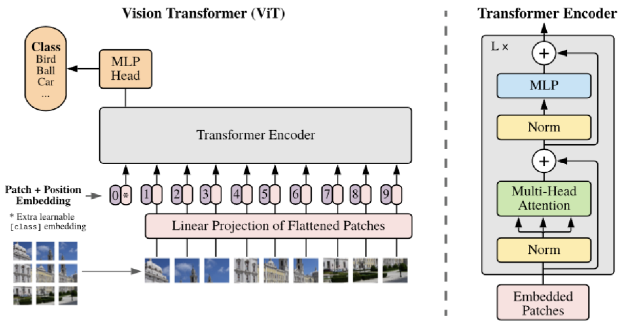

Figure 1. Vision Transformer (ViT) Architecture

We split an image into fixed-size patches, linearly embed each of them, add position embeddings, and feed the resulting sequence of vectors to a standard Transformer encoder. In order to perform classification, we use the standard approach of adding an extra learnable "classification token" to the sequence. The illustration of the Transformer encoder was inspired by Vaswani et al. (2017) in their paper Attention Is All You Need [4].

Specifically, the goal of this project is to replicate the original idea of the paper by:

- Implement from scratch the whole ViT architecture.
- Train the implemented architecture on the MNIST and CIFAR10 dataset and compare it to the vanilla ResNet fine-tuned model.
- Visualize attention maps from the trained ViT.

2. Implementation

2.1. Tokenization and Embedding

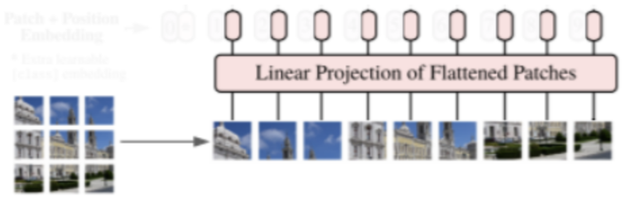

Figure 2. Tokenization and Embedding steps

In the traditional architecture of transformers, the initial stages involve **Tokenization** and **Embedding**.

- Tokenization consists in splitting the raw input data into individual tokens. For example, considering a sentence, it can be splitted into individual words (word-based tokenization), groups of characters (subword-based tokenization), individual characters (charachter-based tokenization). In the ViT architecture, the tokenization step splits the image into a 1D sequence of flattened patches.
- Embedding consists in projecting each individual token into a lower dimensional space. In the ViT architecture, the Embedding projects each flattened patch into a lower dimensional space.

In the ViT architecture, the tokenization and embedding steps can be done efficiently using a 2D convolution. The embedding of the image is implemented in the class PatchEmbedding.

Given a RGB image with dimensions $128 \times 128 \times 3$ and patches with dimensions 16×16, the final embedded shape will be of dimension 64×256 for the 2D convolution.

2.2. Positional Embeddings and Class token

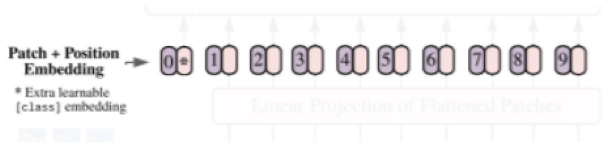
Patch + Position Embedding
* Extra learnable [class] embedding
0
1
2
3
4
5
6
7
8
9

Figure 3. Positional Embeddings and Class token steps

- The role of the positional embeddings is to associate to each token/patch a unique vector in order to inject into the network the positional information of this token/patch and get a sense of it in the overall context of the sentence/image. These positional embeddings can either be learned or fixed. In the original paper of Vaswani [4] in 2017, positonal embeddings are fixed with sine and cosine functions. In the ViT architecture, learnable positional embeddings are initialized randomly with standard gaussian distribution.
- A learnable class token with the same dimension as other tokens is prepended to the sequence of tokens. The output representation of the class token is fed to the MLP (Multi-layer Perceptron) head.

In the code implementation, the positional embeddings is added in the class AddPositionalEmbeddings, whereas the class token is implemented in the class PrependClassToken.

2.3. Transformer Encoder

As described in the Figure 4, the Transformer encoder is made of L stacked transformer blocks, and works identically to the original transformer proposed by Vaswani [4], except that inputs are image patches. Each transformer block is made of the following parts:

1. Layer Normalization
2. Concatenation of Multi-Head Attention representations
3. Add a residual connection
4. Layer Normalization
5. Multi-Layer Perceptron
6. Add a residual connection

The Multi-Head Attention part is the most crucial component of the Transformer architecture as it allows interactions between all the different patches.

Here is how it works as described in the original paper : Given a sequence of patches $Z = \{z_t\}_{\{1 \leq t \leq T\}}$, the patches are linearly projected using weights matrices W_Q, W_K and W_V to obtain respectively the sets of queries $Q = \{q_t\}_{\{1 \leq t \leq T\}}$, keys $K = \{k_t\}_{\{1 \leq t \leq T\}}$ and values $V = \{v_t\}_{\{1 \leq t \leq T\}}$.

Similarities between queries and keys are computed, rescaled and normalized, it is what we call the attention maps:

$$A = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)$$

Values are linearly combined according similarities between queries and keys to produce the ouput $Z' = \{z'_t\}_{\{1 \leq t \leq T\}}$:

$$Z' = AV$$

Eventually the output sequence of tokens Z' is linearly projected into the output sequence $Z'' = \{z''_t\}_{\{1 \leq t \leq T\}}$.

All the architecture transformer block is built in the class TransformerBlock.

2.4. MLP Head and final ViT implementation

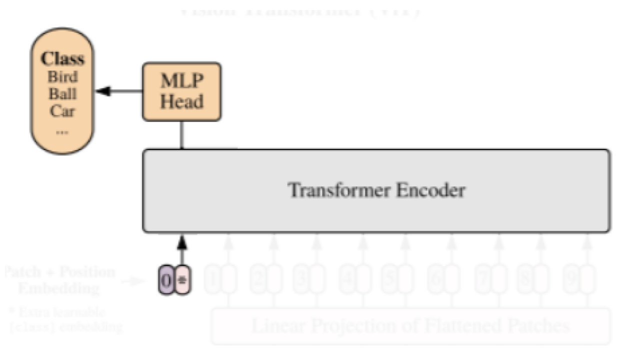

Figure 4. MLP Head and prediction steps

As explained in the paper [1] the final output of the network is produced by feeding the output representation of the class token to the MLP head which is a fully connected layer. This is implemented in the class MLPHead that takes as inputs the dimension of the class token output representation and the number of classes to predict.

The final ViT class takes as inputs the image size (H, W), the patch size (P_H, P_W), the number of channels if input images (e.g 3 if it's RGB images), dimension of the patch embedding which has to be chosen carefully considering the patch and image sizes, the number of transformer blocks and heads in each block, the number of hidden layer in the MLP and eventually the number of classes to predict. ViT renders the predicted class, the output tokens and the attention maps in each blocks of the transformer encoder that we are going to visualize in a last part.

3. Experiments

3.1. MNIST Dataset

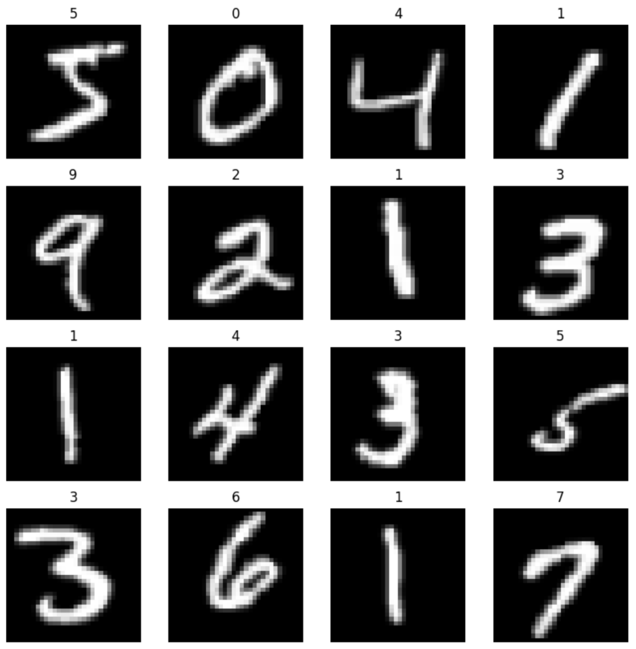

Figure 5. Random images from MNIST dataset

We first wanted to test our brand new model ViT on a multi-classification task on the most famous MNIST dataset [3] that consists of a large dataset of handwritten digits dispatched in 10 labels. We use 60 000 images for the training and 10 000 images for the testing. This implementation of the code and the data processing is quite straightforward for the MNIST dataset and easy to use in the case of ViT. As the images are all resized to 32 $\times$ 32 with only one input channel, we can choose patches of size 4×4 and an embedding dimension of 32. For this first implementation, we choose to stack 4 ($depth$ parameter in the code) transformer blocks composed each of 8 attention heads ($heads$ parameter in the code). For the data preprocessing, we apply the ToTensor() module that converts PIL images to a float tensor in the $[0, 1]$. Additionally, we apply a RandomAffine() transformation that translate randomly the image in both the horizontal and vertical directions by up to 5% with no rotation. For the training, we choose a batch size of 64 on 20 epochs with a cosine annealing scheduler on an initial learning rate of 0.001.

As depicted in the figure 6, ViT model achieve remarkable performances in term of accuracy during the training process. The model doesn't need much iterations through the data to achieve a good score. Final accuracy on the test dataset composed of 10 000 images with ViT is 98.10%. This is an improvement if we consider a classical CNN architecture that obtain a score of a mere 90% for the same number of epochs and parameters. Let's denote that we didn't tune our hyperparameters of our ViT model but it can have a great influence on the training process and the result. For example, the smaller the patch size, the longer the input sequences to the model become and thus it requires a longer computation time. For such images of size 32 $\times$ 32, a patch size of 4 is the most common.

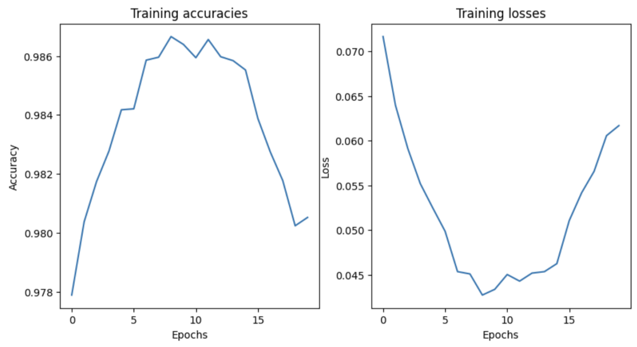

Figure 6. Training accuracies and losses on MNIST

3.2. CIFAR10 Dataset

The CIFAR-10 [2] dataset consists of 60000 32 $\times$ 32 RGB images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images.

Here, we must put 3 input channels to feed our ViT model as we have RGB images. As depicted in the Figure 7, the model has a nice and neat learning behaviour through the epochs on CIFAR10. but its final accuracy on the test dataset composed of 10 000 images with ViT is a mere 56.41%, which could have been improved by adding epochs in the training process. This poor result can be explained by the fact that Transformer models, in contract to CNN architectures, don't have inductive biases. CNN integrates the concept of distance between two pixels in an image, whereas ViT has to learn this information from the sparse learning signal of the classification task. This is why ViT performs best on large enough dataset such as MNIST than small dataset such as CIFAR10.

4. Visualize Attention Maps

ViT attention maps is very useful to get a sense of how the model works and provide an useful tool for explainability. Indeed, given a specific transformer block and attention head, one can retrieve the attention map associated to the similarity between the class token ($query$) and all other

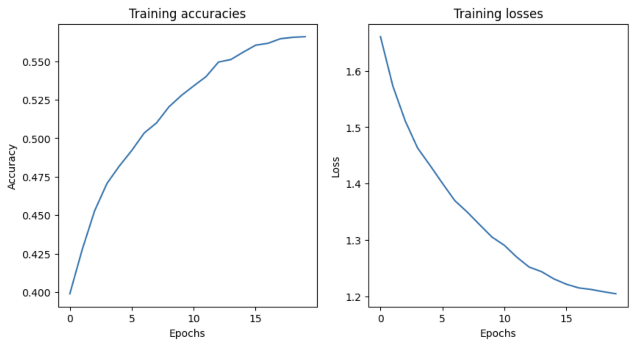

Figure 7. Training accuracies and losses on CIFAR10

tokens except the class token ($keys$). These attention maps give you an information about what part of the image is responsible for the network prediction.

On Figure 8 we plotted the attention maps of each transformer block (4 as defined in the code) of the model on a random image of the MNIST dataset. Here, the model is not very performant to detect where is the

5. Conclusion

In this project, we have implemented a our own Vision Transformer model from scratch and test it on a vanilla classification task using brick-and-mortar datasets, MNIST and CIFAR10. The model outperforms classical CNN networks on this simple task of classification on MNIST but achieves poor result on CIFAR10. ViT achieves state-of-the-art results on large datasets such as MNIST (or we could have tested it with ImageNet as well) but poor results on small datasets such as CIFAR10. By splitting an image in small patches as we would have done with a sentence by splitting it in tokens and by using multiple self-attention layers, ViT is able to embed the information globally across the overall image and learn the relative location of the image patches to reconstruct the structure of the original image.

References

[1] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021. 1, 2

[2] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research). 3

[3] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. The mnist database of handwritten digit images for machine learning research, 11 1998. 3

[4] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023. 1, 2

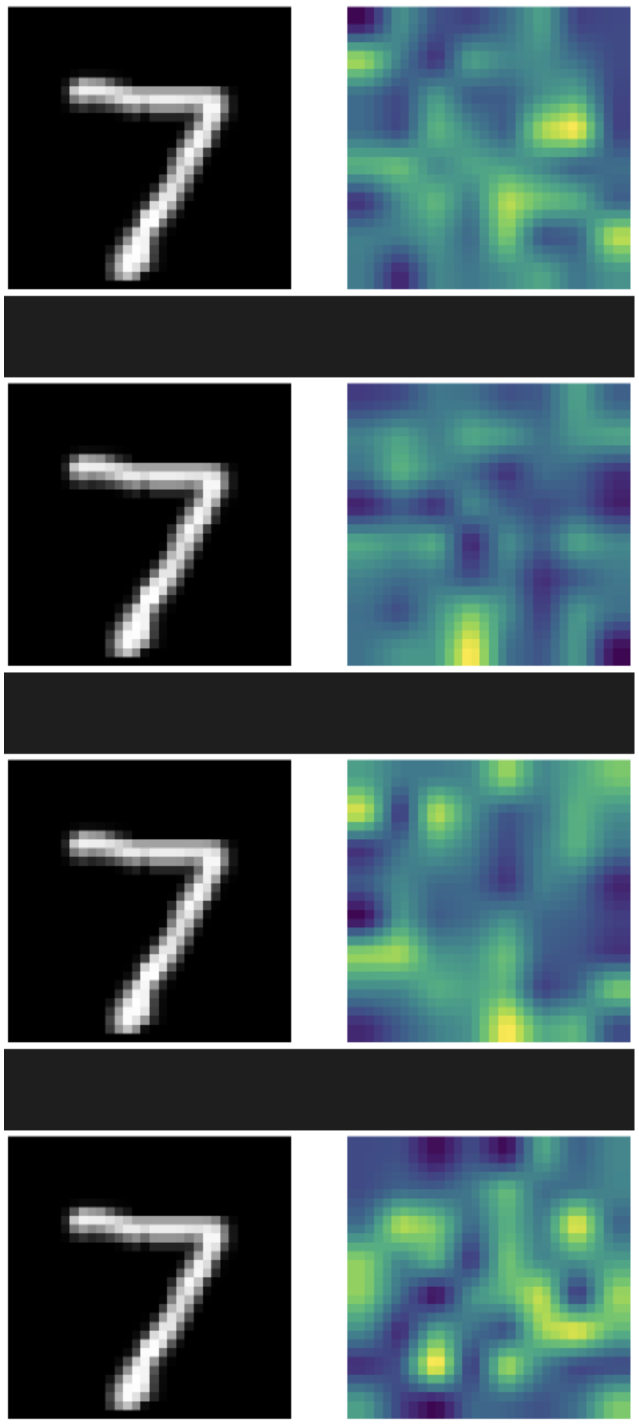

Figure 8. Attention Maps from MNIST