Computer Vision

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

By

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, Neil Houlsby

CVPR 2020

Paper Implementation

Rahul Kumar Chaudhary
(M20MA008)

Code Link

Summary

The basic premise of this paper is that we don't need CNN to solve computer vision problems as it was thought until now. Transformers can do the same thing. Basically it shows that we don't need CNN,so we don't need recurrent networks, we can just do the attention or self attention mechanisms for our computer vision problems. This is the amazing observation shown in this paper. It achieved super results on the computer vision image classification task and less time to compute.It can achieve all the results in Big data regim, thus it needs a lot of images in order to compute the results.

Here are the major steps that were implemented in the paper in order to implement a transformer for computer vision classification tasks.

- It split the input images into small patches, and flatten these patches in rest order. These patches act as word vectors of traditional transformers.

- Linear projection of the flatten images and perform positional encoding.

- Then we feed these positional encoded patches to original transformer and putting simple MLP header (Multi-Linear Perceptron Header) to perform classification

Important Key Points

Here are the important key points of the paper “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”

- As one major advantage of the transformer over traditional Convolution Neural Network is that it can be scaled up. Thus, this model ‘Vision transformer’ has the benefit of transformer and thus can be scaled easily

- In order to achieve better results in Big data regim, it need need a lot of training samples

- During fine-tuning with Visual Transformer on higher resolution images results in bigger number of patches and thus the flatten sequence become larger, and because of that the positional encoding doesn’t make sense if we just apply them during fine tune process, instead we’ve to do 2D interpolation,

Datasets Used in the Paper & Performance

Following table shows all the different datasets used in the paper and their respective performance with the Visual Transformer model.

	JFT ViT-H/14 (This Paper)	JFT ViT-H/14 (This Paper)	JFT ViT-H/14 (This Paper)	BiT-L (ResNet152× 4
ImageNet	88.55 ± 0.04	87.76 ± 0.03	85.30 ± 0.02	87.54 ± 0.02
ImageNet ReaL	90.72 ± 0.05	90.54 ± 0.03	88.62 ± 0.05	90.54
CIFAR 10	99.50 ± 0.06	99.42 ± 0.03	99.15 ± 0.0	99.37 ± 0.06
CIFAR 100	94.55 ± 0.04	93.90 ± 0.05	93.25 ± 0.05	93.51 ± 0.08
Oxford-IIIT Pets	97.56 ± 0.03	97.32 ± 0.1	94.67 ± 0.15	96.62 ± 0.23
Oxford Flowers-102	99.68 ± 0.02	99.74 ± 0.00	99.61 ± 0.02	99.63 ± 0.03
VTAB	77.63 ± 0.23	76.28 ± 0.46	72.72 ± 0.21	76.29 ± 1.70

Table 1 : Comparison with state of the art on popular image classification benchmarks.

Code Implementation : Reproduce results on Dataset Used in the Paper

Datasets and Hyper-Parameter Details

I have used the CIFAR100 dataset to reproduce the result obtained on the paper. CIFAR10 is an image classification dataset with 100 classes . Below table provides the details of datasets CIFAR100.

Dataset Name	*CIFAR-100*
Number of Instances	*50000 Training, 10000 Test*
Associated Tasks:	*Classification*
Features	*Image size : 3*32*32*
Missing Value	*NONE*
Total Number of Classes	*100*

Table 1.1 Dataset Details

Below table provides the Hyper-parameter details used in the classification of CIFAR100 dataset.

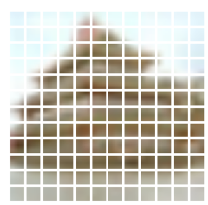
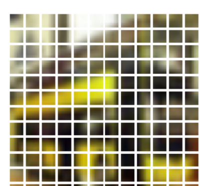
Hyperparameters	*Values*

Number of Epoch	100
Batch Size	256
Initial Learning Rate	0.001
Optimizer	Adam
Loss	Cross Entropy
Weight Decay	0.0001
Patch Size	6×6
Number of head	4
Transformer Layer	8
MLP Head Units (Size of Dense Layer)	[2048 , 1024]

Table 1.2 Hyper-Parameter Details

Results & Discussion

As mentioned in the summary of this report, Visual Transformer converts the input image into patches in order to use it as word vector to add positional encoding. Here are a few examples.

Labels	Original Image	Patches (144 Patches per images)
17		
13		

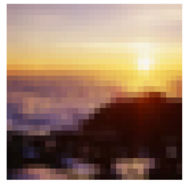
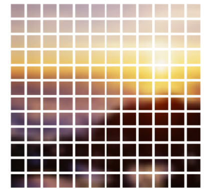
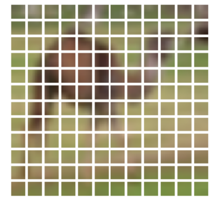
49		
15		

Table 1.3 Training Patches

Performance of the model on the dataset is given by the table below.

	Accuracy	Loss
Training	77.56%	
Validation	54.26%	
Test	55.6%	

Table 1.4 Model Performance

Plots and Graphs

Here are training and validation accuracy and loss plot

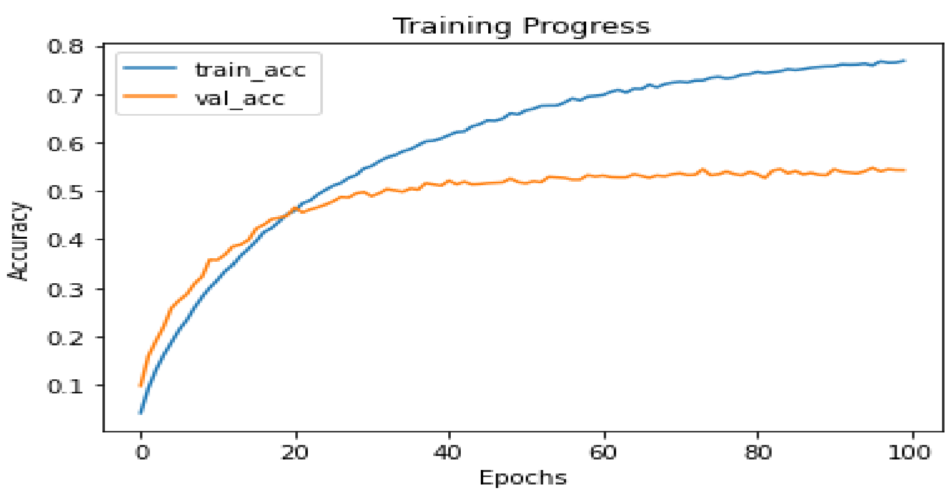


Fig 1 : Training and Validation Accuracy Vs Epoch

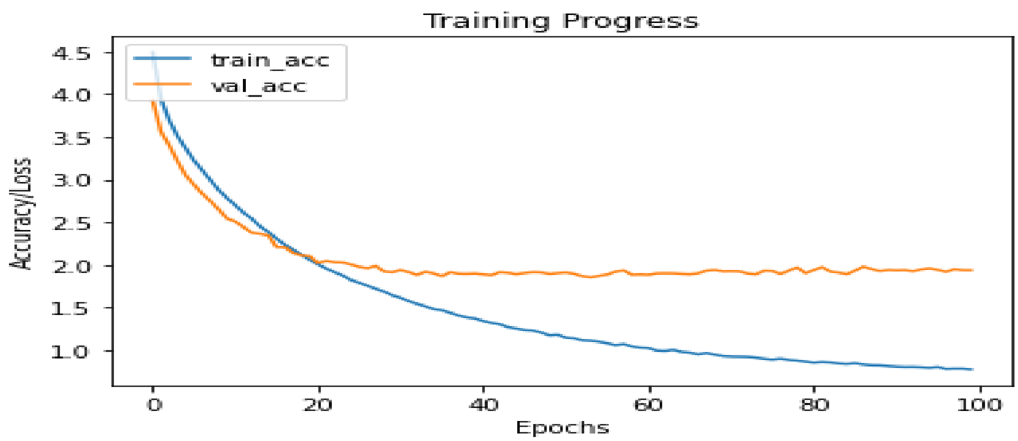


Fig 1.1 : Training and Validation LossVs Epoch

Code Implementation : Reproduce results on Dataset NOT Used in the Paper

Datasets and Hyper-Parameter Details

For this part I have used the **Fashion-MNIST** dataset provided by torchvision dataset for fine tuning with pretrained Visual Transformer on ImageNet dataset. Table 2.1 gives the dataset details.

Dataset Name	*Fashion-MNIST*
Number of Instances	*50000 Training, 10000 Test*
Associated Tasks:	*Classification*
Features	*Image size :* $1 \times 28 \times 28$
Missing Value	*NONE*
Total Number of Classes	*10*

Table 2.1 Dataset Details

Results WIthout Fine Tuning

For this part, apart from fine-tuning my model on Visual Transformer, I have trained a traditional CNN model with 6 layers on the Fashion-MNIST dataset to draw comparison.
Here is the Hyper-Parameter details for simple convolution neural network

Hyperparameters	*Values*

Number of Epoch	50
Batch Size	128
Initial Learning Rate	0.001
Optimizer	Adam
Loss	Cross Entropy

Table 2.2 Dataset Details

Table below shows the accuracy report of CNN model without fine tuning

	Accuracy	Loss
Training	93.2%	0.056
Test	62.23	0.64

Table 2.2 Accuracy Details Without fine tuning

Results With Fine Tuning

Below table provides the Hyper-parameter details used to fine-tune the model with Fashion-MNIST dataset

Hyperparameters	*Values*
Number of Epoch	15
Batch Size	128
Initial Learning Rate	0.001
Optimizer	Adam
Weight Decay	0.0001
Patch Size	7×7
Number of head	4
MLP Head Units (Size of Dense Layer)	[2048 , 1024]

Table 2.3 Hyper-Parameter Details

Following table gives the performance of the fine-tuned model

	Accuracy
Training	88.56%
Test	58.2%

Table 2.4 Model Performance

Discussion : Here simple CNN on data Fashion-MNIST is working comparatively better than fine-tuned models which is not the general case. This is mainly due to the less number of epochs in the fine-tuning part. Increasing the number on epoch would increase the performance of the model.

DUE to GPU limitation on Google Colab, I couldn't run the code for more than 15 epochs.