Comparative analysis of structure and performance of Convolutional Neural Networks and Vision Transformers

Author:
Léo Catteau

Supervision:
Antonio Sclocchi
Prof. Matthieu Wyart

TPIV - Physics project

Physics of Complex Systems Laboratory, PCSL
École Polytechnique Fédérale de Lausanne
Lausanne, Switzerland

January 5, 2024

Contents

1 Introduction

Convolutional Neural Networks (CNN) are artificial networks specifically made for image analysis, which have already proved their worth in machine learning. Emerging from Natural Language processing in the last few years, the attention mechanism of Transformers can also be adapted to computer vision, leading to the Vision Transformers (ViT) architecture. Understanding their non-trivial learning behaviors and comparing the two models is an actual research activity in the field.

In this work, we develop an experimental investigation of the differences between CNN and ViT. We try to reduce the two models to basic architectures and observe classical deep learning characteristics. We then perform quantitative comparisons of the learning curves with varying parameters, and we visualize the found variations with attention maps of the Vision Transformer.

2 Theory

2.1 Convolutional Neural Network (CNN)

Convolutional Neural Networks (CNN) are the default architecture to handle image datasets. This class of networks was created to include locality and translational invariance. They are composed of three types of layers:

- **Convolutional layers:** Basic layer of the CNN, it preserves locality by connecting each neuron only to a subset of the input, its receptive field. It consists in a set learnable filters that are convolved with the input. The corresponding post activation function is then

$$t_{a,\alpha}^{(l+1),\mu} = \varphi^{(l)} \left(\sum_{\beta=1}^{C_l} \sum_{b=1}^{w} W_{\alpha,\beta,b}^{(l)} t_{r_a(b),\beta}^{(l),\mu} \right), \tag{1}$$

  where $\varphi^{(l)}$ is the activation function, a is the position in the convolution layer, w the filter size, α the index of the channel of the convolutional layer, l the layer, μ the sample and C_l the number of channels in the layer l. $r_a(b)$ is the b^{th} neuron in the receptive field of filter a. In a foreward pass, each filter is sled across all the image, producing response at every spatial dimension.

- **Pooling layers:** Layers that combine multiple neuron outputs in a patch and reduce its dimension through a defined function, for example selecting the maximum or computing the mean of the patch. Pooling then reduces computational overhead and prevents overfitting.

- **Fully connected layers:** Classical neural network layer where all neurons have connections with all post activation functions from the previous layer. It performs scalar product between the inputs X_μ to a given layer and the trainable weights $W^{(l)}$, con-

necting this layer to the subsequent one, giving the output

$$f_W(X_\mu) = \varphi^{(L)}\left(\sum_{a_L=1}^{p_L} W_{a_L}^{(L)} \cdots \varphi^{(2)}\left(\sum_{a_2=1}^{p_2} W_{a_3 a_2}^{(2)} \varphi^{(1)}\left(\sum_{i=1}^{d} W_{a_2 i}^{(1)} X_{\mu i}\right)\right)..\right) \tag{2}$$

where $\varphi^{(l)}$ is the activation function, and p_l is the hidden layer dimension. The corresponding post activation function is then

$$t_{a_l}^{(l-1)\mu} = \varphi^{(l-1)}\left(\sum_{a_{l-1}=1}^{p_{l-1}} W_{a_l a_{l-1}}^{(l-1)} t_{a_{l-1}\mu}^{(l-2)}\right). \tag{3}$$

In practice, an arrangement of these layers with different parameters is made to extract features from the image. This choice in the architecture induces varying feature hierarchies. After treating the data to extract high level features, a head of fully connected linear layers is applied to the last stages of CNN to construct desired number of outputs, for example for classification task.

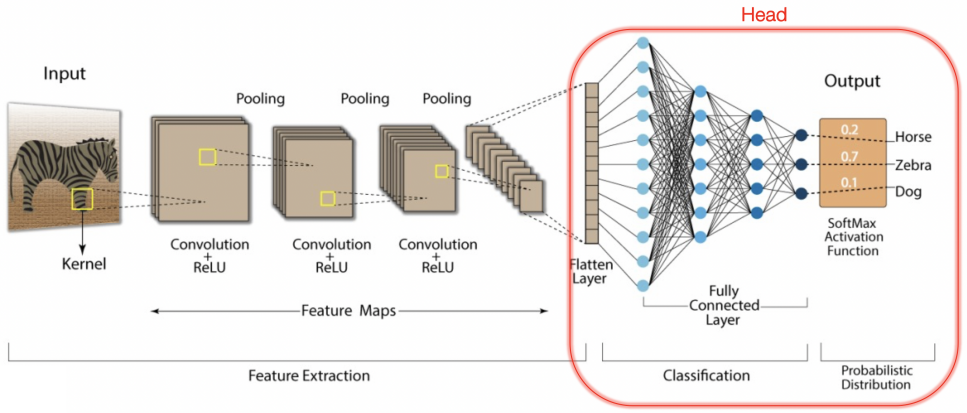


Figure 1: Graphic of the CNN architecture [2].

2.2 Vision Transformer (ViT)

Transformer is the most successfull architecture of the last years, originally created to handle sequences of data and used in natural language processing. To deal with variable length of input size and capture long term dependencies, the Transformer architecture operates through the self-attention mechanism. "The attention mechanism describes a weighted average of (sequence) elements with the weights dynamically computed based on an input query and elements' keys"; in other words, we want to dynamically decide on which inputs we want to "attend" more than others [3]. For its use in computer vision, the Transformer architecture is adapted towards the *Vision Transformer* (ViT) where tokens of the sequence are patches of an image.

From a sequence of tokens $(x_i)_{i=1,\dots,T}$, $x_i \in \mathbb{R}^{d_T}$, we linearly project values towards three vectors in higher dimensions:

- **Query:** $Q_i = W_Q \cdot x_i$, with $W_Q \in \mathbb{R}^{d_T \times d_Q}$, the feature vector describing what we are looking for in the sequence.
- **Key:** $K_i = W_K \cdot x_i$, with $W_K \in \mathbb{R}^{d_T \times d_K}$, the feature vector describing when the sequence might be important.
- **Value:** $V_i = W_Q \cdot x_i$, with $W_V \in \mathbb{R}^{d_T \times d_V}$, the feature vector over which we average.

We then define a score function that takes the query and a key as input, and output the attention weight of the query-key pair. In the case of dot product attention, the scalar product is the similarity metric. The ouput is then a mean of the softmax of the attention $a_{ji} = a(x_i, x_j)$ over the values (as illustrated in figure 2):

$$a(x_i, x_j) = \text{softmax}\left(\frac{Q_i \cdot K_j^T}{\sqrt{d_K}}\right), \quad \text{out}_j = \sum_{i=1}^{T} a(x_i, x_j) V_i \tag{4}$$

where the scaling by $1/\sqrt{d_K}$ is for having equal variance throughout the model.

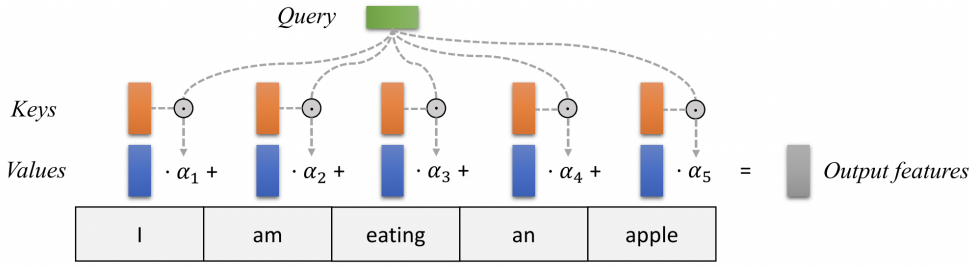
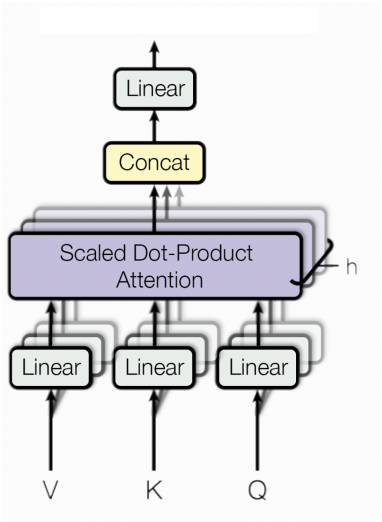
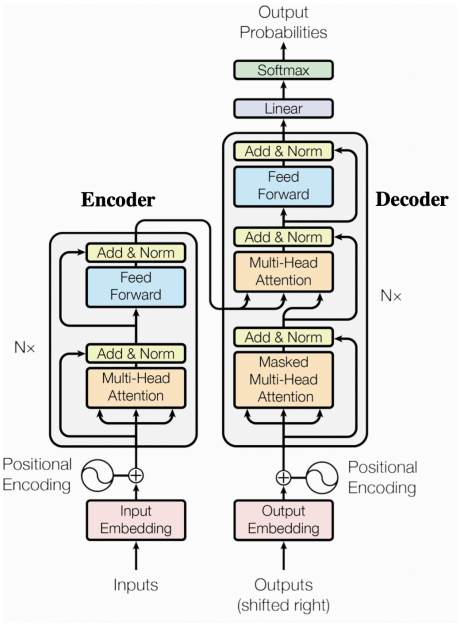


Figure 2: Graphic of the attention mechanism [3].

To pay attention to multiple aspects of a sequence, we can apply this same self-attention process multiple times with different individual (query, key, value) pairs called heads. The outputs from self-attention heads is then concatenated and fed to a feedforward neural network (figure 3a). Finally, positional encoding is added to handle the fact that attention block is permutation-equivariant, and we have all the components of the Transformer architecture (figure 3b).



(a)



(b)

Figure 3: Graphics of (a) the Multi-Head Attention [3] and (b) the Transformer architecture [4].

3 Methods

3.1 Architectures

Four our experiments, the idea is to build architectures of CNN and ViT, first as simple as possible, to compare their performances and behavior in learning for a classification task. Working on images resized to 32×32 form MNIST or CIFAR10 datasets, we create building block of architecture, spanning once all the image with a full CNN or a one-layer ViT as presented in figure 4. The parameters are chosen so that the treatment of the image is the most comparable as possible (similar patch sizes to treat comparable scale of locality), and to let only few free parameters controlling the complexity of the model for the study.

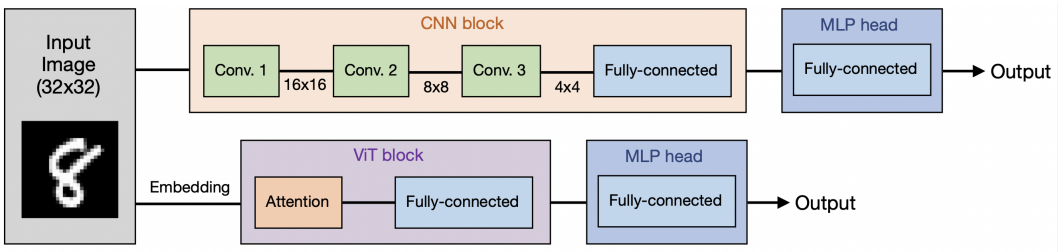


Figure 4: Graphical representation of the basic blocks of CNN and ViT used in experiments.

The CNN model is composed of three convolution layers (kernel=5, stride=1, padding=2),

each followed by a ReLU activation function and an average pooling layer (kernel=2, stride=2). The free parameter is the number of output channels of the first convolutional layer C_1 with $C_2 = 2C_1$ and $C_3 = 2C_2$. Finally, after being flattened, the output is fed to a two-layers fully connected MLP head with fixed hidden dimension ($h_{dim} = C_3$) and output of size 10 for classification (figure 5).

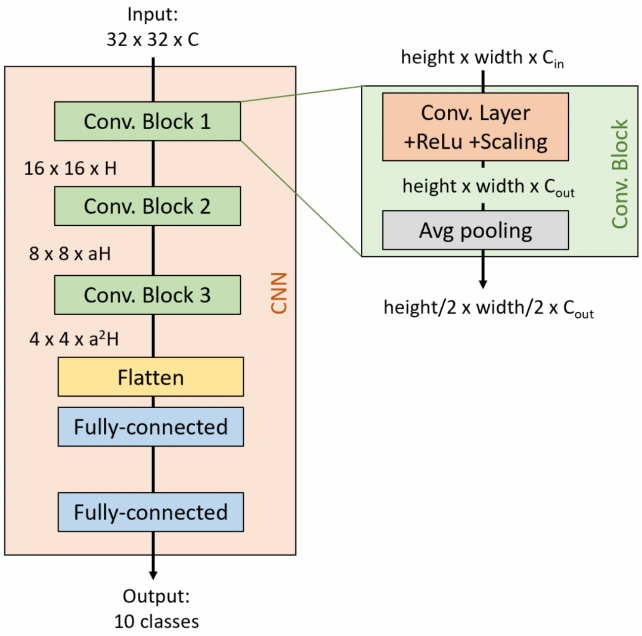


Figure 5: Graphical representation of the CNN architecture used in experiments (courtesy of Marco Di Mambro).

The Vit model is composed of a first part preprocessing the data. The image is divided into patches of size 4×4 (as illustrated in figure 6), witch are then flattened and to which we add one class token (patch), for enhancing attention on the classification task, and positional encoding to add locality at the inter-patches scale. The data is then fed to a encoder block as described before, with layer normalization (LayerNorm) and residual connections. The normalization enables smoother gradients, faster convergence, and better generalization accuracy, and the residual connections smoothes the energy (loss) landscape by allowing the gradients to flow directly through the network. The free parameter of the model is the embedding dimension h_{embed}, used to control the number of trainable parameters of the model.

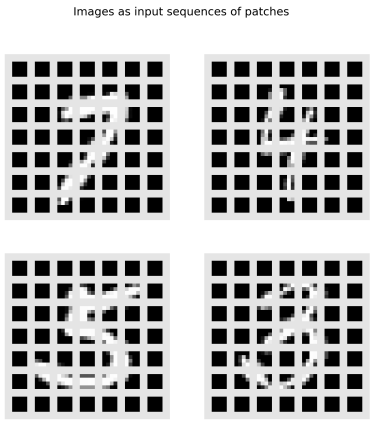


Figure 6: Example of patchifying of MNIST datased with patches of size 4×4.

The hidden dimension of the linear layer of the encoder block is $h_{dim} = 2h_{embed}$ and it's activation function is GELU. In a basic ViT block is one attention head, one encoding layer, and we vary the parameters for experiments. Finally, a MLP head passing from input h_{embed} to output 10 after LayerNorm is applied for classification (figure 7).

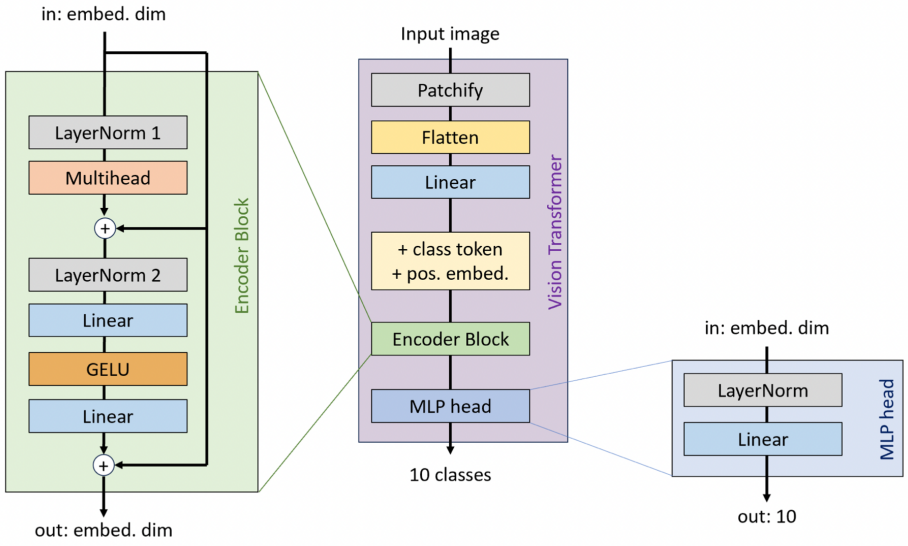


Figure 7: Graphical representation of the ViT architecture used in the experiments (courtesy of Marco Di Mambro).

3.2 Dynamics

We use a general routine of code training a given model on a given dataset with parameters as arguments; storing training quantities for plots. The code also takes as arguments differents parameters of the dynamics of training.

For our experiments, CNN models are trained using Stochastic Gradient Descent, while ViT's are trained using AdamW. As complextity evolves with the free parameters through our experiments, gradient scale as $1/h$ for the CNN. We then introduce effective learning rate

$$\eta_{eff} = \frac{\eta}{h} \tag{5}$$

where h is the width (number of output channels C_1 for CNN), to ensure that the model trains with steps of constant size, independently of h. In practice, we use $\eta_{eff} \approx 1$ for CNN and classical learning rate $\eta \approx 1 \cdot 10^{-3}$ for ViT.

The stopping of the training is managed by the following stopping criteria

- time exceeding defined max_wall value,
- number of steps exceeding definet max_step value,
- convergence of the training error, constant in the last δ_{err} epochs,

varying with the purposes of experiments.

4 Results

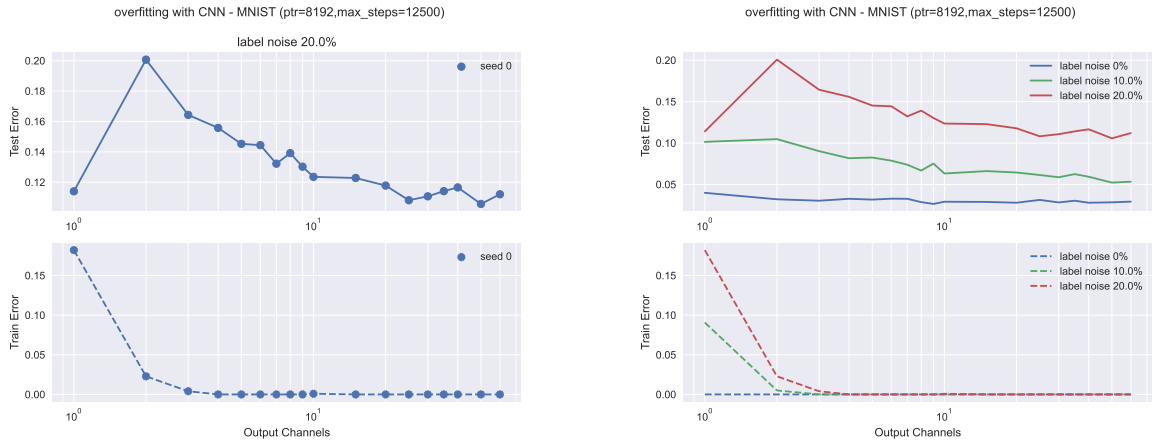
4.1 Overfit and double descent

As a first investigation on the behavior of our two models, we characterize how the classification performances evolves with increasing complexity. When training a model with too low complexity, the amount of trainable parameters involved ins't sufficient to fit the minimum features needed to fully resolve the classification task, we are in underfitting regime. At some point in complexity, it is possible to bring down the train error to zero, the model is interpolating the training set and enters the overfitting regime. At this point, more trainable parameters allows the model to learn non relevant features of the data like noise, which introduces variability and worsen it's performances (the test error augments). Although, a classically encountered behavior in deep learning is the new improvement of the performance when further overfitting, featured by a second descent of the test error, it is the double descent [5]. In fact, current State Of The Art networks usually performs in overparametrized regime.

The peak of interpolation is reduced by regularization or early stopping (as it can be seen in supplementary results A.1). Following [6], it is enhanced by label noising: "for model-sizes at the interpolation threshold, there is effectively only one model that fits the train data and this interpolating model is very sensitive to noise" in the train set and/or model miss-specification. That is, since the model is just barely able to fit the train data, forcing it to fit even slightly-noisy or miss-specified labels will destroy its global structure, and result in high test error". We then introduce label noising in our training to observe double descent behavior of our models. Label noise of probability p refers to training on a samples which have the correct label with probability $(1 - p)$.

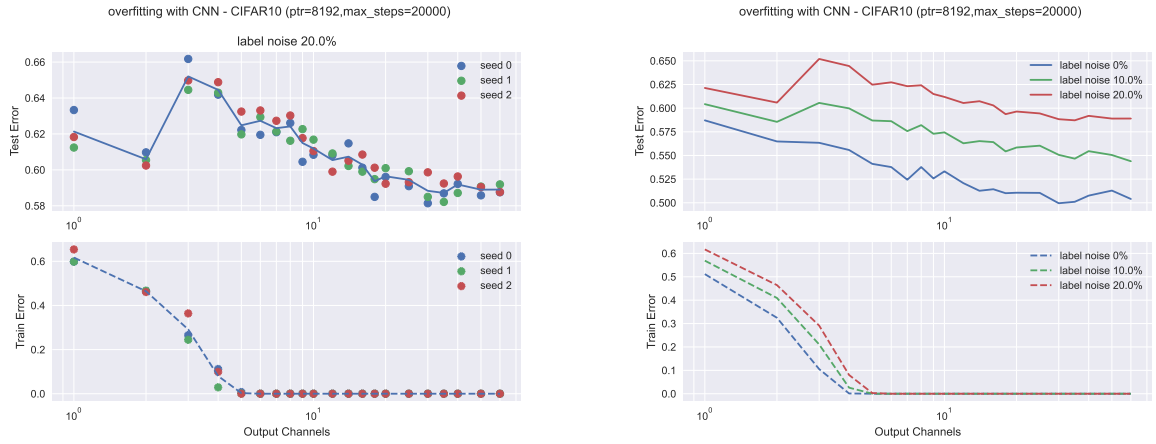
4.1.1 CNN

We then train the CNN model on MNIST and CIFAR10 with increasing output channels, with label noise 0%, 10% and 20%, and over three different initialization seeds to average fluctuations. Results for the train and test error as a function of the complexity are presented in figure 8 for MNIST and figure 9 for CIFAR10. The simple dataset MNIST is almost immediately fitted by the CNN. On CIFAR10, the CNN model shows a clear double descent with even increasing performances far enough in the over parametrization regime. Finally, we also find the number of output channels allowing overfitting on each dataset ($h = C_1 \approx 2$ for MNIST and $h = C_1 \approx 6$ for CIFAR10), useful for the next experiments.



(a) (b)

Figure 8: (a) Test error and train error as a function of the number of output channels for CNN trained on MNIST with 20% label noise (here only one initial seed). (b)Test error and train error as a function of the number of output channels for CNN trained on MNIST with different label noising. (log scale in abscissa is for visibility.)



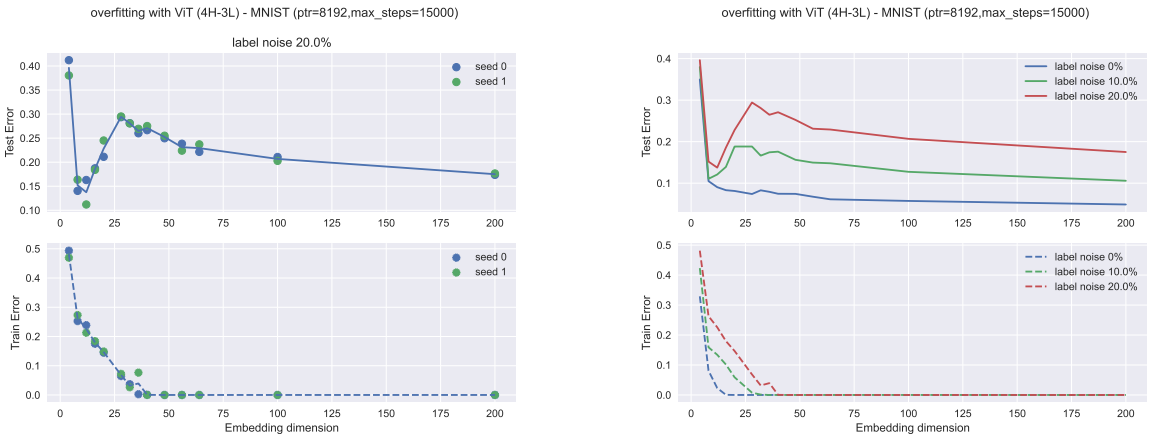
(a) (b)

Figure 9: (a) Mean over different seeds of test error and train error as a function of the number of output channels for CNN trained on CIFAR10 with 20% label noise. (b)Mean over different seeds of test error and train error as a function of the number of output channels for CNN trained on CIFAR10 with different label noising. (log scale in abscissa is for visibility.)

4.1.2 ViT

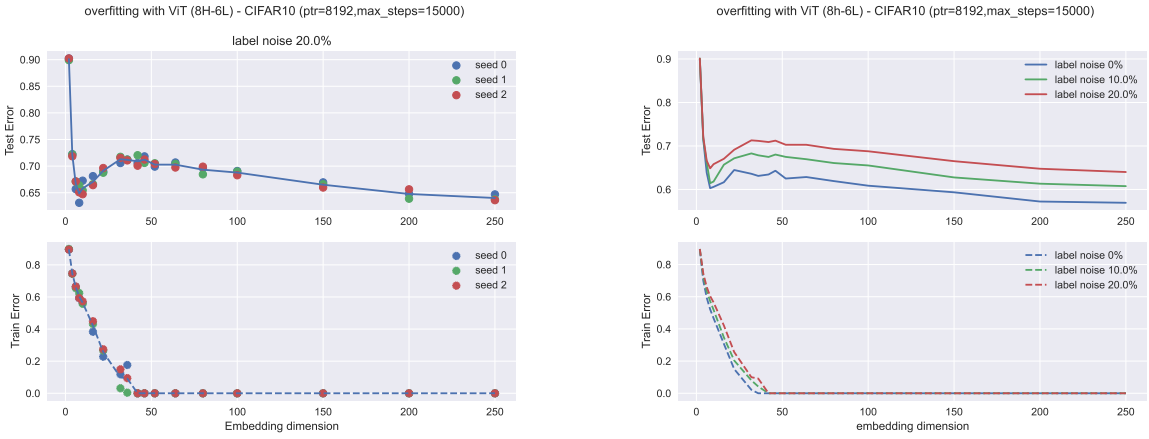
In the same way, we plot in figures 10 and 11 the train and test errors for ViT as a function of the embedding dimension. The choices of the number of heads or layer are made to reach overfitting in a reasonable amount of computation time in the range of embedding

dimension studied. Again, we observe the double descent for the two datasets, with a less marked peaked and less good improvement of performances in overfitting regime, maybe due to weight decay regularization introduced by training with AdamW algorithm. We reach a clear overfit for the ViT (8 heads, 6 layers) on CIFAR10 at embedding dimension $h_{embed} \approx 45$, and for the ViT (4 heads, 3 layers) on MNIST at $h_{embed} \approx 30$. Of course, these values will change with the parameters of the ViT.



(a) (b)

Figure 10: (a) Mean over different seeds of test error and train error as a function of the embedding dimension for ViT trained on MNIST with 20% label noise. (b)Mean over different seeds of test error and train error as a function of the embedding dimension for ViT trained on MNIST with different label noising.



(a) (b)

Figure 11: (a) Mean over different seeds of test error and train error as a function of the embedding dimension for ViT trained on CIFAR10 with 20% label noise. (b)Mean over different seeds of test error and train error as a function of the embedding dimension for ViT trained on CIFAR10 with different label noising.

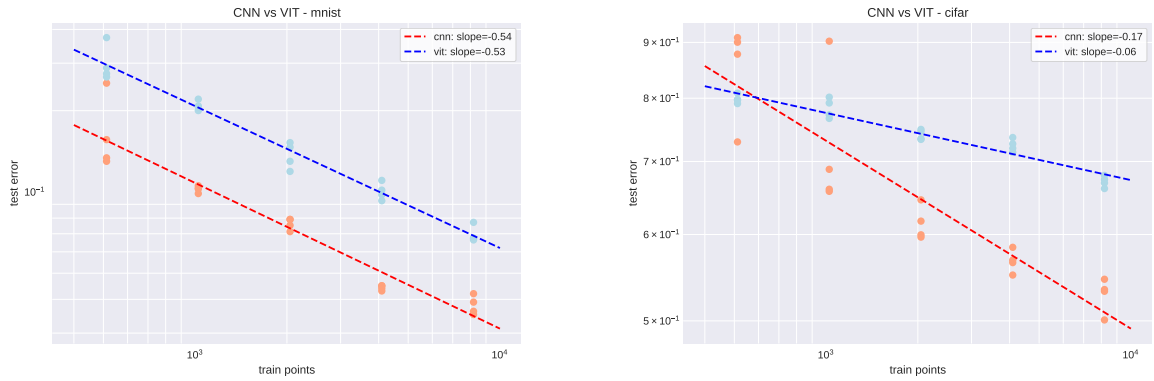
4.2 Learning curves

We now want to experimentally compare the performances in learning of the Convolutional Neural Network and the Vision transformer. We then look at how the performances improve when varying the size of the training set. In the overfitting regime (which we found earlier), it is possible to observe a scaling of the test error with the number of training points:

$$\epsilon_{test} \sim p_{train}^{\alpha}. \tag{6}$$

4.2.1 Comparing basic CNN and ViT blocks

Starting with our basic model blocks, one-layer one-head ViT, we plot the learning curve by fitting the test error as a function of the training points in log-log scale (again averaging over various seeds), results are presented in figure 12. On MNIST, the two models exhibits similar learning curves ($\alpha \approx -0.53$) and good performances, but with CNN always at lower test error. On CIFAR10, CNN exhibits a clearly faster improvement, and performs better than ViT overall already around $p_{train} = 700$ training points. Although, the performances of the two architectures with these parameters on CIFAR10 are bad ($\epsilon_{test}^{CNN} = 52.8\%$ and $\epsilon_{test}^{ViT} = 67.1\%$ for $p_{train} = 8192$) compared to state of the art results ($\epsilon_{test}^{CNN} = 4.5\%$ [7] and $\epsilon_{test}^{ViT} = 0.05\%$ [8]), already suggesting that they are too simple for this dataset.



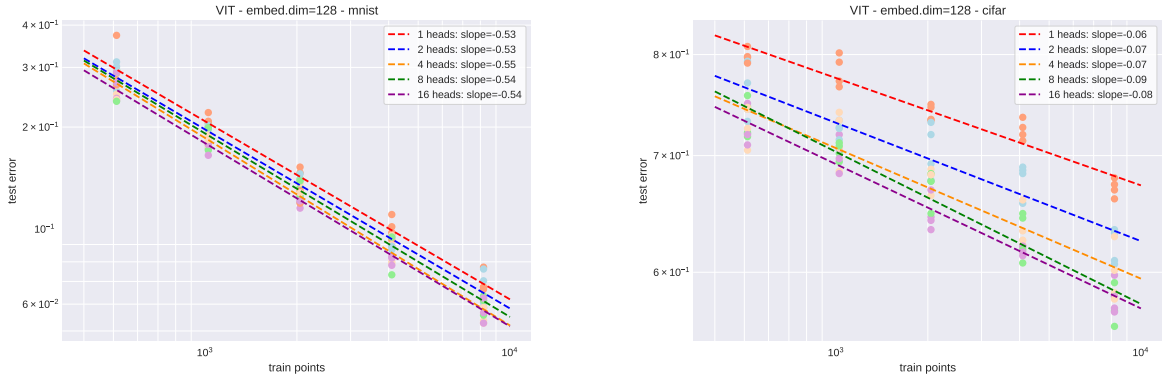
(a) (b)

Figure 12: Learning curve fitted from the mean of test error over four different seeds as a function of the number of training points. Comparison between CNN ($h = C_1 = 10$) and ViT($h_{embed} = 128$, $H = 1$, $L = 1$) on (a) MNIST and (b) CIFAR10. The stopping criterion is convergence with $\delta_{err} = 20$.

4.2.2 Changing the number of heads of ViT

A first variation we can do on the transformer architecture is changing the number of heads. On the two datasets, we then plot the learning curves for ViT at embedding dimension $h_{embed} = 128$, with varying heads from 1 to 16 in figure 13. Even if it is less clear on MNIST, in both datasets, increasing the number of heads improve performances (allows lower α) up

to a certain point. The optimal number of heads for the one-layer Vision Transformer is then found to be $H = 4$ on MNIST and $H = 8$ on CIFAR10.

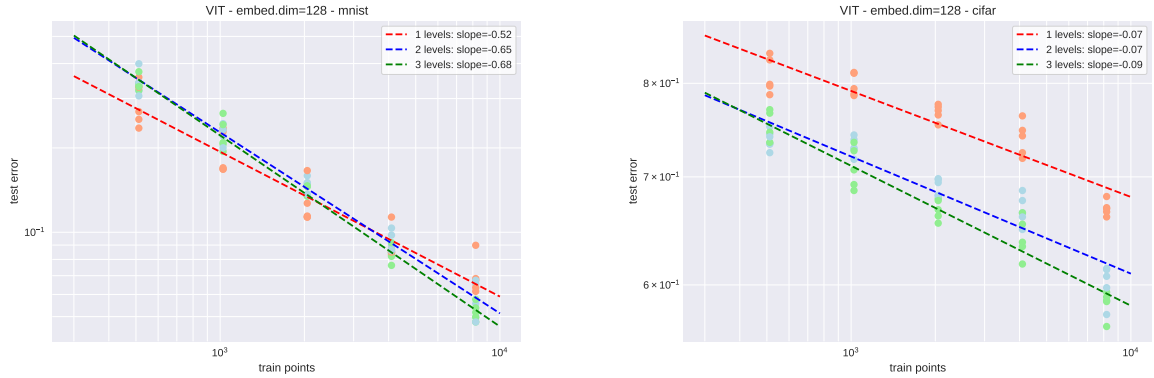


(a) (b)

Figure 13: Learning curve fitted from the mean of test error over four different seeds as a function of the number of training points. Comparison between different numbers of heads for ViT($h_{embed} = 128$, $L = 1$) on (a) MNIST and (b) CIFAR10. ($\delta_{err} = 20.$)

4.2.3 Changing the depth of ViT

The second parameter for Vision transformer is the depth (the number of layers). In the same way, we compare learning curves for a one-head ViT with $h_{embed} = 120$ for various number of layers in figure 14. On MNIST, increasing the number of layers always improves the slope of the learning curve (then performances at large training sets), but doesn't decrease test error at specific number of training points (the point at which overall performances gets better for deeper model is around $p_{train} = 3000$). In the case of CIFAR10, increasing the depth first only decreases overall test error (1 to 2 layers), and then decreases the learning curve slope.

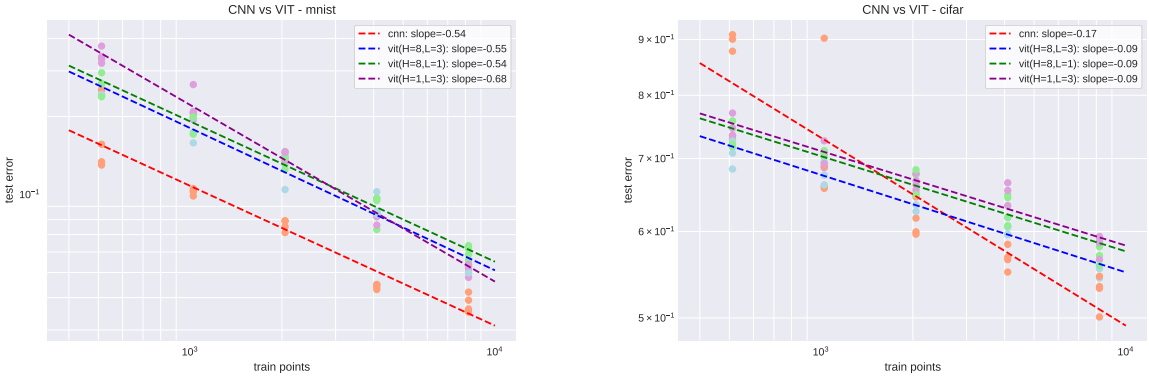


(a) (b)

Figure 14: Learning curve fitted from the mean of test error over four different seeds as a function of the number of training points. Comparison between different numbers of layers for ViT($h_{embed} = 128$, $H = 1$) on (a) MNIST and (b) CIFAR10. ($\delta_{err} = 20.$)

4.2.4 Changing number of heads and layers together

Finally, changing both the number of heads H and of layers L, we can try to reach an optimal configuration of ViT for a given dataset. From the previous results, learning curves of configurations ($H = 1$, $L = 3$), ($H = 8$, $L = 1$) and ($H = 8$, $L = 3$) are investigated on CIFAR10 in figure 15b. They show similar learning curve slope and optimal overall performances for the most complex ($H = 8$, $L = 3$). ViT offers then better performances than CNN up to $p_{train} = 3000$ training points. As we saw before on MNIST (figure 15a), optimal configurations lead to a better learning slope α for ViT than CNN. One could expect the optimal ViT to show better performances than CNN at some point p_{train} further than the studied number of training points.



(a) (b)

Figure 15: Learning curve fitted from the mean of test error over four different seeds as a function of the number of training points. Comparison between CNN($h = C_1 = 10$) and differently optimized ViT on (a) MNIST and (b) CIFAR10. The stopping criterion is convergence with $\delta_{err} = 20$.

5 Discussion

Even if the previous results aren't exhaustive enough to conclude on a clear comparison between Convolutional Neural Networks and Vision Transformer, they clearly exhibit the non trivial learning behaviors of the models, depending on both the complexity of the architecture and of the dataset, but not only. We saw for example that augmenting the number of heads doesn't improve linearly the performances, and that the type of improvement is different for the two datasets. Of course, even in their basic block form, the two models don't handle the same way the complexity of the datasets. The simplicity of MNIST may not allow see all improvements a different architecture can bring, and the complexity of CIFAR10 for our basic blocks may influence differently the models.

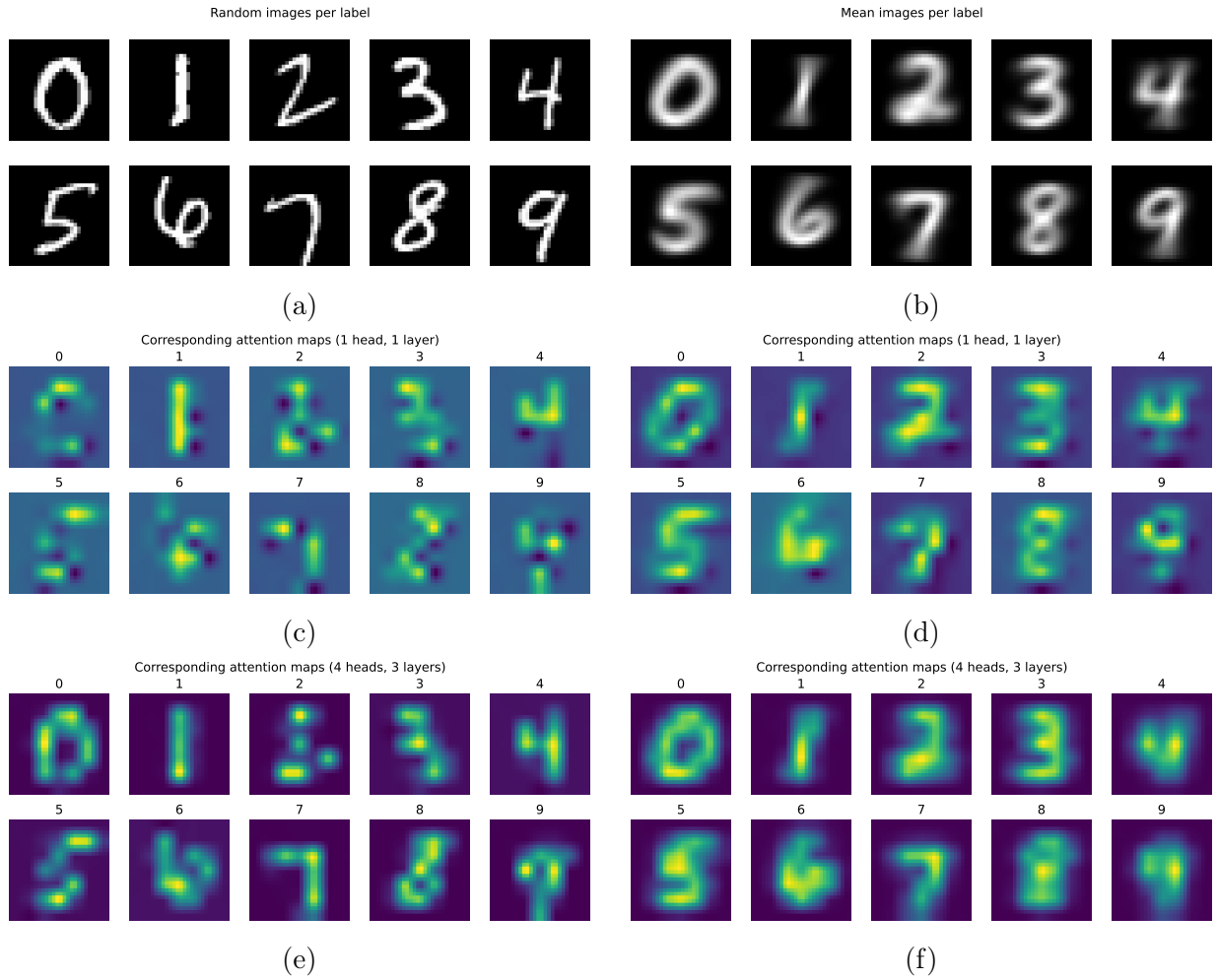
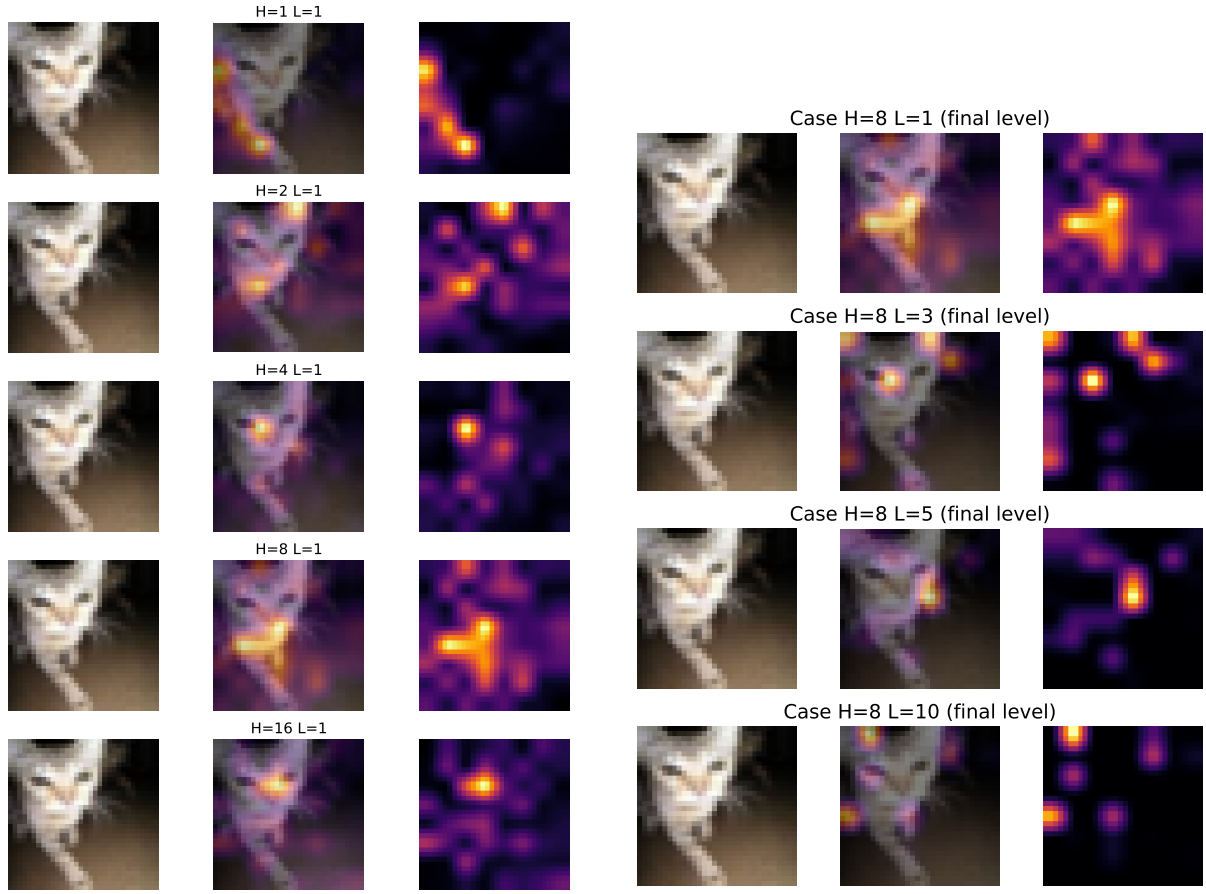


Figure 16: (a)(c)(e) Attention maps on a random instance of each class of MNIST for Vit models wirh ($H = 1$, $L = 1$) and ($H = 4$,L=3). (b)(d)(f) Attention maps on the mean over the instances of each class of MNIST for Vit models wirh ($H = 1$, $L = 1$) and ($H = 4$, $L = 3$).

To illustrate and visualize the difference in performances of the ViT models, we can recover the self attention weights before the linear feed-foreward, and plot the attention maps. In figure 16, we then plot attention maps for one instance of each class of MNIST dataset fed to a trained ViT model with either 1-head 1-layer, or 4-heads 3-layers (optimally chosen following the learning curve experiment). Although the basic 1-layer 1-head model already performs well on MNIST, we can see that the optimized architecture exhibits better defined features, clearly identified numbers and background.

In the same way, in figure 17 we plot the attention maps for varying heads with one layer, or varying layers for one head (other examples are put in supplementary results A.2). We coherently observe that well performing architectures are associated with better concentration of the attention on the target of the image.

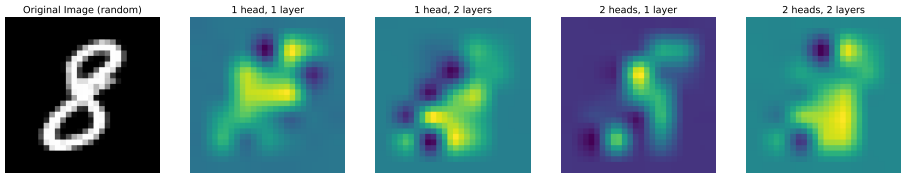


(a) (b)

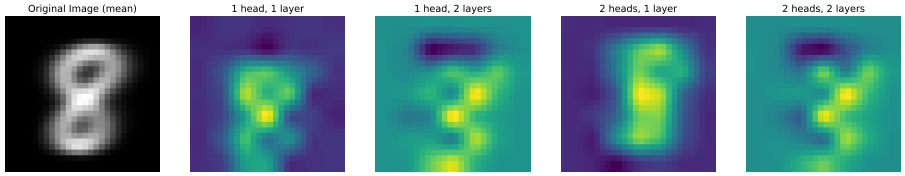
Figure 17: Attention map of on one instance of CIFAR10 for ViT models with (a) $L = 1$ and varying heads, (b) $H = 8$ and varying depth.

Originally, our building blocks of models are created to span entirely the image in the two cases. But of course in reality, they treat the data differently: the CNN model has a built in hierarchy, while only one head of the ViT has attention on all the features of the image at once. One can imagine that varying the number of head and the depth of ViT allows introducing separability and hierarchy in the treatment of the data. A qualitative idea to see it would be to exhibit simple enough features in the attention maps and vary on few parameters.

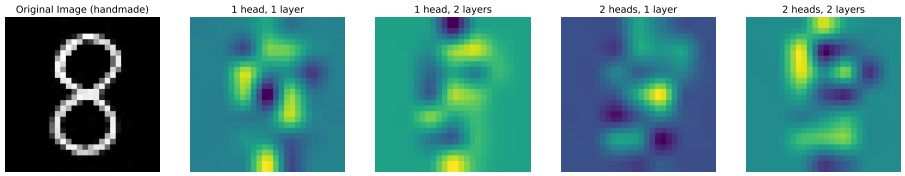
In that sense, an attempt on MNIST dataset is made in the simple case of ViT with one or two layers and heads. Imagining that the classification is a learning based on differences, we try to see the concentration of attention only on a few points of the image by classifying with only two classes (for example classes 5 and 8 with differences located in the lower left and higher right parts of the anti-diagonal). Resulting attention maps on an instance of the class 8, the mean of all images of the class, and a handmade typical image are presented in figure 18. These results don't allow to conclude on a clear interpretation. With more time, we could try to observe separability or hierarchy by investigating attention weights on individual heads and layers.



(a)



(b)



(c)

Figure 18: Attention maps on (a) one random instance of class 8 on MNIST, (b) mean of instances of class 8 of MNIST and (c) a handmade image typical of class 8 of MNIST; for ViT trained on two classes of MNIST (classes 5 and 8) with varying number of layers and heads.

6 Conclusion

To conclude, even if the presented experiments aren't exhaustive enough to conclude on established differences between Convolutional Neural Network and Vision Transformer, they allowed us to characterize the basic functionalities of some parameters of the architectures. With increasing complexity, the two models exhibited double descent with overfitting. In the range of parameters studied, changing the number of heads and layers of the ViT allowed to find optimized configurations for the studied datasets, and quantitatively compare performances with the CNN. These differences in performances with the parameters of the ViT were visualized through the attention maps. In comparison with the intrinsic characteristics of the basic CNN block for treating image data, we can then imagine that the number of heads and depth of the ViT can introduce separability in hierarchy in the learning, enabling it to approach the performance of the CNN for optimized parameters.

References

[1] *Machine learning for physicists [PHYS-467]*, Lecture Notes, Prof. Lenka Zdeborova, EPFL, January 2022

[2] *CS502: Deep Learning in Biomedicine*, Maria Brbić, EPFL, Fall 2023

[3] *Tutorial 6:Transformers and Multi-Head Attention*, UvA Deep Learning notebooks, https://uvadlc-notebooks.readthedocs.io/en/latest/tutorial_notebooks/tutorial6/Transformers_and_MHAttention.html

[4] *Attention Is All You Need*, Ashish Vaswani et al., arXiv, August 2023

[5] *A jamming transition from under- to over-parametrization affects generalization in deep learning*, S Spigler et al 2019 J. Phys. A: Math. Theor. 52 474001

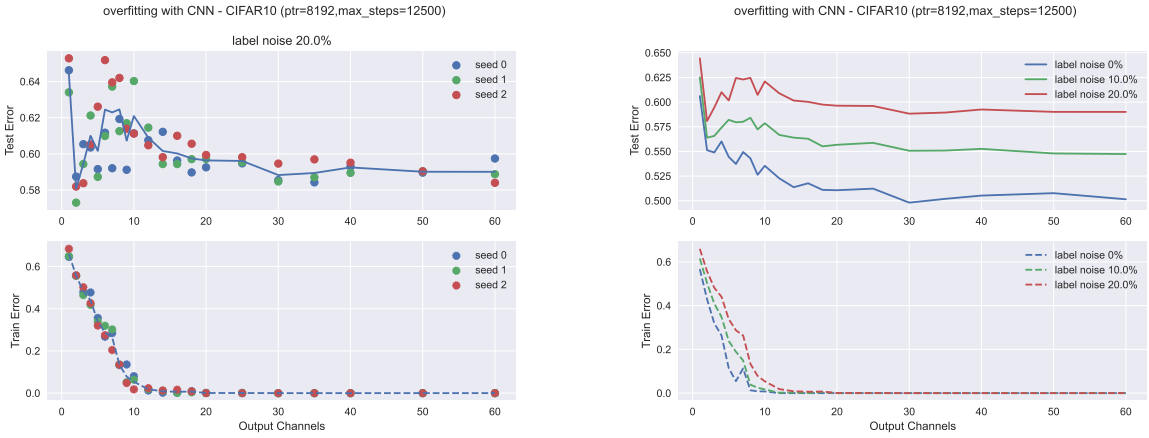
[6] *Deep Double Descent: Where Bigger Models and More Data Hurt*, Preetum Nakkiran et al., ICLR 2020

[7] *Densely Connected Convolutional Networks*, Gao Huang et al., CVPR 2017

[8] *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*, Alexey Dosovitskiy et al., ICLR 2021

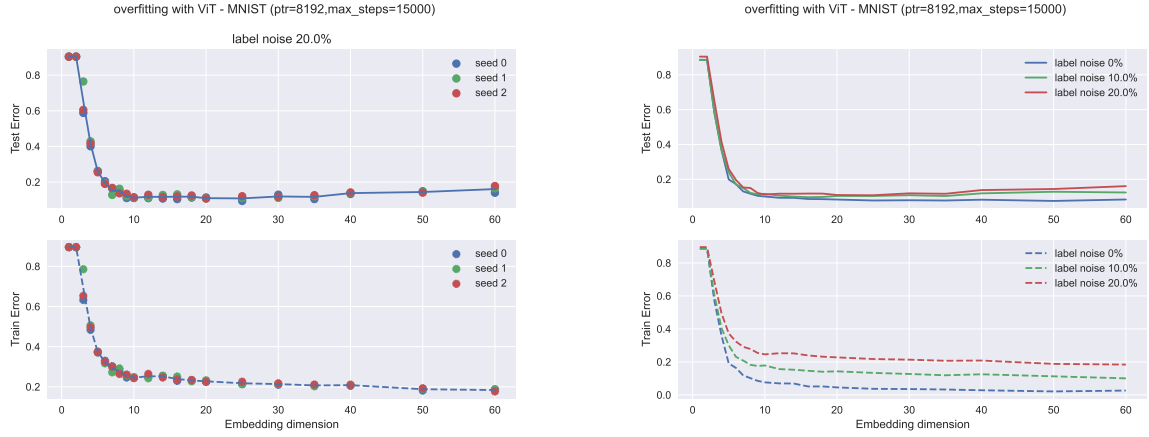
A Supplementary results

A.1 Double descent



(a) (b)

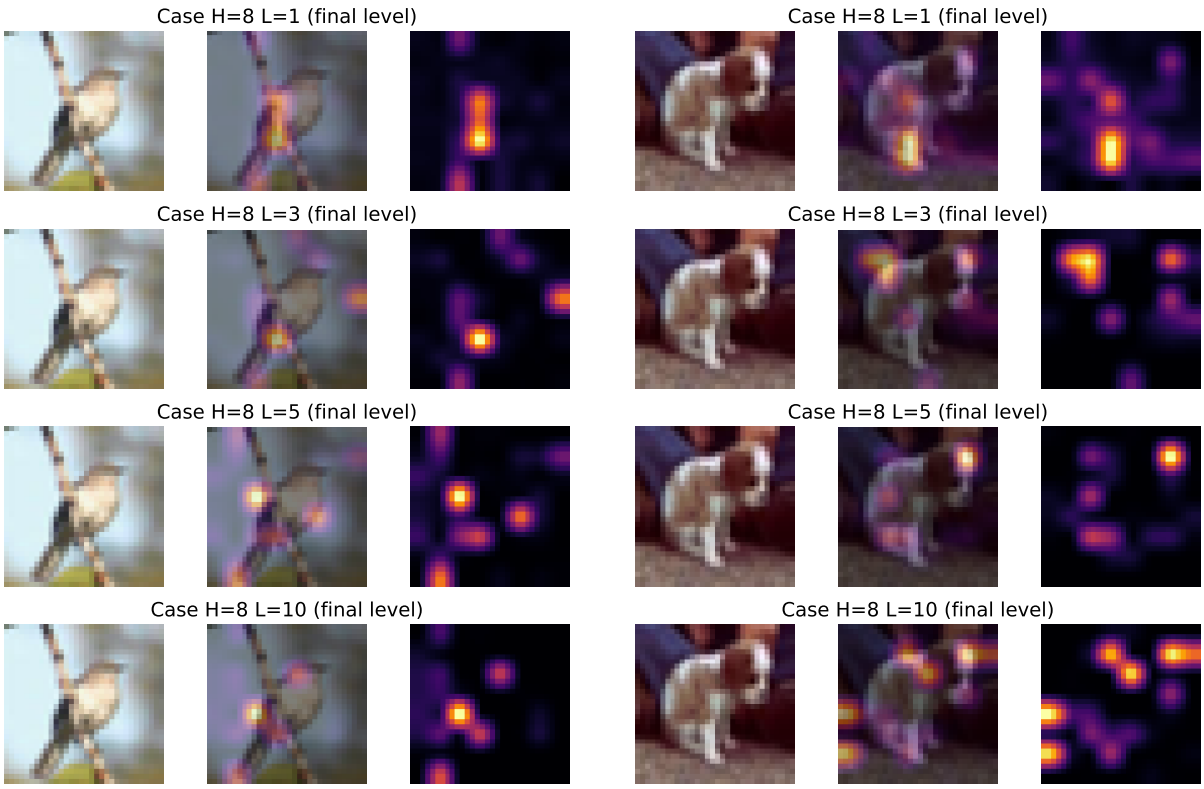
Figure 19: (a) Mean over different seeds of test error and train error as a function of the number of output channels for CNN trained on CIFAR10 with 20% label noise. (b)Mean over different seeds of test error and train error as a function of the number of output channels for CNN trained on CIFAR10 with different label noising.



(a) (b)

Figure 20: (a) Mean over different seeds of test error and train error as a function of the number of output channels for ViT ($H = 1$,$L = 1$) trained on MNIST with 20% label noise. (b)Mean over different seeds of test error and train error as a function of the number of output channels for ViT ($H = 1$,$L = 1$) trained on MNIST with different label noising.

A.2 Attention maps



(a) (b)

Figure 21: Attention map on one instance of classes (a) "bird" and (b) "dog" of CIFAR10 for ViT models with $H = 8$ and varying depth.