Object Detection in an Urban Environment

Setup of Code

The URL of all the code and associated document (Including this one) is:

https://github.com/HarisAshraf/Project_CV1

The repository includes the following Files:

Exploratory_Data_Analysys.ipynb

Explore augmentation.ipynb

create_splits.py

README.pdf (This file)

requirement.txt (not needed if using classroom workspace)

The analysis will follow the following steps (It is assumed that classroom workspace is provided)

Exploratory data Analysis

The Exploratory data analysis is done by using the Juypter workbook `Exploratory Data Analysis.ipynb`

Creating the splits

Use the python script create_splits.py to split the data into three directories. These are `training, validation and testing.`

The actual command is:

```
python create_splits.py --data_dir /home/workspace/data/
```

edit the config file:

```
python edit_config.py --train_dir /home/workspace/data/train/ --eval_dir /home/workspace/data/val/ --batch_size 4 --checkpoint ./training/pretrained-models/ssd_resnet50_v1_fpn_640x640_coco17_tpu-8/checkpoint/ckpt-0 --label_map label_map.pbtxt
```

Training and Evaluation

Use the command fro training

```
python model_main_tf2.py --model_dir=training/reference/ --pipeline_config_path=training/reference/pipeline_new.config
```

Then evaluation

```
python model_main_tf2.py --model_dir=training/reference/ --pipeline_config_path=training/reference/pipeline_new.config --checkpoint_dir=training/reference/
```

Improving the performance

Edit the `pileline.config` file to change the parameters as desired and then execute the above commands. Make sure that the directory name reference is changed to `ExperimenrtNN`, where NN is the experiment number.

Download and Process data

This step has already been done and the data is in the folder `/home/workspace/data`

Creating The animation (Copied from README.md)

Modify the arguments of the following function to adjust it to your models:

```
python exporter_main_v2.py --input_type image_tensor --pipeline_config_path training/experiment0/pipeline.config --trained_checkpoint_dir training/experiment0 --output_directory training/experiment0/exported_model/
```

Finally, you can create a video of your model's inferences for any tf record file. To do so, run the following command (modify it to your files):

```
python inference_video.py -labelmap_path label_map.pbtxt --model_path training/experiment0/exported_model/saved_model --tf_record_path /home/workspace/data/test/tf.record --config_path training/experiment0/pipeline_new.config --output_path animation.mp4
```

NEXT

THE DOCUMENT NOW FOLLOWS THE SUBMISSION TEMPLATE

Project Overview

In this project, we use Neural Networks to classify objects (Cars, Motorcycles and Pedestrians) in a video stream taken by a camera with a global shutter. The global shutter ensures that each frame is a complete independent image and be treated as such.

The learning dataset is a collection of color images taken at one second apart (decimated 10Hz data). The test images are taken at 10Hx.

The purpose of the project is to divide the data into training and validation dataset, train the neural network and that run the test data through it. Validation data will be used to fine tune the network and the test data will be run after training is complete.

Some of the changes made to improve the performance is trying various Augmentations, change annealing rate and increase number of steps.

Tensor board was used to visualize the data an perform analysis.

Dataset

Dataset Analysis

One shot is in beginning of the record files, other is middle and the last one is the end. 1st can be seen that the driving situation are different in all situations. There are some night images also.

There are quite a few cars but few pedestrians. So, the NN may not be able to learn to classify them with high precision.

Some images are taken in bad weather so the NN should be able to be independent of change in weather. However, this may make the learning more difficult also.

The images are taken within city and also city highways.

Figure 1: Initial Set of Images

Figure 2: Images from middle of teh set

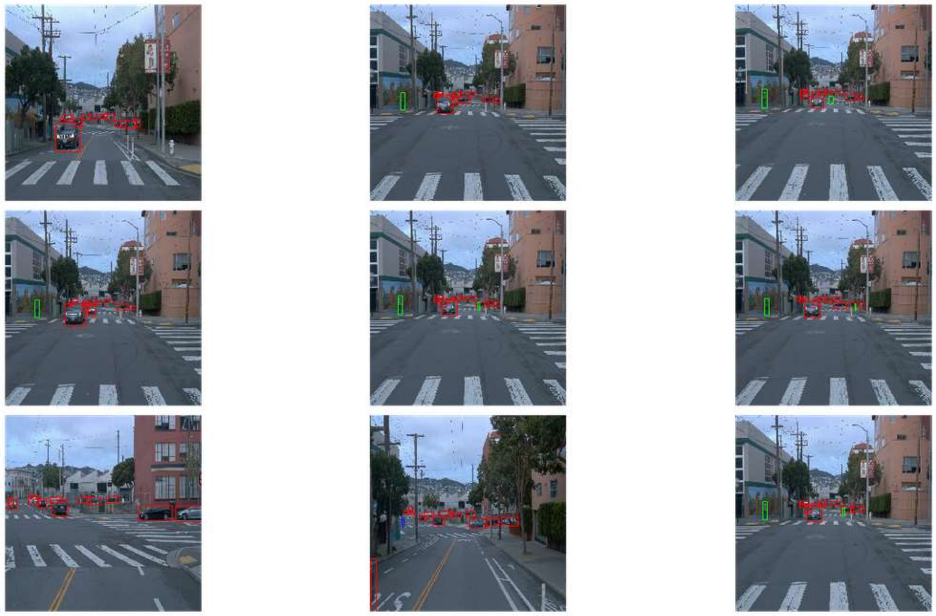

Figure 3: Images form the end of the set

Cross Validation

The technique that was used is called “Hold out Cross Validation”

The process of this king of validation is as follows: Split the dataset into two parts: the training set and the test set. Usually, 80% of the dataset goes to the training set, 10% to the test and 10% validation set.

The model is then trained on the training set, then it is validated. This process is continued until at Neural Network is trained to an acceptable level.

Finally, the neural network is tested with the test set and results plotted.

Augmentations

First augmentation that was tried was “Monochrome”

Second Was “Random Distort Color”

Third was “Random Black Patches”

Figure 4: Monochrome and Random Distort Color Augmentations Examples

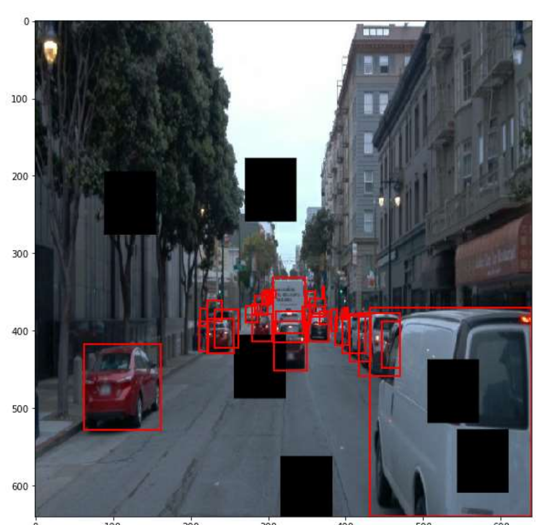

Figure 5: Random Black Patches

Splitting the data

A python program was written to split the data into train, validate and test directories. The test split is not really needed by this project, but the instructions asked for it.

The list of file was shuffled and an 80:10:10 split was performed by moving the files.

The section detailing the training and eval dataset is presented here.

Training

Reference Experiment

The data splitting script saves 80% of the files into a directory called **training**. Then the provided training application was executed. The config file was already configured for 2500 steps and that setting was used.

The reference model did not fare too well. In the video it was able ot occasinaly classify a vehicle. No vehicles were classified at night.

```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.018
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.045
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.011
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.005
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.052
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.055
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.008
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.030
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.065
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.031
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.148
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.168
```

<Loss Data not shown>

It can be seen that the recall and precision numbers are rather small. No vehicles were classified in the night images. During the day the performance was a little better but most of the time no vehicle were classified. There was success in few images.

TensorBoard output is shown in Fig 6.

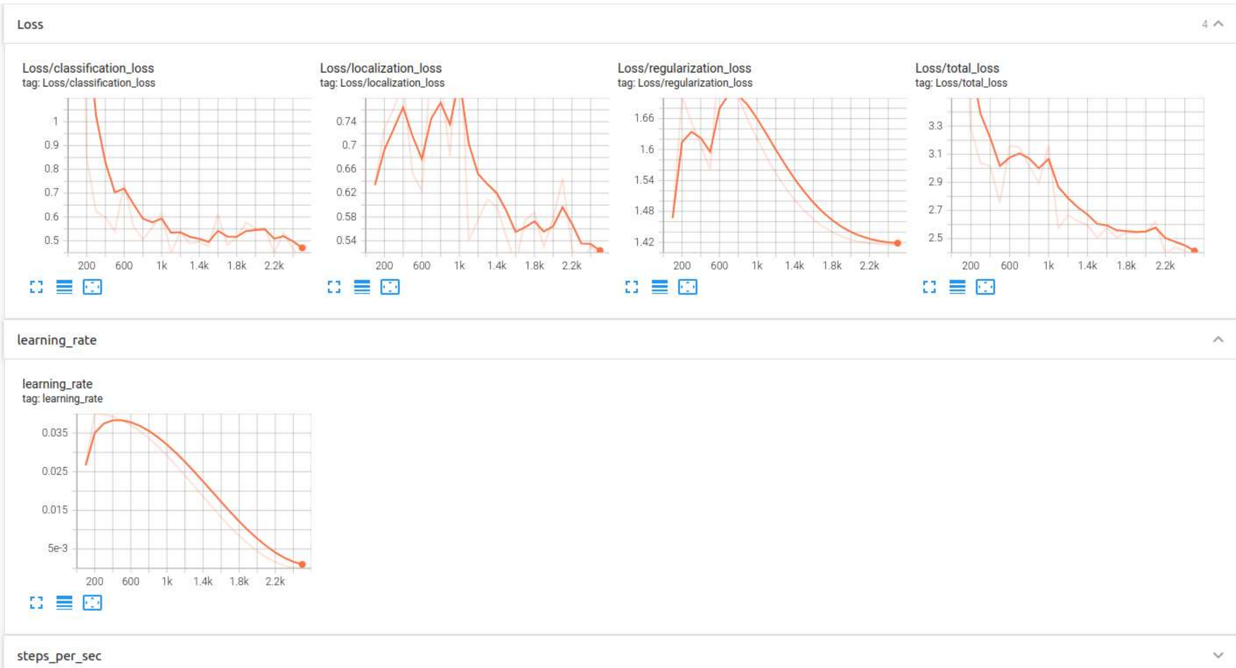

Figure 6:Tensor Board output for reference model

Training and Validation Loss Comparison

Table 1: Loss Comparison Reference Model

	Training	Validation
Classification Loss	0.45	0.58
Localization Loss	0.52	0.63
Regularization Loss	1.42	0.99
Total Loss	2.40	2.19

Improve on the reference

Following experiments were performed to improve the performance:

Experiment 0: 12500 Steps

First experiment was performed by changing the step to 12,500. Addition of extra steps made the loss curves really go down and the classification of vehicles was very accurate both during the daytime and nighttime test cases.

```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.162
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.314
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.145
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.063
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.408
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.641
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.048
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.175
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.250
Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.143
Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.561
Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.721
```

<Loss Data not shown>

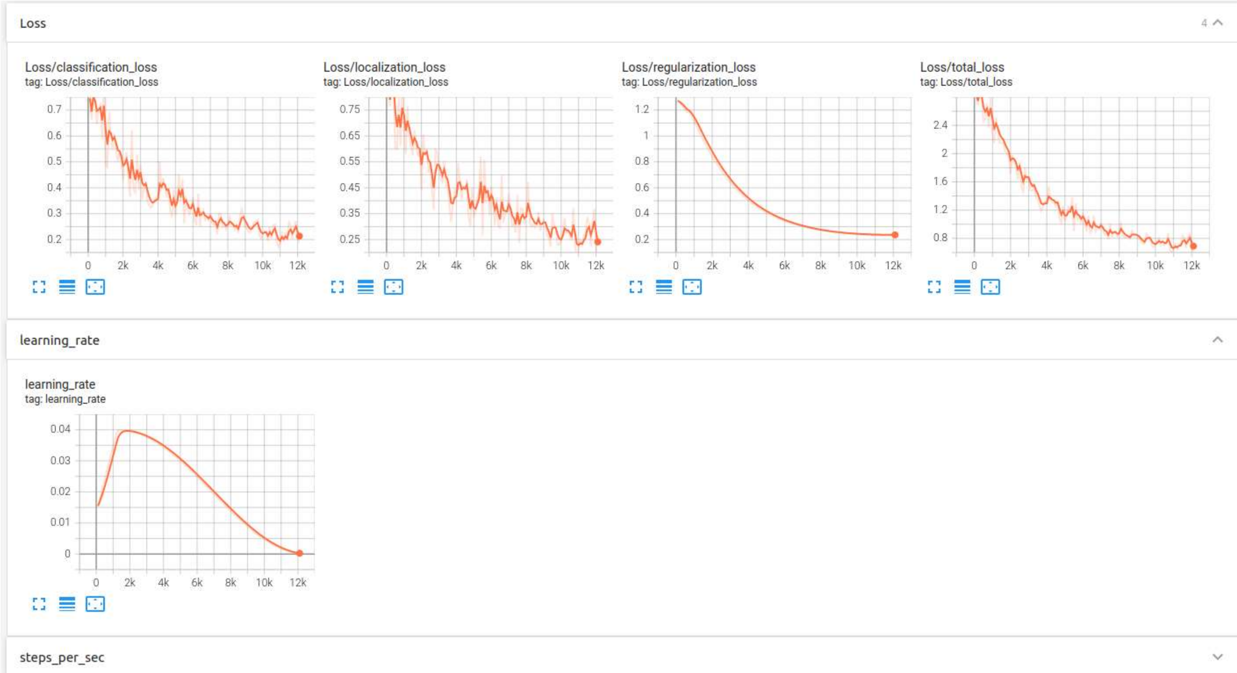

Figure 7: Tensor Board output of Experiment 0

Training and Validation Loss Comparison

Table 2: Loss Comparison Reference Model

	Training	Validation
Classification Loss	0.20	0.26
Localization Loss	0.25	0.28
Regularization Loss	0.25	0.24
Total Loss	0.75	0.77

Screenshot of validation output is shown in Figure 9.

Figure 8: Experiment 1 Validation Output

Screen shot of Test videos with this augmentation are shown below.

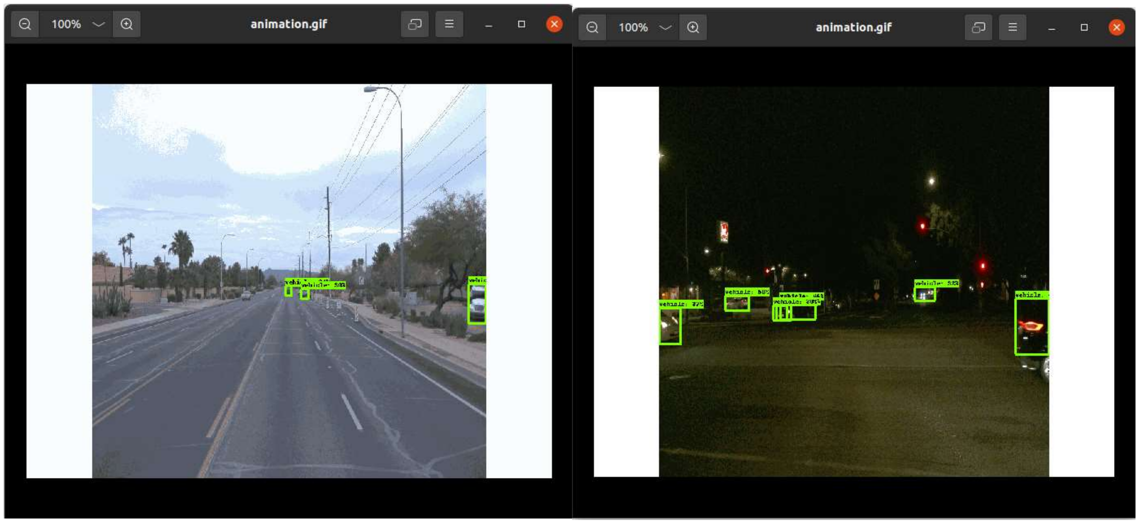

Figure 9: Experiment0, video screenshots

Experiment 1: Applying Random Color Distort Augmentation

Next experiment was performed adding the Augmentation: Random Distort. The no of steps was reduced to 2500 again.

Not much improvement was realized over the reference model..

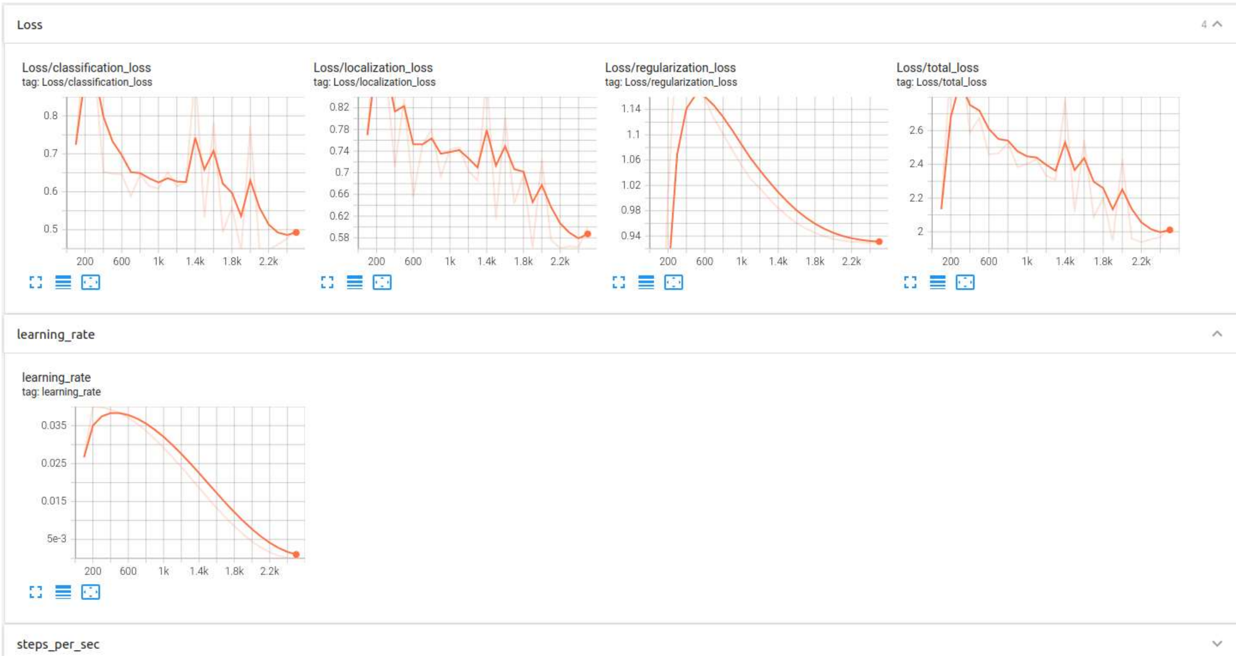

Figure 10:Tensor Board output of Experiment 1

Training and Validation Loss Comparison

Table 3: Loss Comparison Reference Model

	Training	**Validation**
Classification Loss	0.50	0.61
Localization Loss	0.60	0.61
Regularization Loss	0.94	0.93
Total Loss	2.00	2.15

Experiment 2: Double learning rate.

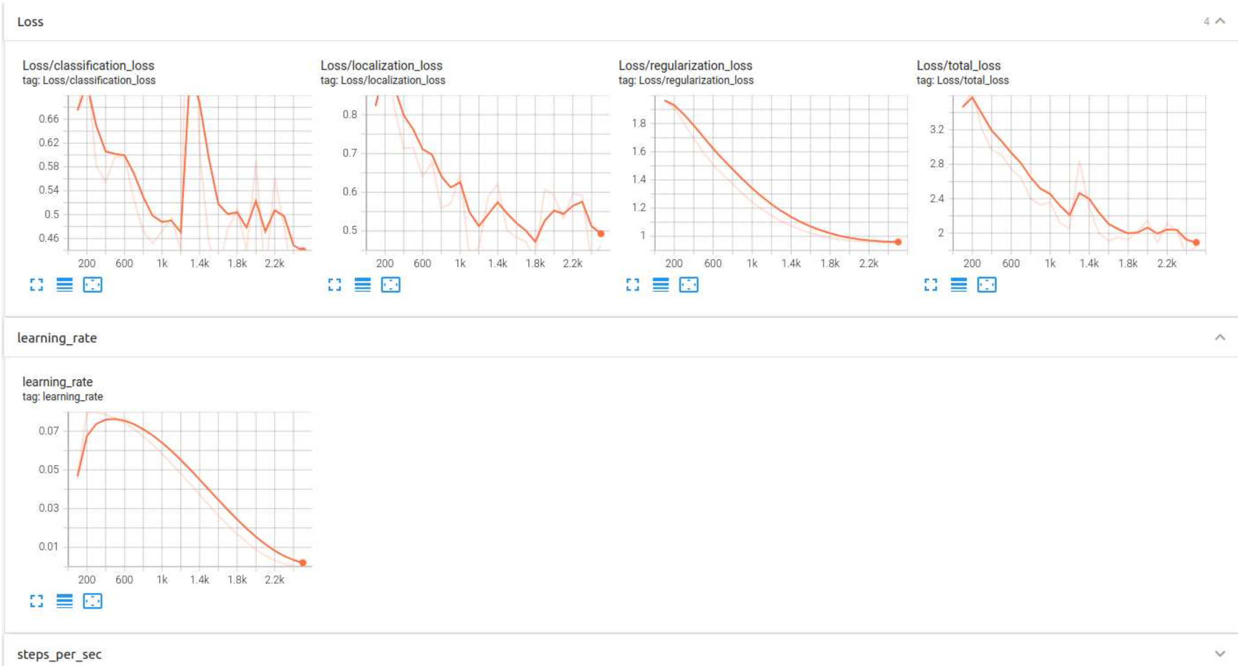

Figure 11:Tensor Board Output of Experiment 2

Training and Validation Loss Comparison

Table 4: Loss Comparison Reference Model

	Training	Validation
Classification Loss	0.45	0.60
Localization Loss	0.50	0.58
Regularization Loss	0.95	0.96
Total Loss	1.80	2.14

Conclusion

In this project we used the Waymo open data set to train and then validate and test a Neural Network. The goal was to train the NN by using images of the front camera. In the training images, Cars, Motorcycles and Pedestrians were classified.

A reference model was given and then to improve performance certain number of experiments were performed.

Of all the effort made, it was found that increasing the number of steps was the most efficient way to increase performance. Other methods did improve performance, but only by a little bit.