Assignment 3

Sagarika Sharma
2017CSM1012
Sujit Rai
2017CSM1006

Department of Computer Science & Engineering,
IIT Ropar

1 Introduction

The objective of this assignment was to attempt to optimize an existing state-of-art detector. The optimization can be in the terms of any of the following - the training model size, training time and/or test time.

For this assignment we used the basic YOLO object detector. The base line model that was used for optimization consists of 23 convolution layers each followed by batch normalization and Leaky relu with max pooling of stride 2 after few intervals. The model took input of dimensions (416x416x3) and produced output of dimension (13,13,5,25)

2 Optimization Techniques

2.1 Depthwise Separable Convolution

Traditional convolutional neural network performs function of learning the cross-channel correlations as well as the spatial correlations. However the hypothesis behind Xception net was that the spatial and cross channel correlations can be learned completely separately. The underlying advantage behind this approach is that the number of parameters required gets reduced to a large extent leading to the development of models such as Xception net and mobile Nets.

We also incorporated the idea of depthwise separable convolutions in this YOLO architecture. Each Convolution layers in the YOLO model where replaced by Separable 2D convolution and Pointwise convolution both Separable-Convolution and pointwise convolution was followed by Batch Normalization and Leaky Relu. We eliminated the Max Pooling layer and used strided Separable Convolution when required.

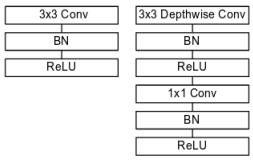

Figure 1: Depth-wise Separable Convolution.

The main purpose of the separable convolution is to learn the spatial correlations and that of the pointwise convolution is to learn the cross-channel correlations.

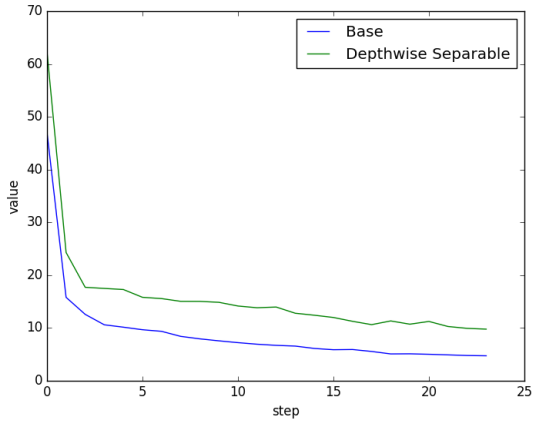

Figure 2: Loss comparison Base model and Base model implemented with Depthwise Separable convoltion. x-axis represents the number of epochs and y-axis represents the loss

Is was observed that the implementing the Depthwise Separable convolutions reduced the number of parameters 6 times and training time was reduced by almost 2 times. But the model was converging very slowly

2.2 Depth-wise Separable Convolution with Residual Connections

The idea behind residual connections was to reduce the complexity of deep models by learning the residuals also known as the approximation error between the layers. This idea was first implemented in Resnet and then was proved to be effective in building deeper more complex models.

We also incorporated the idea of residual connections in our model.

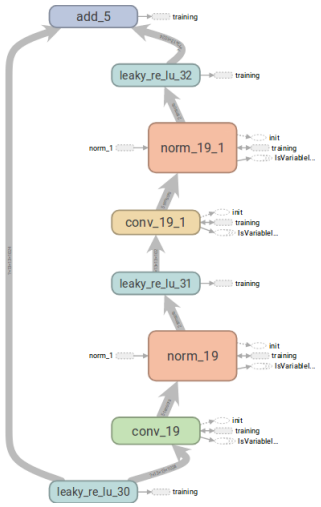

Figure 3: Residual Connection implemented using addition

The figure above was obtained by visualizing the graph of our model in the Tensorboard. As can be seen from the image we used the depth-wise separable convolutions for learning the residuals and then the residuals was added to the output of the previous layers output using the identity mapping.

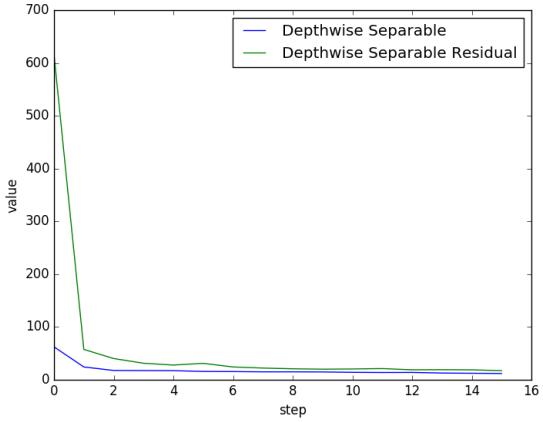

Figure 4: Loss Comparision of Depthwise Separable model with residual connections implemented using additions. x-axis represents the number of epochs and y-axis represents the loss

The number of parameters were same as that of the model with depth-wise separable convolution but the training time per epoch increased by

almost 0.2 times. This increase in training time can be because of the number of additions performed as an effect of residual connections. We observed that after a few epochs of the training, the total loss was decreasing very slowly which lead to early stopping of the training process. One hypothesis behind such behaviour can be that the gradients present in the initial layers are learning very slowly which can be because of the reduced gradients as an effect of the addition operation in the residual connections. In order to check this hypothesis we implemented residual connections using concatenation.

2.3 Depth-wise Separable Convolution with Residual Concatenation

We incorporated the idea of using concatenation instead of additions in the residual connections.

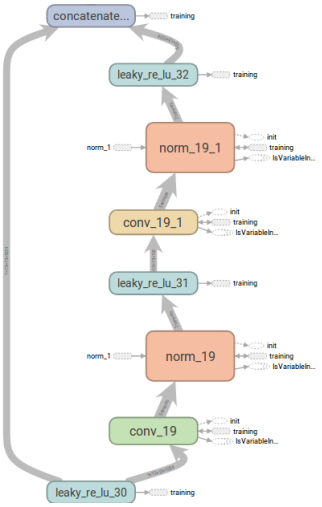

Figure 5: Residual Connection implemented using concatenation

The figure above was obtained by visualizing the graph of our model in Tensorboard. As can be seen from the image above, it is the same architecture. The only difference is that here we are concatenating the residual with the previous layers output.

After Implementing the model, we observed that the number of parameters was 3 times lesser than the base line but almost 2 times more than that of the model using depthwise separable convolutions. This can be because of the increase in the depth of the input channels because of the concatenation operation performed on the residuals.

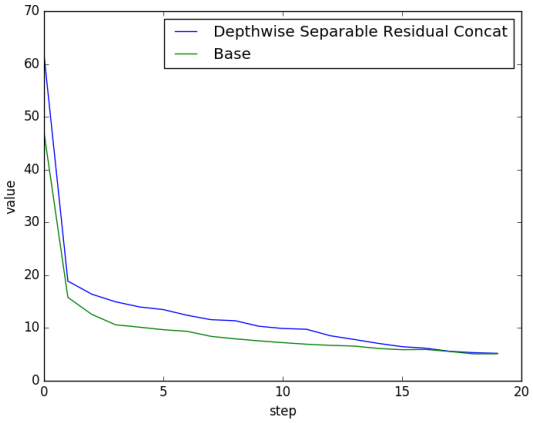

Figure 6: Loss Comparison of Base model and Depthwise separable convolution implemented using Residual Concatenation. x-axis represents the number of epochs and y-axis represents the loss

We observed that the Depth-wise separable convolution performed similar to the base model. The loss kept on decreasing gradually with time. and became almost equal to base model after 18 epochs.

2.4 Grouped Convolutions with Shuffling

Grouped convolutions was first introduced in alexnet architecture wherein the computation cost was divided among 2 processing units. Similar concept can be used for reducing the FLOPs in an deep learning model. The architecture introduced here is similar to the shufflenet architecture. Whereing the model is divided into 3 stages named stage-2, stage-3, stage-4 as shown in the figure below.

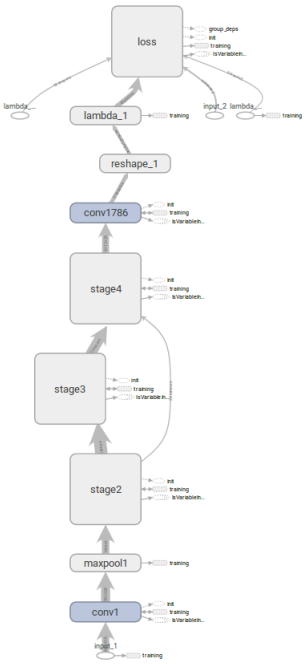

Figure 7: Grouped Convolutions in stages

Stage 2 and stage 4 consists of 4 shuffle-units named as blocks while stage 3 consists of 7 shuffle-units.

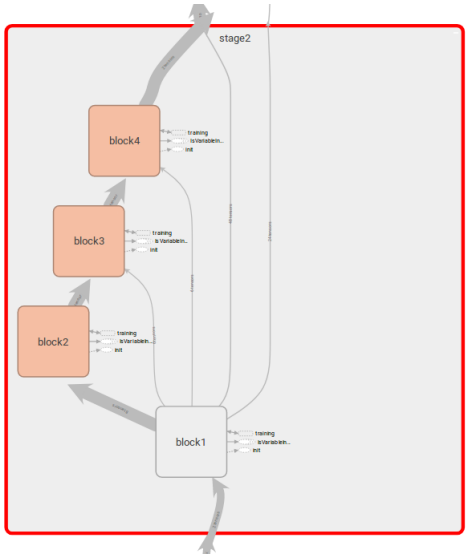

Figure 8: 4 shuffleunits in the stage 2 of our architecture

Each shuffleunit consists of 1x1 grouped convolution which is then shuffled, The output then applied with 1x1 depth wise separable convolution which is then followed by 1x1 grouped convolution in order to reshape the dimensions. Batch normalization was applied after each convolution in order to reduce the inter-class co-variance between the layers.

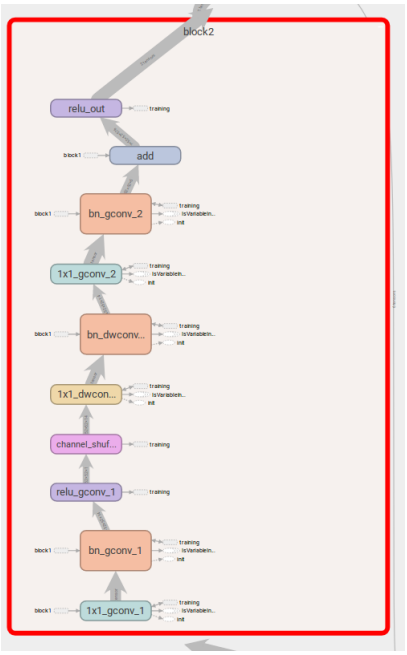

Figure 9: Shuffleunit

The number of parameters of shufflenet were 4 times less than that of the base model.

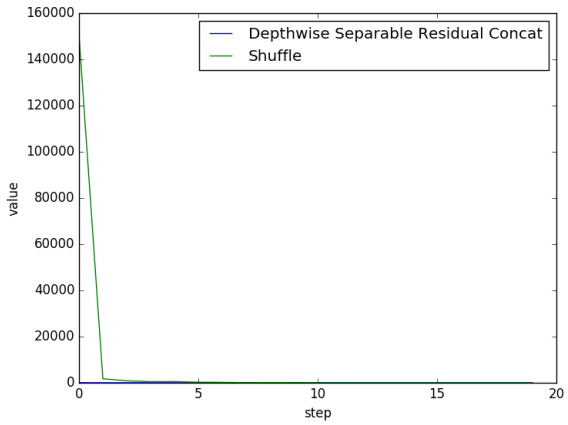

Figure 10: Loss Comparison of Depthwise Separable convolution using concatenating Residuals and Shuffle net architecture

We observed that it took shufflenet around 9 hours to reach a loss value of 5, while it took only 2 hours to reach the same value for the model with concatenating residuals.

3 Comparison

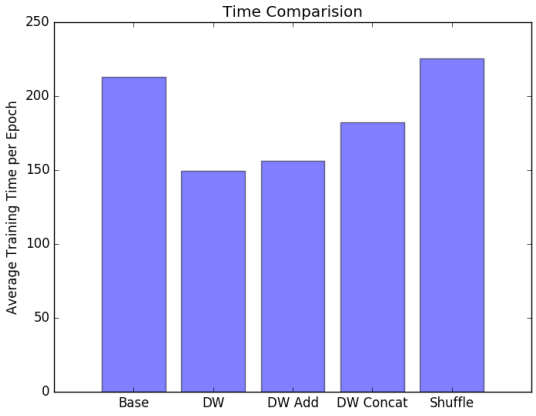

Figure 11: Average Training Time per Epoch for different techniques as compared to Base model

As can be seen from the table above the training of the models with depth-wise separable convolutions is lower than that of the base model, the residual connections with additions increases the time whereas the residual connections with concatenation increases the time even further. Surprisingly shufflenet here has the maximum training time, One hypothesis behind this might be due to the shuffling operation which would lead to transfer of data from GPU to CPU and vice-versa more frequenlty.

Model	Total Params	Trainable Params
Base	50,676,061	50,655,389
DepthWise Separable	11,851,181	11,818,407
Residual Addition	11,851,181	11,818,407
Residual Concatenation	22,299,949	22,260,263
Grouped Convolution with shuffle	15,299,688	15,240,984

Table 1: Table representing number of parameters obtained by each layer

Depthwise separable convolution with concatenating residual connections seems to perform better in terms of training time, parameters and well as accuracy, therefore we decided to use this technique for our final test of accuracy.

Model	Loss Value	Time
Base	5.474	2h 4m
Residual Concatenation	5.202	1h 55m
Grouped Convolution with shuffle	5.8	8h 45m

Table 2: Table representing the time taken to obtain loss value of around 5

4 Results

Parameters	Value
Loss XY	0.0648
Loss WH	0.1997
Loss Conf	0.0026
Loss Class	0.2040
Total Loss	0.4713
Average Recall	0.1093

Table 3: Table representing parameters obtained after training for 200 epochs

Figure 13: Detected bike as well as person

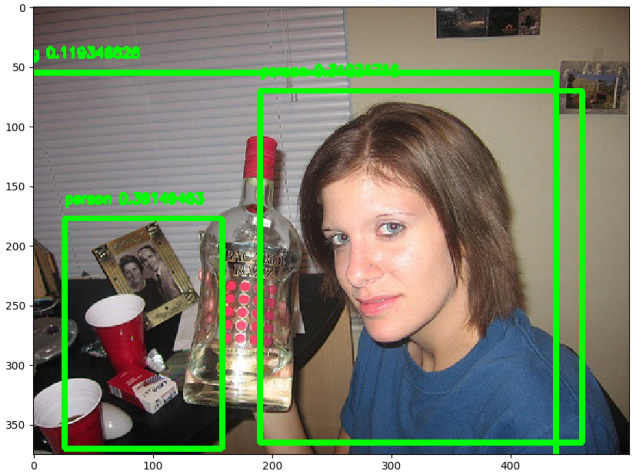

Figure 14: Detected person in the foreground as well as the person in the photo in background

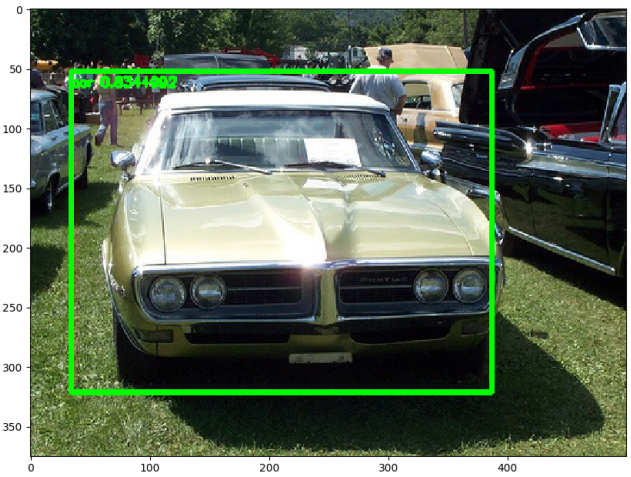

Figure 12: Detected car

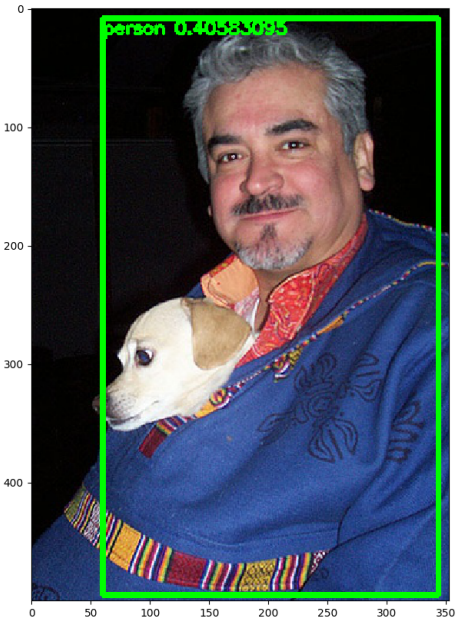

Figure 15: Detected single person