Assignment – Choice 1 – Visual Media

An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale

I chose to reimplement the paper "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". It is important for several reasons, and its acceptance to the conference is due to its technical contributions and implications in the field of computer vision.

The paper introduces the Vision Transformer (ViT) architecture, which applies the Transformer architecture directly to images for image recognition tasks. This is different to traditional architectures for image recognition which use convolutional neural networks (CNNs). So therefore, this is a significant change and something new in the field. ViT demonstrates that Transformers, originally designed for natural language processing (NLP), can achieve competitive performance on image recognition tasks without using convolutional layers.

ViT offers scalability and flexibility, allowing it to handle large-scale datasets such as ImageNet effectively. By treating images as sequences of patches and applying self-attention mechanisms, it can capture long-range dependencies in images. This approach makes it scaleable to larger datasets and achieve state-of-the-art results without requiring complex architectural modifications. This is demonstrated on several image classification benchmarks, including CIFAR-10, CIFAR-100, and ImageNet. This proves the effectiveness of the Transformer architecture in handling visual data and its potential to surpass the performance of traditional CNN-based approaches.

Compared to traditional architecture this simplifies image classification which can be an advantage for other tasks.

In summary, the paper is important because it introduces a new approach to image recognition using Transformers, demonstrating state-of-the-art performance, scalability, and flexibility. Its technical contributions have led to advancements in the field of computer vision and sparked further research into the application of Transformers to various visual tasks.

My implementation

Once I found this paper I started with the reimplementation of the paper. Initially I wasn't aware of resource intensiveness of the whole thing, however that was a problem with most Computer Vision papers, so I decided to stick to this paper and try my best with the limited resources I had on Google Collaboratory. First, I had to decide on a Dataset

that I wanted to use for the implementation. I ended up going for the CIFAR10 dataset as it was described in the paper as well. It was the smallest one I from the paper with 40000 training dataset size and 10000 evaluation and test set size.

Then I started to reimplement what the paper was describing. I used the Neural Network module of torch library to define my transformer. Firstly, I would have to split up the pictures into the patches. These patches are like small squares that we can focus on individually. For that I used a Conv2d embedding layer, however the Conv2d layer in the patch embedding step of the ViT model does not perform typical convolutional operations for feature extraction as in CNNs. Instead, it is only used to for the patch embedding.

Once we have our patches, we need to understand the relationships between them, so we add a positional embedding to the pipeline. Now that we have our patches and positional embeddings we can feed them to the transformer.

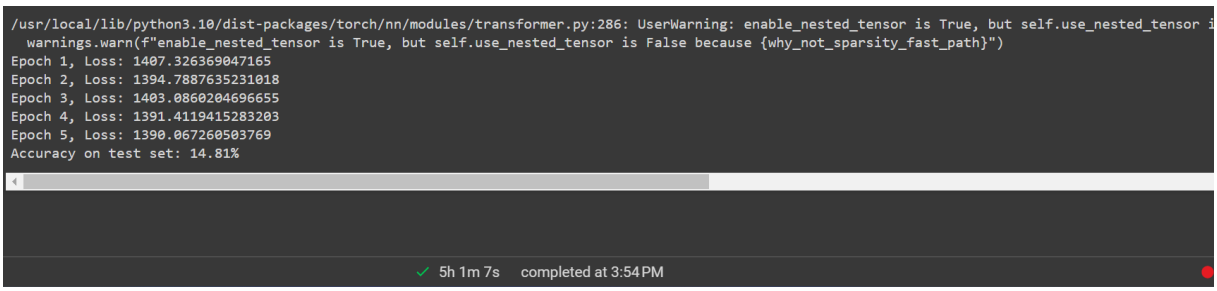
Now that the model has looked at all the patches and understood their relationships, it's time to make a decision. To do this, we add a layer called the Classification Head. It takes the information from the Transformer Encoder and gives us the final classification. Like described in the paper I use a Linear Classification head.

After setting up the model and loading the data set all that was left was to train the model. To measure how well our model is doing Cross Entropy Loss and accuracy was used. Like described in the paper we pick the Adam optimizer with a learning rate of 0.001 and weight decay.

For the training loop, we run 5 epochs where we make predictions and calculate how far off these predictions are from the actual labels using the loss function and update the model's parameters to improve its predictions using the optimizer.

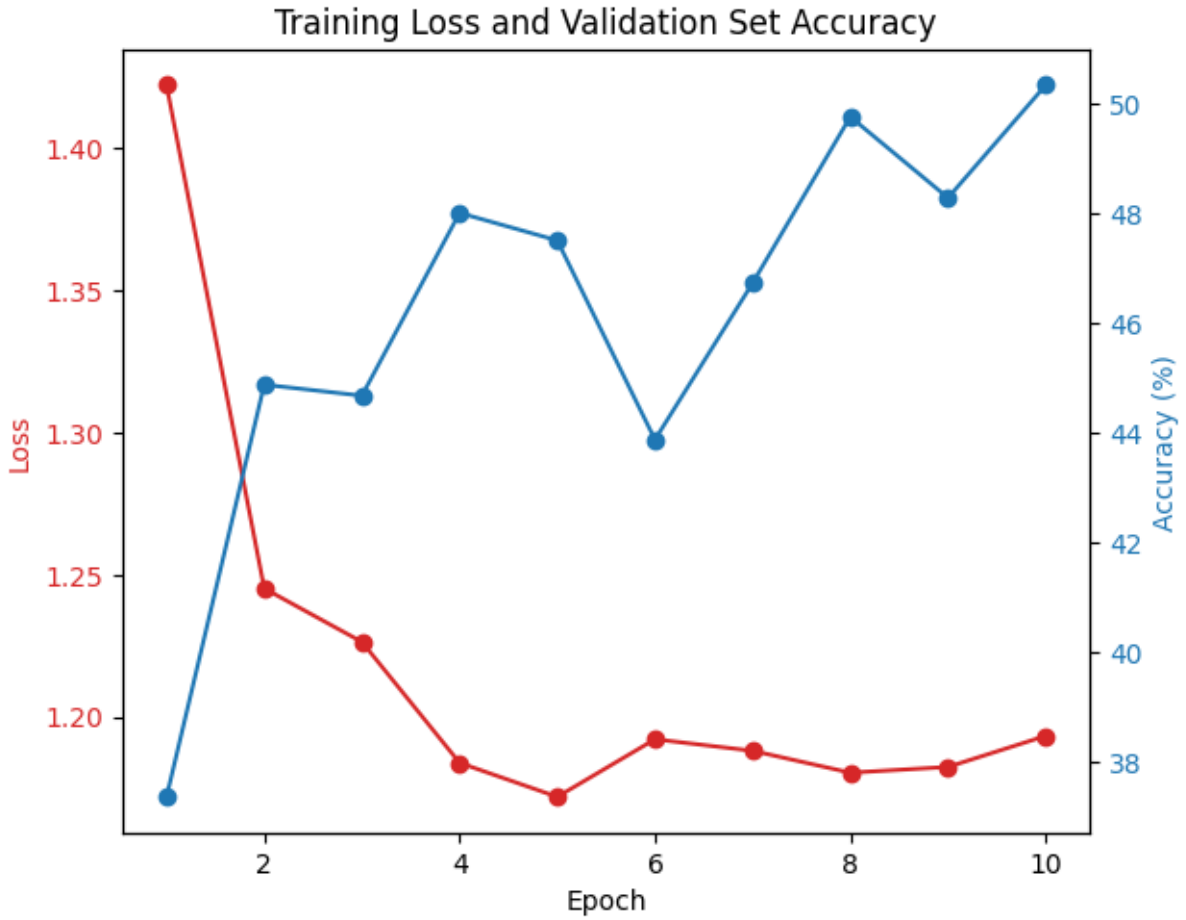
However, to be able to run the whole thing I had to make the model considerably smaller which would definitely sacrifice performance, but it was the only way to implement it without using a huge computation cluster. So I reduced the Hidden dimensions to 256, the number of layers to 4, number of attention heads to 4 and the MLP dimension to 1024.

This made it possible for me to run the model, it took 5 hours however and I only got an accuracy of 14.81% as can be seen from the picture below.



Comparing this score to a base line of random guessing that achieves 10% accuracy, we could see already that what I was trying to do was working. I also only trained for 5 epochs since there weren't any big improvements after the first 4 epochs. So while this was a proof of concept that I was happy with, I wanted to know if I could do better, but for that I obviously needed a bigger model which would be more computationally expensive.

So I decided to try to make the model size bigger going back to the ViT Base model described in the paper with 768 Hidden Dimension, 12 layers, 12 attention heads and an MLP dimensions of 3072. However, I trimmed the CIFAR10 Dataset down to only 4 classes to make the training a bit more manageable. The 4 classes I kept were 'airplane', 'automobile', 'bird' and 'cat'. This left me with a training set with size 16000 and an evaluation set, and test set with 4000 images respectively.



We can see that this boosted the performance drastically with now an accuracy of 51% on the test set, while random guessing would give us 25% accuracy. Considering the limited resources available to me I was very happy with this result.

```
Epoch 1, Loss: 1.422316777229309, Accuracy on val set: 37.375%
Epoch 2, Loss: 1.2451710357666015, Accuracy on val set: 44.875%
Epoch 3, Loss: 1.2263434090614318, Accuracy on val set: 44.675%
Epoch 4, Loss: 1.1837899525165558, Accuracy on val set: 48.0%
Epoch 5, Loss: 1.1718097343444824, Accuracy on val set: 47.5%
Epoch 6, Loss: 1.1920758292675018, Accuracy on val set: 43.875%
Epoch 7, Loss: 1.1880222277641297, Accuracy on val set: 46.725%
Epoch 8, Loss: 1.1803135380744934, Accuracy on val set: 49.75%
Epoch 9, Loss: 1.1823317251205445, Accuracy on val set: 48.275%
Epoch 10, Loss: 1.1932609915733337, Accuracy on val set: 50.325%
Accuracy on test set: 51.1%
```

Final Remarks

The core of the assignment, I think was more the reimplementation than achieving perfect results, which is not possible anyhow. I even took it a step further and tried to play around with the parameters of the model. I also tried a few things to achieve a better performance than 51% but in the end, the task was to reimplement the paper and not to change the architecture so I stuck with what I had. However, I think this also helped me to better grasp the concept the paper I tried to reimplement introduced. Which is a very interesting one. For me the paper is so interesting because it poses a fundamental question in Machine Learning, which is whether in the future multiple different architectures will be needed for different tasks or we can find a architecture that fits all use cases, although that sounds unrealistic from todays standpoint to me it is an interesting question. And I think in the broader sense this is what this paper is getting at.