AI2100 PROJECT PRESENTATION

May 06, 2022

Topic

Object Detection using Transfer Learning Approach

Group Members

- Kanumuri Bheemeswara Vijay Varma
- Sachin Karumanchi
- Vallepu Manikanta
- Abhiroop Chintalapudi
- Abin S

Abstract

- we aim to build a Deep Convolutional Neural Network for object detection through Transfer Learning.
- Pre-trained networks are taken which are already trained for Object Detection.These pre-trained networks are used to reduce the computational cost and also decrease the training time to train the model.
- The main advantage of using Transfer Learning technique is to cut down the number of training instances.
- As large datasets may not always be available for training Neural Networks and with smaller data sets, using Transfer Learning gives more accurate results with improved efficiency while training new models.

Introduction

- **Deep Convolutional Neural Nets** have achieved significant success and had led a series of breakthrough for Image Recognition and Object Detection.
- The progress is attributed to new methodologies, algorithms, and network topologies, as well as better hardware, new models, and larger datasets.
- **Transfer Learning** refers to a process where a trained model on one problem is used on another problem which is similar (or) related to first problem.
- We examined the Transfer Learning approach with three top-performing models from keras Library.
- Top performing models in the Image Net challenge.
  1. VGG (2011, 12)
  2. Residual network (2015)
  3. GoogleLeNet (2014)

Motivation

- The main problem with **Convolutional Neural Nets** is that they require a large quantity of data to train, which many organizations lack.
- Only a few organizations have the resources to collect and preserve huge amounts of data.
- Using **Transfer Learning** approach, we can reduce the amount of data we are using to train our model and also reduce the computational expenses and the training time.

Preliminary Results

Dataset

- We collected the data from the Internet that is similar to the **COCO Dataset** but with a lesser number of classes and lesser number of training data.
- Compared VGG16, ResNet50, InceptionV3 Model Accuracies with custom data.

VGG16

- Imported the Model from tensorflow.keras.applications.vgg16.
- Saved the Model with the Output Layer.
- Predicted the accuracy of the Image "car.jpg".
- The Output turns out to be a Sports Car with accuracy of 85.15% .

Preliminary Results

Residual Network

- Imported the model from tensorflow.keras.applications.resnet50.
- Saved the Model with the Output Layer.
- Predicted the accuracy of the Image “car.jpg”.
- The Output turns out to be a Sports Car with accuracy of 85.95%.

Google LeNet

- Imported the model from tensorflow.keras.applications.inception_v3.
- Saved the Model with the Output Layer.
- Predicted the accuracy of the Image “car.jpg”.
- The Output turns out to be a Sports Car with accuracy of 81.90%..

Transfer Learning

- It is a process in which we use an already Pre-trained Model as the starting point for achieving similar task.
- We can get better performance, fast convergence, less computational power, and more initial accuracy by employing Transfer Learning.
- The Codes and Implementation can be found here.

Workflow

- Trained an Object Detection Model using Transfer Learning from a Pre Trained Model.
- **Collecting Data**
  1. Collected Data from the Internet and created our Data Set for training and testing the Models.
  2. Decided to identify objects of 10 classes which include Person, Dog, Cat, Chair, Table, Car, Bike, Water Bottle, Mobile, Laptop. We collected 15 Train Images, 3 Test Images for every class in the Data.

Workflow contd

- **Annotating**
  1. Started annotating the Images and drew Bounding Boxes for each object in the given Image of Training Data and Test Data.
  2. **labelimg** tool is used to open source tool for annotating the images. Through this tool we labelled the 150 Training Images and 30 Test Images manually in a Linux Machine.
- **Generating TF Records**
  1. Used Tensor Flow for Object Detection Model.
  2. Since Tensor Flow models train on Tensor Flow Records, we had generated TF Records of our Custom DataSet.
- **Pre-trained Model Info**
  1. Used Tensor Flow Object Detection API to train our Transfer Learning Model.
  2. Chose **SSD**(Single Shot Detector) ResNet50 640*640 model previously trained on COCO2017 dataset and download the respective Pipeline Configuration file.
  3. SSD is chosen over other models becauase SSD takes less training time than any other models.

Workflow contd

- **Setting up Colab Environment**
  1. Used **GPU** provided by Google Colab.
  2. Enabled GPU Runtime for Colab. Installed Tensor Flow in the Notebook. then cloned the tensorflow models Repository from GitHub into the Colab Contents.
  3. Downloaded the pretrained model(SSD ResNet50) configuration file into Colab Contents.
  4. Installed **Protobuf** (Protocol Buffers), for working with Configuration Files and Training Parameters as TensorFLow Object Detection API uses Protocol Buffers.
  5. Tensor Flow Object Detection API is not installed via TF installation. Installed it separately using some commands.
  6. After setting up our whole environment and tools, we started training our Object Detection Model using the training commands.

SSD ResNet50 Architecture

- Chose SSD ResNet50 Model for Object Detection.
- The architecture of the Model is as follows:
  - It reshapes any given Image into 640*640 pixels.
  - ResNet50 is a 50 Layer Deep Convolutional Neural Network.
  - The SSD Model uses ResNet50 for Feature Extraction with L2 Regularization(weight = 0.0004).
  - Weights are Initialized from a Gaussian Distribution of Mean 0.0 and Standard Deviation of 0.03.
  - The layers have a Rectified Linear Unit (ReLU) activation.
  - Also Batch Norm is applied to the Layers with a decay of 0.997 and epsilon = 0.001.
  - It uses Momentum Optimizer and Stochastic Gradient Descent with varying Learning Rate whose initial value is 0.04.
  - The Momentum Optimizer Value is 0.9.
  - The Training takes place for 8000 steps with a Batch Size 4.

Training Results

- Trained the SSD ResNet50 Model using our Train Data for 8000 steps with a batch size of 4. The training took around 1 hour 20 minutes.
- The Loss Values for every 100 steps is given by the Tensor Board Server. By initializing the Tensor Board server, The Plots for Classification Loss, Localization Loss, Regularization Loss, Total Loss for every 100 steps up to 8000 steps are generated.
- For each plot, X axis represents the Number of Steps and Y axis represents the Loss Value.

Training Results

Total Training Loss

- The Model almost converged for 8000 steps.

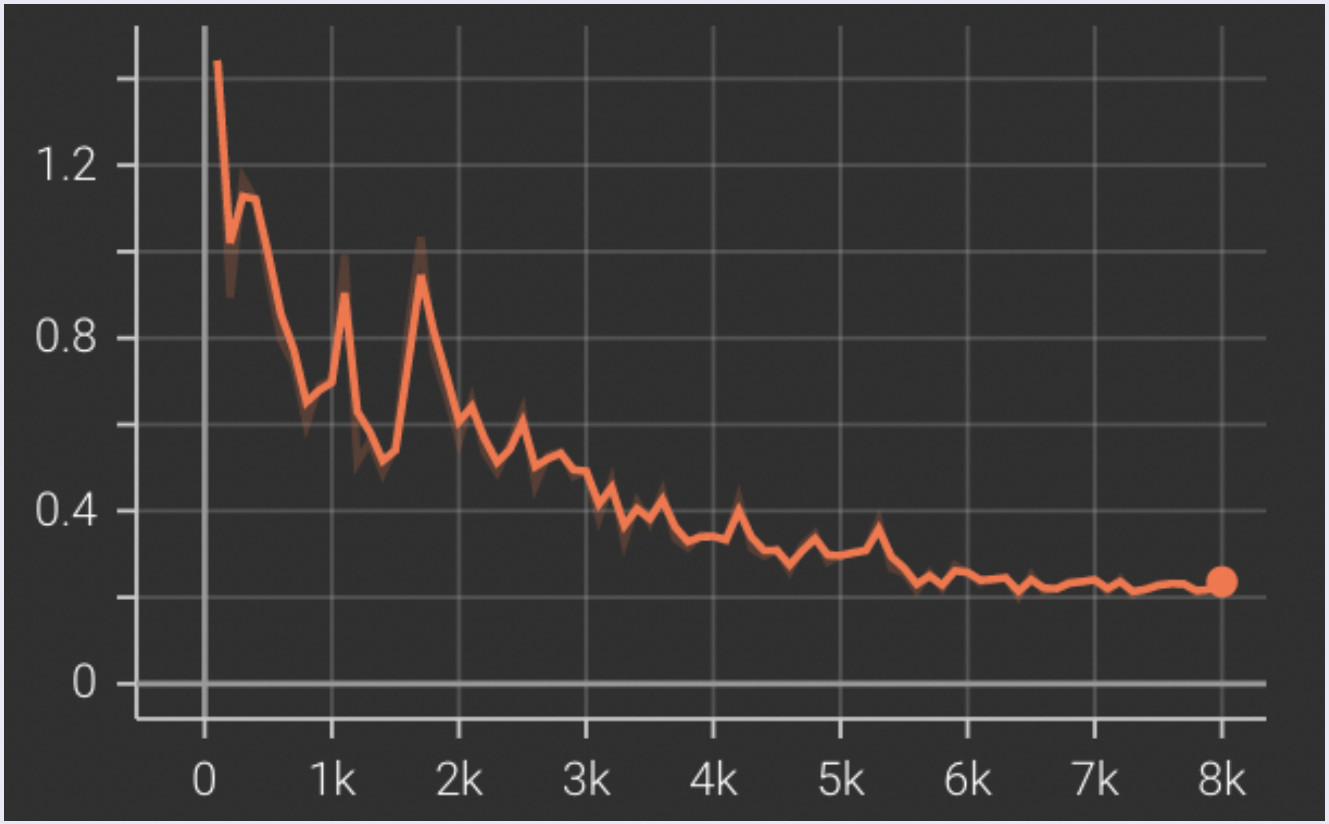

Figure: Total Training Loss

Training Results

Classification Loss

- Classification Loss is the loss of predicting different Class Label for a given Class label when Bounding Box is already predicted.

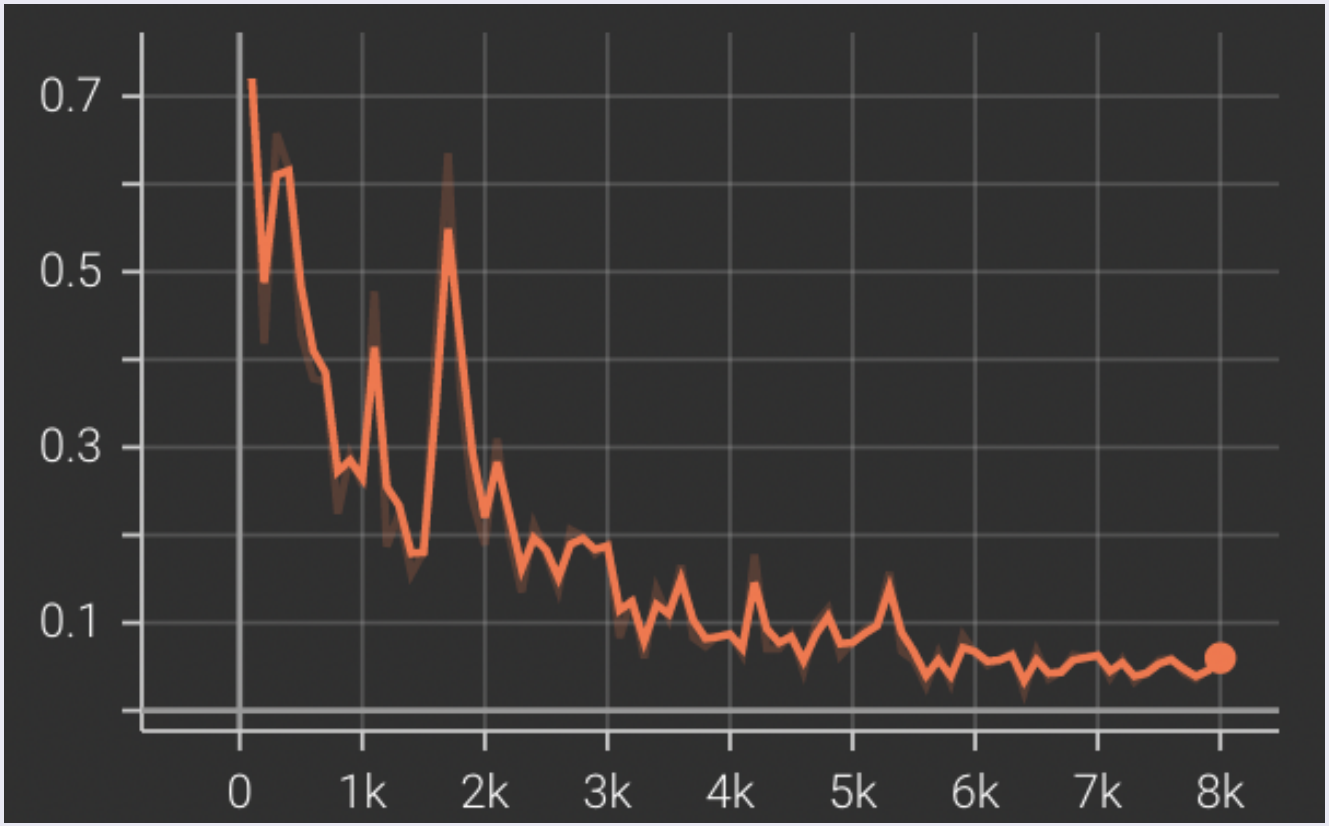

Figure: Classification Loss

Training Results

Localization Loss

- Localization Loss is the loss of improperly detecting the Boundaries of the Bounding Box for an Image.

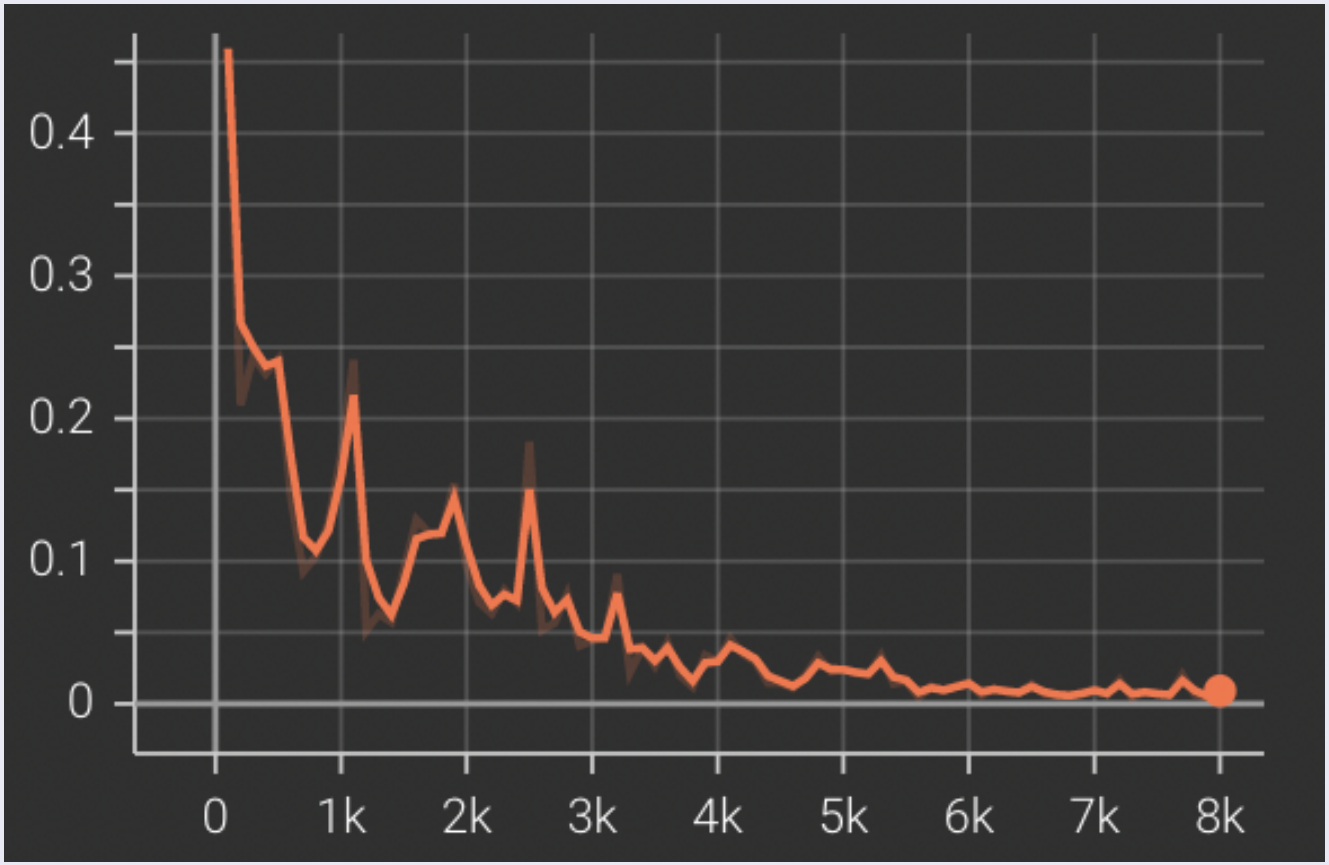

Figure: Localization Loss

Training Results

Regularization Loss

- Regularization Loss is the Loss produced by the Regularization Function which is used to optimize our model to generalize better.

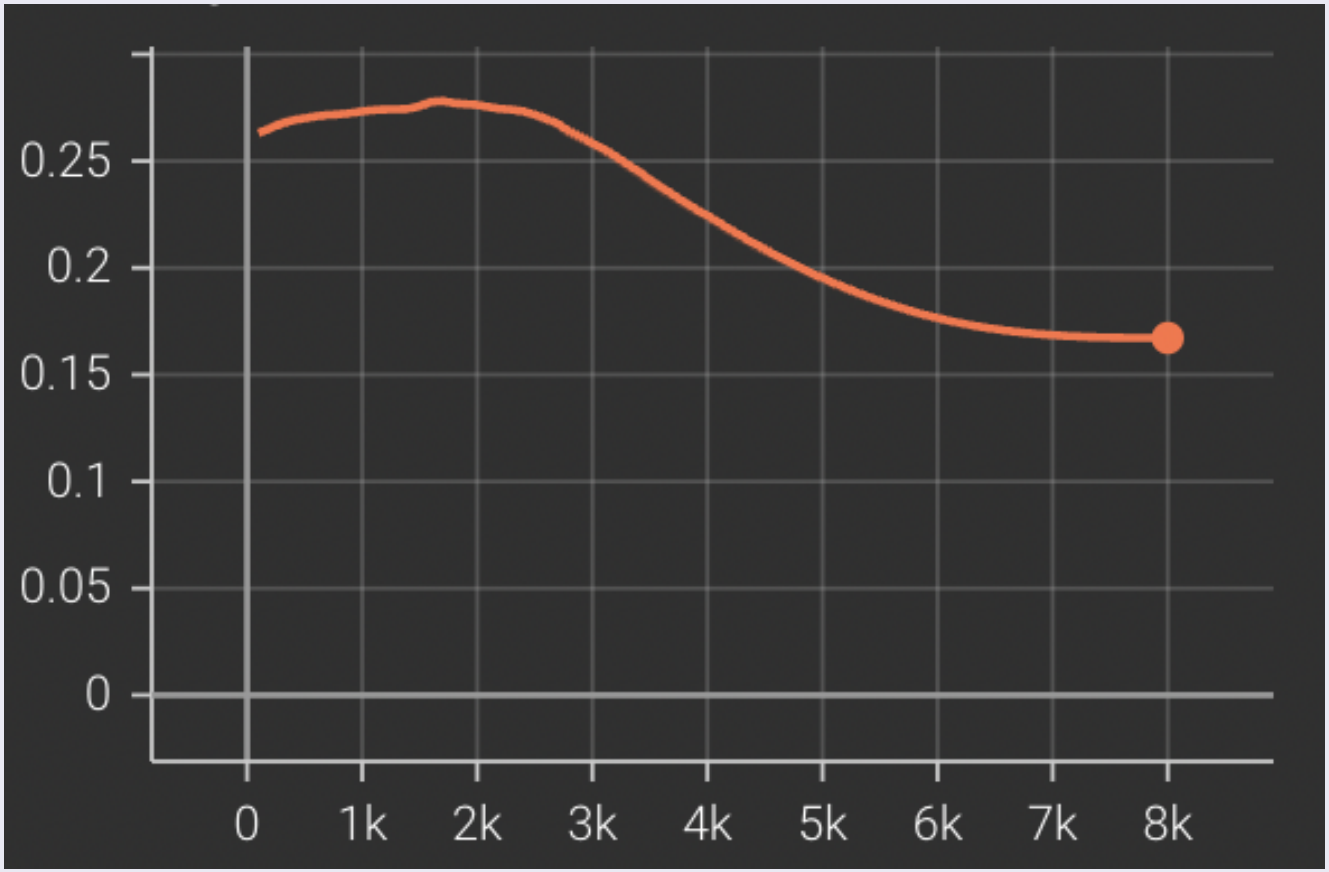

Figure: Regularization Loss

Training Results

Learning Rate

- Learning Rate increased in the Initial stages and then it steadily decreased and nearly converged at the end.

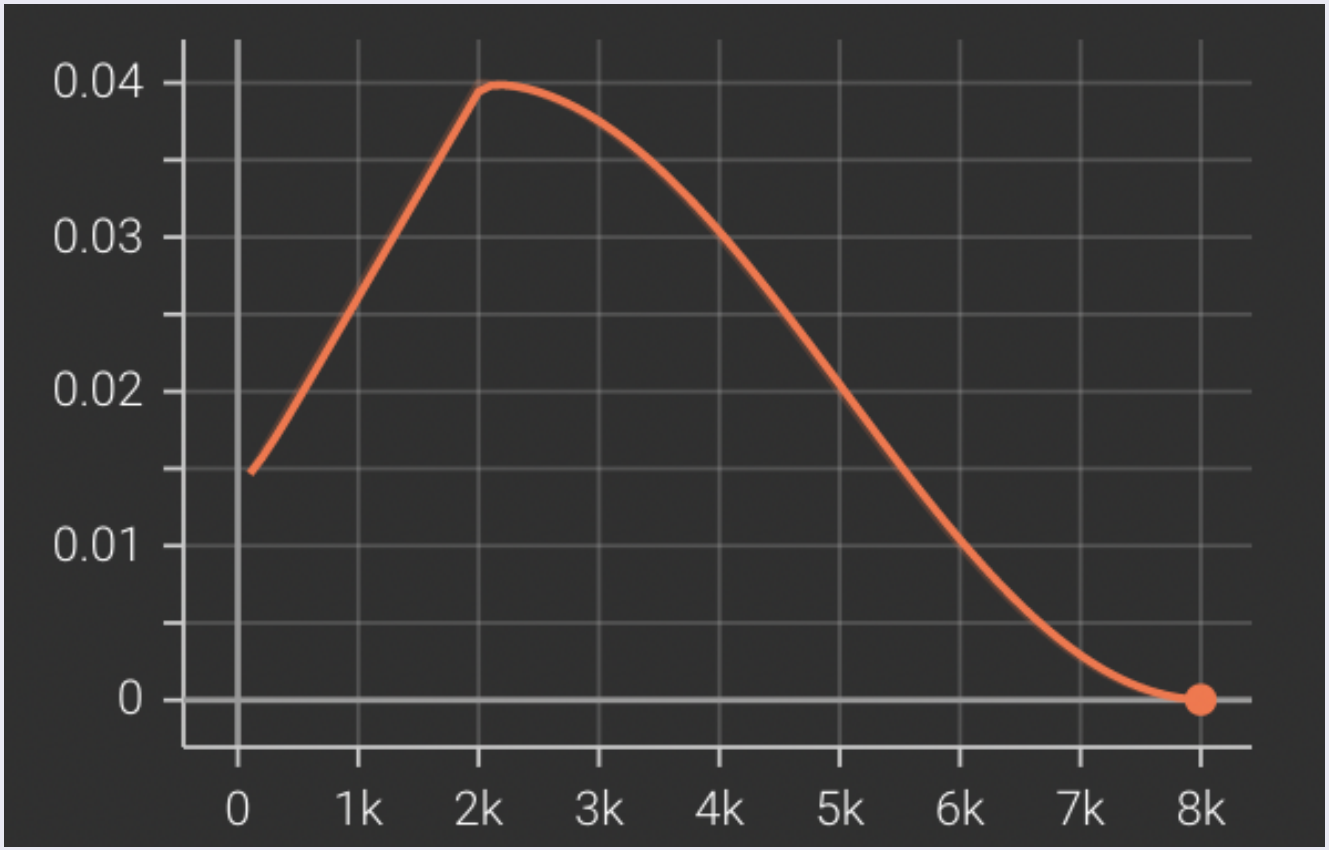

Figure: Learning Rate

Testing Results

- After the training we tested the Model from our collected Test Data.
- The Model predicted very well for classes with less variations in train data whereas for biased and many variations of train data, the Model did not accurately detect the other classes.
- The Model detected Car with 95% since cars do not have large variations.

Figure: Car

Testing Results

- The Model predicted the below Cat Image with 60% accuracy, as these type of cats are less in Training Data.

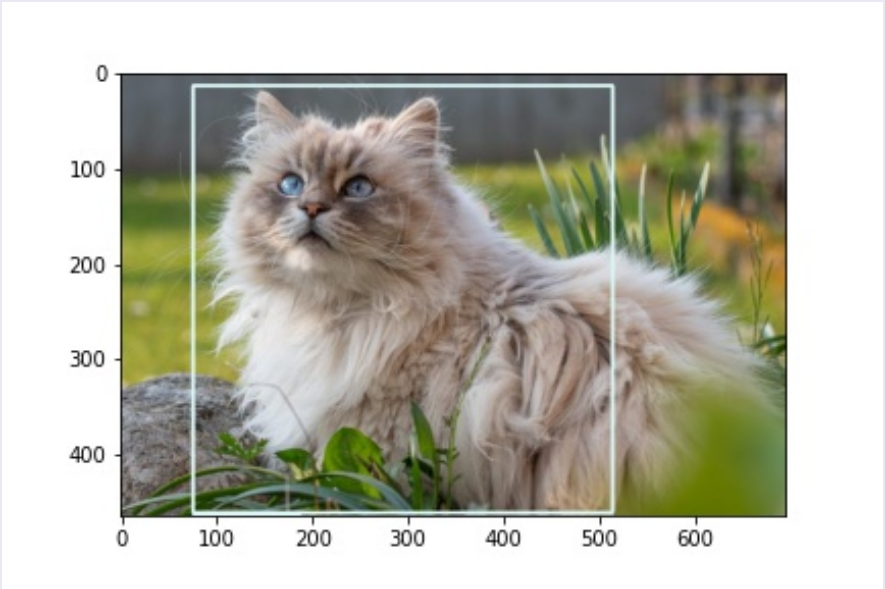

Figure: Cat

Testing Results

- Similar to Cars, Motor Bike does not have many variations, so it predicted the below Image as Motor Bike with 98% accuracy.

Figure: Motor Bike

Testing Results

- The Model predicted the below Dog Image with 53% accuracy. The Dog's training data contained many dog breeds. So, the Model could not predict Dog images with higher accuracy.

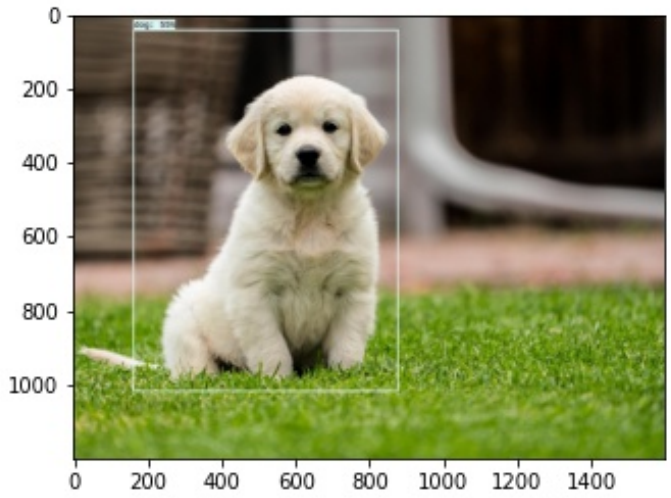

Figure: Dog

Conclusions

- The Training Time of Our Model is around 1hr 20 minutes which is quite efficient compared to a model which is trained from scratch for Object Detection.
- Our Model converged for nearly 8000 steps. It converged fast in comparison with other models.
- It uses less computational power for training and is also more efficient.
- From the above results, we can conclude that Transfer Learning produces much High Accuracy with less Training Data, less computational power, converges faster than a Model trained from scratch for Object Detection.

THANK YOU