Assignment 3 - ELL793

Shauryasikt Jena, 2018EE10500
Amokh Varma, 2018MT60527

Indian Institute of Technology, Delhi

Abstract

A Vision Transformer (ViT) is a transformer that is targeted at vision processing tasks such as image recognition. Transformers found their initial applications in natural language processing (NLP) tasks, whereas the typical image processing system uses a convolutional neural network (CNN). Here, we developed a vision transformer for classification tasks considering images as sequence of patches. Object detection is a computer technology related to computer vision and image processing that deals with detecting instances of semantic objects of a certain class (such as humans, buildings, or cars) in digital images and videos. We implement this by studying the PASCAL VOC 2007 dataset via Faster RCNN (Region-based CNN). The methods discussed in this work are — "Convolutional Neural Networks", "Transformers", "Region-based Convolutional Neural Networks".

Introduction

While the Transformer architecture has become the de-facto standard for natural language processing tasks, its applications to computer vision remain limited. In vision, attention is either applied in conjunction with convolutional neural networks, or used to replace certain components of convolutional neural networks while keeping their overall structure in place. We show that this reliance on CNN's is not necessary and a pure transformer applied directly to sequences of image patches can perform very well on image classification tasks. When pre-trained on large amounts of data and transferred to multiple mid-sized or small image recognition benchmarks (ImageNet, CIFAR-100, VTAB, etc.), Vision Transformer (ViT) attains excellent results compared to state-of-the-art convolutional networks while requiring substantially fewer computational resources to train.

Deep convolutional neural network models may take days or even weeks to train on very large datasets. A way to short-cut this process is to re-use the model weights from pre-trained models that were developed for standard computer vision benchmark datasets, such as the Imagenet image recognition tasks.

1. Transfer learning involves using models trained on one problem as a starting point on a related problem.
2. Transfer learning is flexible, allowing the use of pre-trained models directly, as feature extraction preprocessing, and integrated into entirely new models.

RESNET-50: ResNet-50 is a convolutional neural network that is 50 layers deep. We can load a pre-trained version of the network trained on more than a million images from the ImageNet database. The pre-trained network can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals. As a result, the network has learned rich feature representations for a wide range of images. It is used by us as a feature extractor in our object detection pipeline.

Faster RCNN: This is a detection model which was introduced as an improvement over the tradition RCNNs with the Selective Search based region proposal. In this, the RPN is replaced by a deep model and the input to it is the feature extracted using a pretrained CNN backbone, like Resnet-50. We use the pytorch impelementation of resnet50 faster-rcnn for this project [1]

Experimentation

Part 1

First we use CIFAR dataset and train a Vision Transformer on it. As a baseline, we train a Resnet18 model (from Expt2 [2]) to see the kind of improvement that we can get. Then we vary certain aspects of the transformer to try and ascertain some clarity about the importance of different intrinsic parameters of the Vision Transformer.

Part 2

In this, we train a Faster RCNN model on the Pascal VOC 2007 dataset and find the performance of the model as per the metric mAP at instances where IOU values are 0.5 and 0.9.

- **mAP:** Mean Average Precision
- **IOU:** Intersection Over Union, a method to measure the correctness of bounding boxes

[1] https://pytorch.org/vision/stable/_modules/torchvision/models/detection/faster_rcnn.html#fasterrcnn_resnet50_fpn

[2] https://github.com/amokhvarma/ELL793_Assignments

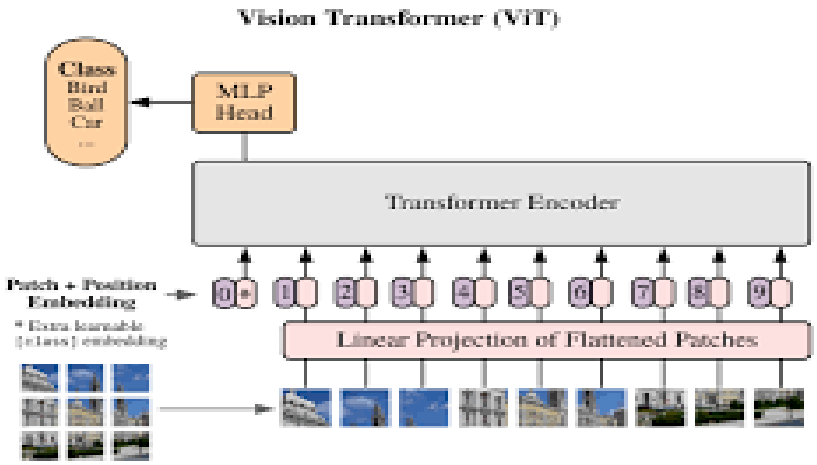

Figure 1: General Vision Transfer model

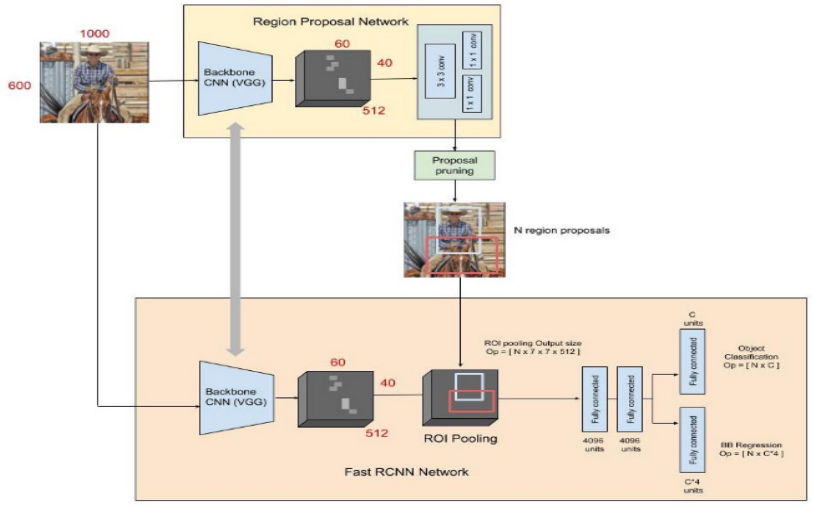

Figure 3: The RPN for region proposals and Fast R-CNN as a detector in the Faster R-CNN detection pipeline

Figure 2: Faster RCNN model

Dataset

CIFAR-10: The CIFAR-10 dataset consists of 60,000 32x32 colour images in 10 classes, with 6,000 images per class. There are 50,000 training images and 10,000 test images.

The dataset is divided into five training batches and one test batch, each with 10,000 images. The test batch contains exactly 1,000 randomly-selected images from each class. The training batches contain the remaining images in random order, but some training batches may contain more images from one class than another. Between them, the training batches contain exactly 5,000 images from each class.

Pascal VOC 2007: Pascal VOC is a collection of datasets for object detection. It has around 1,400 images with approximately 2,500 bounding boxes and labels. That is, each image can possibly have multiple objects in it. An example image can be seen in Figure 3

Figure 3: A sample image from Pascal VOC 2007

Results

We divide our results section into different subsections. Firstly, we talk about Problem 1, where we train a Vision Transformer for object detection and classification and compare its performance with that of a standard CNN (RESNET-50) for the same task. Afterwards, we discuss Problem 2, where we train Faster RCNN for object detection on the PASCAL VOC 2007 dataset. Then, we report the performance of Faster RCNN against IOU values of 0.5 and 0.9 for the bounding boxes. We used PyTorch for all purposes.

Problem 1: Vision Transformer

In this, we try two different experiments. Firstly, we will train a CNN network from scratch, with a RESNET backcbone. Then, we train using the imagenet pretrained models.

a) CNN from scratch: Here, we let the model learn from scratch and also learn the residual layers of the Resnets. The results can be seen in Table 1. We used a high value of batch size (128), as the images are quite small, so we need to train more data to get less noisy gradients. In this, we also do LR scheduling, where the learning rate is decreased by a factor of 10, whenever there is no improvement for 3 epochs. Further, we see over-fitting in this case. Due to this, we use dropout in the last layer to improve our results. The best result is shown after the full process.

S.No	Layers	lr	Train Accuracy	Test Accuracy
1	1	1.0	77.21	67.62
2	1	0.1	84.23	69.93
3	1	0.01	**91.35**	**79.21**
4	1	0.001	86.71	76.36
5	1	0.0001	86.53	77.23

Table 1: Performance on CIFAR10 Dataset using CNN with resnet built from scratch

b) Transfer Learning with ResNet-18: Note that, in the scratch case, we still initialise our model using the ImageNet parameters. We, however, noticed that we could not get good results with transfer learning. This is because, the transfer learning model was unable to capture the complexity of the problem. We tackled this problem by increasing the number of fully connected layers on the top of the Resnet backbone. The results for this can be seen in Table 2 .

S.No	Layers	activation	Train Accuracy	Test Accuracy
1	1	relu	52.21	47.13
2	1	sigmoid	50.22	46.17
3	2	relu	56.33	48.19
4	2	sigmoid	53.02	47.46
5	3	relu	61.04	53.13
6	3	sigmoid	58.77	50.62
7	4	relu	**61.21**	**55.93**
8	4	sigmoid	57.12	49.22
9	5	relu	60.12	51.14
10	5	sigmoid	58.53	49.23

Table 2: Performance on CIFAR10 Dataset with Tranfer Learning with layers after RESNET

c) Vision Transformer: Now we vary the different hyper-parameters of the Vision Transformer and see the classification performance. Note that, the train, validation and test sets are same across all settings. For this, we will be making small changes to each hyper-parameter after finding out the optimal hyper-parameters (stride 7, dim 128, patches 8) .

S.No	Patches	Emb Dimension	Depth	Test Acc.
Effect of Patches				
1	8	128	7	**81.13**
2	16	128	7	78.13
3	32	128	7	76.06
Effect of Emb. Dim				
4	8	64	7	69.11
5	8	256	7	73.32
Effect of Depth				
6	8	128	5	72.13
7	8	128	9	70.08

Table 3: Performance on CIFAR10 Dataset with different hyper-parameters of the vision transformer

We can notice from Table 3 that the ViT owes its performance to 3 parts :

- Number of Patches : This is the number of pieces that the initial image is broken down into, before being fed into the transformer. We can see that as we increase this value, we see a very small decline in performance. This is because, making too many pieces leads to loss of interdependent spatial features, which makes the classification weaker.
- Embedding Dimension : This is the dimension of the hidden representation created by the ViT. We can see that both higher and lower values are giving worse results. This is because very small feature space leads to underfitting and large feature space might lead to overfitting.
- Depth : This is the number of repeating encoder blocks that we use in the ViT. The concept behind this remains similar to the Embedding Dimension, in that, it is directly related to the complexity of the feature space. We need to ensure that the value isnt too low or high.

d) Vision Transformer with Image Augmentation: In this, we study the effect of different transformations on the performance of the transformer network. For this, we also show the training accuracy, because train set and test set are actually slightly different from each other, in terms of type of images. Vision Transformer doesnt seem to have particularly noteworthy test performance with data augmentation.

Sno	Augmentation	Train Accuracy	Test Accuracy
1	None	**83.12**	**81.13**
2	RH	82.34	78.41
3	RV	81.91	79.06
4	RR	81.18	79.11
5	RH+RV	83.15	79.71
6	RH+RV+RR	84.27	78.13
7	R	73.11	79.94

Table 4: Performance of ViT on CIFAR10 Dataset. RH, RV, RR , R stand for Random Horizontal Flip, Random Vertical Flip, Random Rotation and Random combination of above 3 respectively. We do this with the best model as reported above.

This is because many augmentation change the data in such a way that after breaking into even smaller parts, we are actually making the problem much more harder. This can be seen from Table 4

Problem 2: Faster RCNN

In this part, we attempt to train a Faster RCNN on Pascal VOC and observe its performance.

a) From Scratch: Firstly, we train the model from scratch and see the performance of Faster RCNN on Pascal VOC.

b) Pre-trained: Now, we repeat the previous experiment with model pre-trained on COCO dataset.

c) Bonus Experiment: As an extra baseline, we compare the traditional Faster RCNN with RetinaNet RCNN with Resnet-50 backbone.
The results are summarised in Table 5.

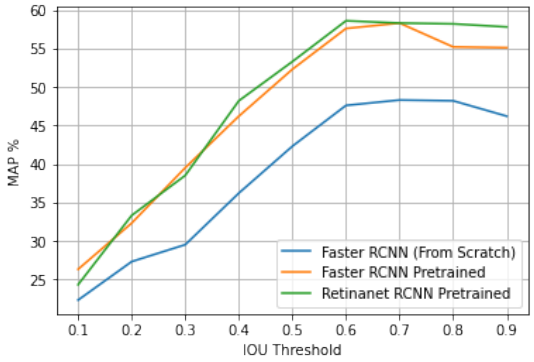

Figure 4: mAP scores for varying IOU over PASCAL VOC for given models

Model	IOU Threshold	Train mAP	Test mAP
From Scratch	0.5	44.1	42.5
From Scratch	0.9	48.7	46.2
Pre-trained	0.5	54.2	52.7
Pre-trained	0.9	57.3	55.2
Pre-trained RetinaNet	0.5	55.7	53.8
Pre-trained RetinaNet	0.9	60.5	57.9

Table 5: Performance on Pascal VOC 2007 with Faster RCNN RetinaNet RCNN

We can notice that clearly, RetinaNet has a slight edge over Faster RCNN. Further, we can also notice that the pre-trained models seem to be doing better than the models trained from scratch. This is because our dataset is not very large, compared to datasets like Imagenet. Further, we can see that as that IOU threshold increases, the mAP score also increase but after a certain value, it becomes slightly low, which is probably because a large number of the good boxes must be getting rejected.

d) Visualisation: First we look at effect of IOU threshold and plot it against the reported MAP values. Afterwards, we show a few examples of our Faster RCNN and its improvement in detection with training. Look at Figure 5

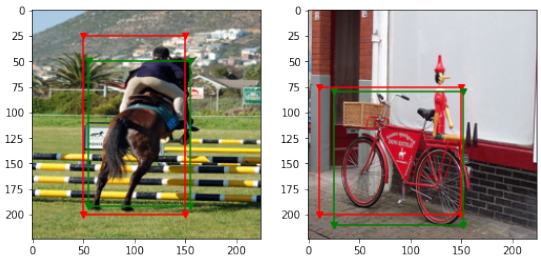

Figure 5: Object Detection on PASACL VOC
Red — After training 1 Epoch
Green — After training 10 epochs

Conclusion

In this assignment, we observed that the Vision Transformer is capable of performing at par with simple standard CNN architectures and even outperforming it for object detection and classification, when the concerned models are optimized for hyper-parameters and inputs for the best possible output. We also observed that embedding dimensions is the most crucial hyper-parameter for ViT as it affects the maximum change in performance over variation about the optimal value.

From performing object detection via Faster RCNN and RetinaNet, and tallying mAP scores over different IOUs, we observe that the best performance is obtained about IOU values in the range 0.7-0.8. Further, RetinaNet performs slightly better than the pre-trained Faster RCNN due to its architecture being more robust towards smaller datasets.