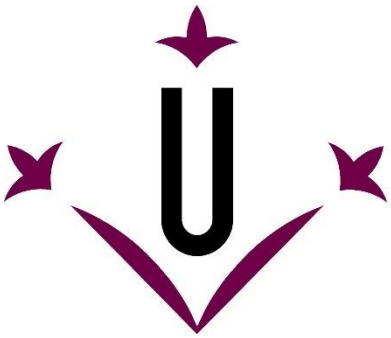


Universitat de Lleida
Escola Politècnica Superior

Degree in Computer Engineering

Course:

Advanced Programming in Artificial Intelligence

Machine learning project:

Vision Transformer for Image Classification

Authors:

Camara, Florin

Taló López, Pau

Professor:

Planes Cid, Jordi

day 31 of May of 2021

Year: 2020 – 2021

INDEX

Introduction

The main goal of this project is to compare the performance of a traditional Convolutional Neural Networks (CNN) with the Vision Transformer (ViT) model for image classification.

To do so, we decided to create a Pokémon image classifier using a dataset from Kaggle named: “7000 Labelled Pokémon” which we divided into Train and Test folders.

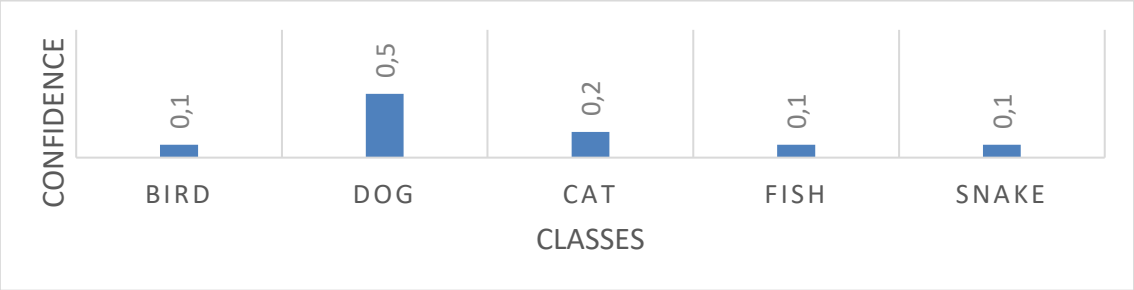
From this dataset, we took an already implemented Pokémon classifier using a CNN model and modified it to use a ViT model to finally compare their performance.

What is image classification?

If we show this image to a model, the model should infer that this image contains a dog.

Feed this image into a neural network and it will output a vector where each position represents the probability of the image to belong to a certain class.

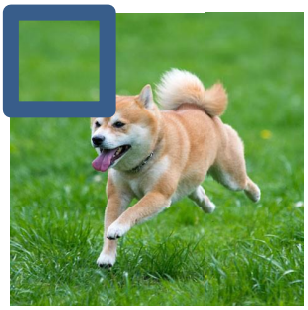
This probability is called confidence, it is a number between 0 and 1, and they all add up to 1. The result provided by the network will be the class with the highest confidence.



How does a ViT work?

Split the image into patches

ViT requires an image to be split into patches of the same shape, these patches can overlap.

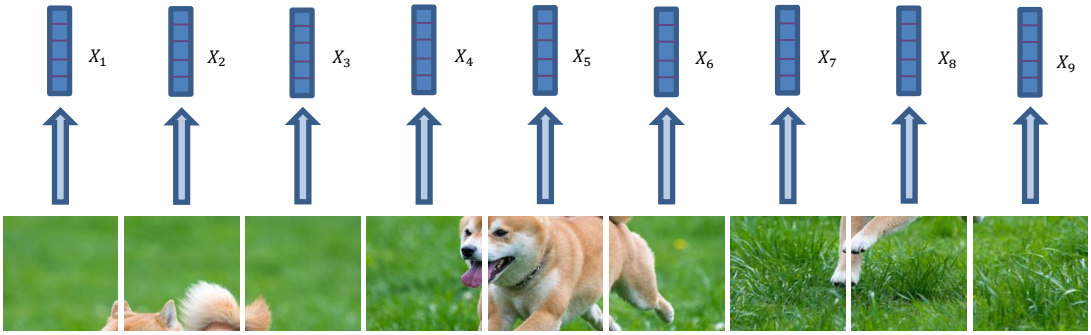


When splitting the image, we use a sliding window that moves several pixels each time, the number of pixels that it moves is called stride. A smaller stride will result in a bigger number of patches, which means very heavy computation.

Vectorization

Suppose the image is split in 9 patches, every patch is a small image with RGB channels, and it is an order 3 tensor (A tensor is a generalization of vectors and matrices and is easily understood as a multidimensional array).

The next step is to vectorize the patches, which means reshaping the tensors into vectors.



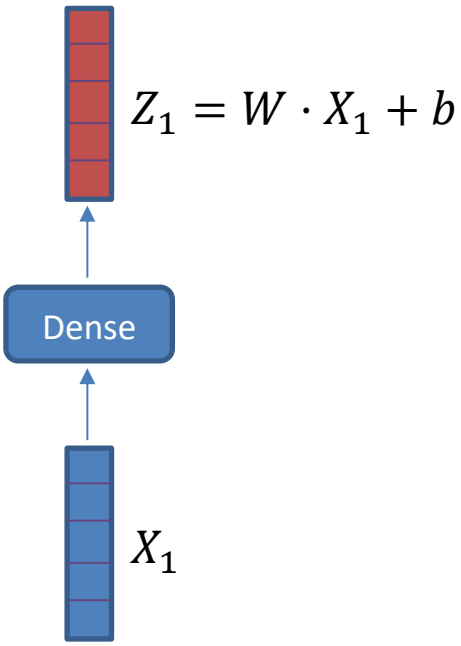
Neural Network

Once we have all the vectors, we apply a dense layer to each one of them.

A dense layer is a neural network layer that is deeply connected, which means each neuron in the dense layer receives input from all neurons of its previous layer.

This layer outputs an “m” dimensional vector by performing a matrix-vector multiplication, in this case W and b. W and b parameters are learned from training.

The dense layers share parameters W and b.

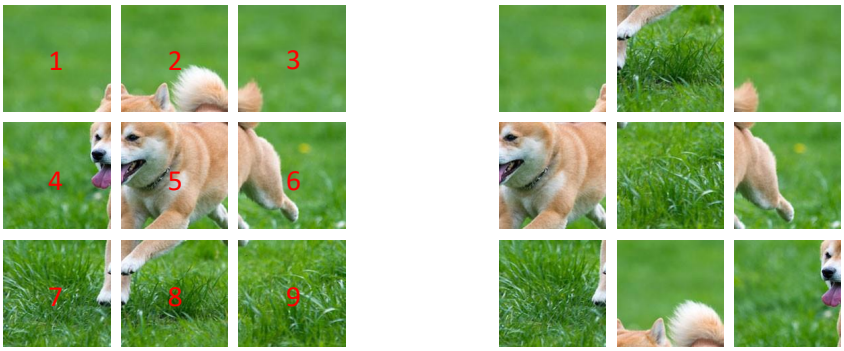


Positional Encoding

In addition, we add positional encoding to all the vectors, so now a z vector contains both the content and the position of a patch.

The ViT paper demonstrated that without positional encoding, accuracy drops by 3% and it works with any kind of positional encoding.

The reason why we add this type of encoding can be seen in the next example:



As we can see, we have two images that are the same but the patches are ordered differently.

If the ViT doesn't contain positional encoding, it will say that they are two different images, however if it does contain positional encoding it will say that they are the same.

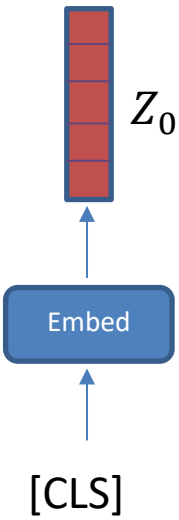
Collective Learning System Token

Aside from the patches we also use the Collective Learning Systems (CLS) token for classification.

Using CLS, we propose a hierarchy of collective learning layers to learn colour and texture feature patterns of images to perform three basic tasks: recognition, classification and segmentation.

An embedding layer (a layer that represents the input using a dense vector representation) takes this token as input and outputs vector Z0, this vector has the same shape as the other Z vectors.

We use the CLS token because the output of the transformer in this position will be used for the classification.



Multi-Head Self-Attention Layer

All Z vectors are fed to a Multi-Head Self-Attention layer, this layer is a sequence of n plus 1 vectors. The term multi-head is just doing the same thing by multiple heads, in this case a self-attention layer.

But what is self-Attention?

Say the following sentence is an input sentence we want to translate:

” The animal didn't cross the street because it was too tired”

What does “it” in this sentence refer to? Is it referring to the street or to the animal? It’s a simple question to a human, but not as simple to an algorithm.

When the model is processing the word “it”, self-attention allows it to associate “it” with “animal”.

As the model processes each word (each position in the input sequence), self-attention allows it to look at other positions in the input sequence for clues that can help lead to a better encoding for this word.

Transformer Encoder Network

The dense and multi-head self-attention layers can be combined as many times as you want, one by one, forming the Transformer Encoder Network. Apart from the layers, transformers also use other tricks for improving its performance like skip connection and normalization.

The outputs of this encoder network are vectors C, to perform the classification task it will only use vector C0, it is the feature vector extracted from the image.

This C0 vector is fed to a SoftMax Classifier.

The SoftMax classifier gets its name from the SoftMax function, which is a mathematical function that converts a vector of numbers into a vector of probabilities, where the probabilities of each value are proportional to the relative scale of each value in the vector.

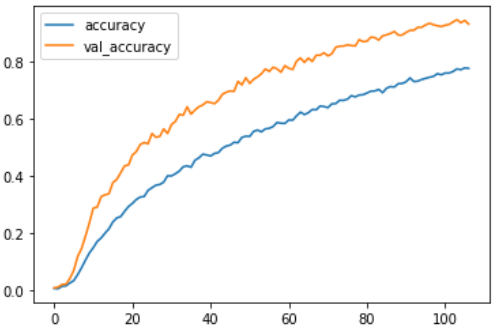
This classifier outputs the vector p, which contains the result of the classification. Each position represents a possible class from the classification. The value of each position is the confidence with which the transformer says that the input image belongs to that class.

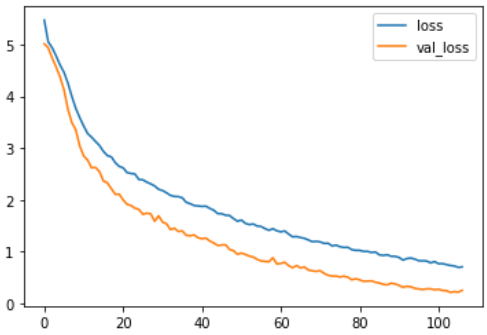
The class with the highest confidence value will be the one outputted as a result.

Results

After training the model with only our Pokémon dataset, we can see that it has achieved much better accuracy than the CNN version from Kaggle, that being 93% against as much as 85% with increased stopping patience value.

As we can see in the following plots, our model might actually be able to achieve an even higher value since we believe that the algorithm stopped training because we use early stopping as call back function with patience = 3 and while training it might have found a local minimum, making the hole algorithm stop.





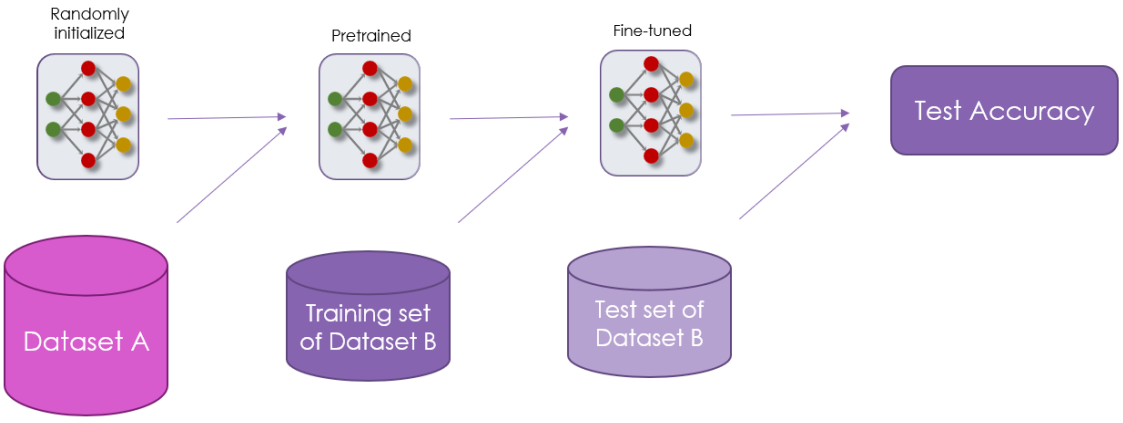
If we had set a bigger patience value, it might have gotten out of that minimum and kept improving the accuracy.

Conclusions

Our ViT version of the Pokémon classifier has better accuracy with fewer epochs than the CNN version from Kaggle which for the purpose of this project is enough.

But if you read the paper for vision transformers for image recognition, you will see some interesting notes on this model's performance against ResNet.

First of all, to improve ViT's performance it should be first pretrained on a large dataset, then it should do fine-tunning on the training set of the target database and finally test its accuracy on the test set of this same database.



Unlike ResNet, the larger the pretraining dataset is, the better the ViT performs.

Comparing ResNet with ViT, when the pretraining dataset is smaller than 100M images, ViT is worse than ResNet while for datasets bigger than 100M images, ViT performs better.

In the paper it is shown that ResNet performs better than ViT when it is pretrained with ImageNet (1.3M images) and ImageNet-21k (14M images) datasets, but not for JFT (300M images).

Dataset	# of Images	# of Classes
ImageNet	1,3 Million	1 Thousand
ImageNet-21K	14 Million	21 Thousand
JFT	300 Million	18 Thousand

So, to conclude, **Vision Transformer is the new state-of-the-art for image classification** as long as it is pretrained on a sufficiently large dataset.

Bibliography

Image Classification with Vision Transformer:
https://keras.io/examples/vision/image_classification_with_vision_transformer/

Vision Transformer for Image Classification - Shusen Wang:
https://www.youtube.com/watch?v=HZ4j_U3FC94

An image is worth 16x16 words: Transformers for image recognition at scale - Alexey Dosovitskiy:
https://arxiv.org/pdf/2010.11929.pdf

Vision Transformer (ViT):
https://huggingface.co/transformers/model_doc/vit.html#:~:text=A%20%5BCLS%5D%20token%20is%20added,can%20be%20used%20for%20classification.&text=As%20the%20Vision%20Transformer%20expects,normalize%20images%20for%20the%20model.

7,000 Labeled Pokemon - Lance Zhang:
https://www.kaggle.com/lantian773030/pokemonclassification

Pokemon Image Classifier - Dharmin Shah:
https://www.kaggle.com/dharminshah/pokemonclassifier/notebook

A CLS Hierarchy for the Classification of Images:
https://link.springer.com/chapter/10.1007/11579427_37#:~:text=The%20adaptive%20algorithm%20used%20in,%3A%20recognition%2C%20classification%20and%20segmentation.

The Illustrated Transformer - Jay Alammar:
https://jalammar.github.io/illustrated-transformer/

SoftMax Activation Function with Python:
https://machinelearningmastery.com/softmax-activation-function-with-python/