Object Detection using Transfer Learning Approach

Kanumuri Bheemeswara Vijay Varma
BTech Artificial Intelligence 2020
AI20BTECH11012
IIT Hyderabad

Sachin Karumanchi
BTech Artificial Intelligence 2020
AI20BTECH11013
IIT Hyderabad

Vallepu Manikanta
BTech Artificial Intelligence 2020
AI20BTECH11014
IIT Hyderabad

Abhiroop Chintalapudi
BTech Artificial Intelligence 2020
AI20BTECH11005
IIT Hyderabad

Abin S
MTech Biomedical Engineering 2020
BM20MTECH11005
IIT Hyderabad

***Abstract*—In this work, we aim to build a Deep Convolutional Neural Network for object detection through Transfer Learning. We had taken some pre-trained networks which are already trained for Object Detection. The necessity of using these pre-trained networks is 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.**

I. Introduction

Deep Convolutional Neural Nets have achieved significant success and have led a series of breakthrough for Image Recognition and Object Detection. The advancements are not only due to powerful hardware, new models and larger datasets but its also due to new techniques and algorithms and network architectures. Here we talk about Transfer Learning approach. In general 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 examine the Transfer Learning approach with three top-performing models from keras Library.

(1) VGG (2011, 12)
(2) Residual network (2015)
(3) GoogleLeNet (2014)

These are the top performing models in the Image Net challenge.

Motivation:

The main problem with these Convolutional Neural Nets is that they require a huge amount of data to train where data is mostly unavailable to many organizations. Only a few organizations have resources to collect and preserve huge amounts of data. By 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.

II. Literature Review

The Reference Paper[1], the Comprehensive Survey on Transfer Learning describes the improvement of performance of Transfer Learning. It shows the advantages of Transfer Learning like the need for a lesser amount of training data, computational power. Different transfer learning approaches are described in the paper.

In Reference paper[2], Object Detection and Tracking for Community Surveillance using Transfer Learning is discussed about pedestrian object detection in real time using Transfer Learning. A Yolov3 pre-trained model is used. It shows how the computation and training is feasible with the Transfer Learning approach.

VGG:

This paper tells about the effect of the Convolutional Network Depth on it's accuracy on large datasets. Here these models use very small (3x3) Convolutional filters and by pushing the depth to 16-19 weight layers, the model shows a significant improvement on the Image Net Dataset.

GoogleLeNet:

This paper proposes a deep Convolutional Neural Network architecture. This architecture is the improved utilization of computational resources inside the network. It is designed in a way of keeping the computational budget constant while increasing the depth and width of the network. The architectural decisions were based on the Hebbian principle and the intuition of multi-scale processing. GoogleLe Net is a 22-layer deep network based on this architecture used in ILSVRC14.

Residual Networks:

The Learning process of Deep Convolutional Neural Networks is difficult. In-order to ease the training, a Residual Learning framework is presented. Here the layers are reformulated as residual functions with respect to the input layers. These networks are easier to optimize and can gain accuracy by increasing depth. These networks are called Resnet. A 152 Layer Resnet was presented in Image Net challenge 2015 and won first place in classification task. This 152 layer Resnet has lower complexity compared to 16 layer VGG network.

III. Preliminary Results

Dataset

We are working on collecting Data from the Internet that is similar to the COCO Dataset but with a lesser number of classes and lesser number of training data.

A. Comparing VGG16, ResNet50, InceptionV3 Model Accuracies with custom data:

VGG16:

We imported the VGG16 Model from tensorflow.keras.applications.vgg16. Then we saved the Model with the Output Layer. Then we predicted the accuracy of the Image "car.jpg" using the VGG16 Model. The Output turns out to be a Sports Car with accuracy of 85.15%.

Residual Network:

We imported the ResNet50 model from tensorflow.keras.applications.resnet50. Then we saved the Model with the Output Layer. Then we predicted the Accuracy of the Image "car.jpg" using the ResNet50 Model. The Output turns out to be a Sports Car with accuracy of 85.95%.

Google LeNet:

We imported the InceptionV3 model from tensorflow.keras.applications.inception_v3. Then we saved the Model with the Output Layer. Then we predicted the Accuracy of the Image "car.jpg" using the InceptionV3 Model. The Output turns out to be a Sports Car with accuracy of 81.90%.

IV. Transfer Learning

Transfer Learning is a process in which we use an already Pre trained Model as the starting point for achieving similar task(similar objects detection).

By using Transfer Learning, we can achieve more Performance, fast Convergence, less Computational Power, more initial accuracy.

Now, we are gonna practically prove the above points and verify them.

The Codes and Implementation can be found here.

V. Workflow

We decided to train an Object Detection Model using Transfer Learning from a Pre Trained Model.

A. Collecting Data

We collected Data from the Internet and created our Data Set for training and testing the Models. We 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.

B. Annotating

After collecting the data, we started annotating the Images and drew Bounding Boxes for each object in the given Image of Training Data and Test Data. For this we used the tool 'labelimg' which is an 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.

C. Generating TF Records

We decided to use Tensor Flow for our Object Detection Model. Since Tensor Flow models train on Tensor Flow Records, we have generated TF Records(A binary format file which enhances the efficiency of Models while training) of our Custom DataSet.

D. Pre-trained Model Info

We used Tensor Flow Object Detection API to train our Transfer Learning Model. We chose SSD(Single Shot Detector) ResNet50 640*640 model previously trained on COCO2017 dataset and download the respective Pipeline Configuration file. SSD is chosen over other models becauase SSD takes less training time than any other models.

Initially, we thought of training three Transfer Learning Models each for the above specified models. But, we could not find Pipeline Configuration File for VGG16 and we had pipeline configuration file for InceptionV3 network but we don't have the weights of the pretrained model. Even if we decided to train the InceptionV3 model without Transfer Learning, we don't have enough Computational Resources for our model to converge.

So, we decided to go with SSD ResNet50 Model.

E. Setting up Colab Environment

- We wanted to take advantage of GPU provided by Google Colab. So we shifted to Colab for training our Model. For continuing our work in Colab, we installed different tools.
- We enabled GPU Runtime for Colab. First we installed Tensor Flow in the Notebook. After that, we cloned the tensorflow models Repository from GitHub into the Colab Contents.
- We also downloaded the pretrained model(SSD ResNet50) configuration file into Colab Contents.
- For working with Configuration Files and Training Parameters, TensorFLow Object Detection API uses Protobuf (Protocol Buffers). We installed the same.
- Coming to Tensor Flow Object Detection API, it is not installed via TF installation. We had to separately install the API using some commands.
- After this, we tried to train the Model using our collected Data but we got some errors from incompatible Library Versions. We found the incompatible Libraries and reinstalled the Compatible Versions of those Libraries.
- After setting up our whole environment and tools, we started training our Object Detection Model using the training commands.

VI. SSD ResNet50 Architecture

We 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.

VII. Challenges Faced

- We spent so much time on finding and downloading the Data because we wanted our model to train perfectly.
- Coming to Labelling the Images, we manually labelled the Images using LabelImg tool.
- For generating TF records for train and test data, we used some functions from TensorFlow. Some of these are not compatible with our present TF version. Some are only available for TF Version 2 and some for TF Version 1.
- It took so much time for resolving and updating the functions from TFv1 to TFv2.
- Due to lack of working GPU in our Team, at first it took so much time for us to train the data on a CPU. Then for training the model we shifted to Colab which includes a GPU. We got the GPU after many attempts and the allocated Ram is very less. So we had to decrease the Batch Size to 4(Otherwise it crashed).
- We faced many errors while working in Colab due to incompatible versions. It took many attempts to find the suitable versions from Internet Forums.

VIII. Training Results

We started Training the SSD ResNet50 Model using our Train Data for 8000 steps with a batch size of 4 in the Colab-GPU-Notebook. 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 in the Notebook. We got the Plots for Classification Loss, Localization Loss, Regularization Loss, Total Loss for every 100 steps up to 8000 steps. The results are as follows:

For each plot, X axis represents the Number of Steps and Y axis represents the Loss Value.

A. *Total Training Loss*

The Model almost converged for 8000 steps, which can be verified from the below plot of Total Loss for 8000 steps.

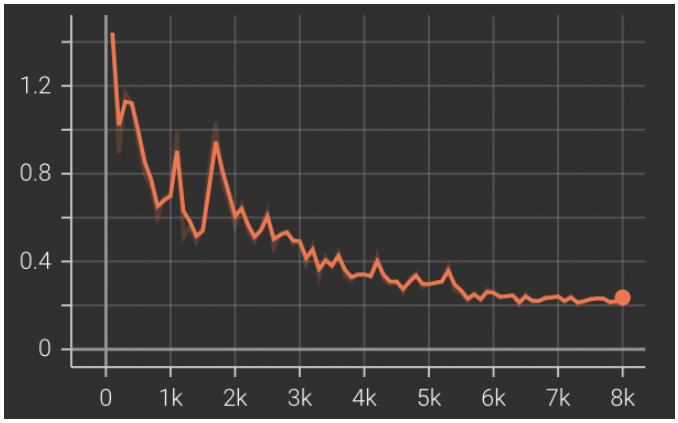

Fig. 1. Total Training Loss

B. *Classification Loss*

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

Classification Loss vs Number of Steps is:

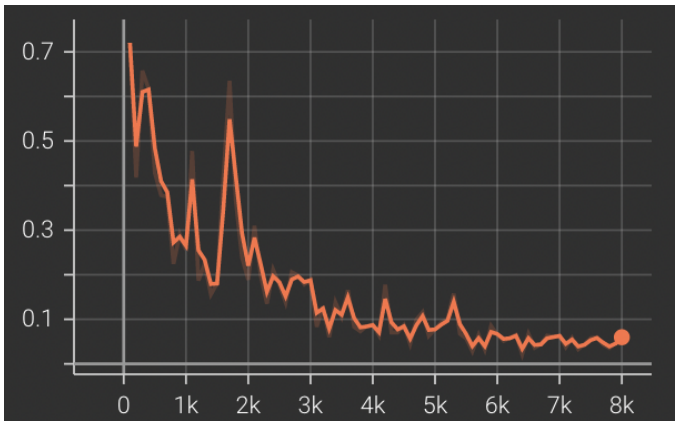

Fig. 2. Classification Loss

C. *Localization Loss*

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

Localization Loss vs Number of Steps is:

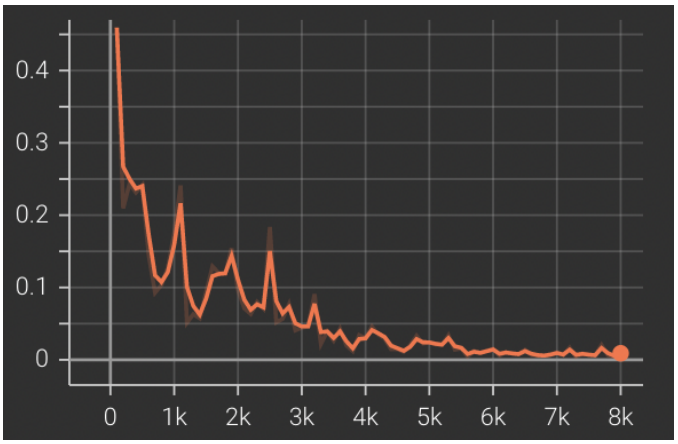

Fig. 3. Localization Loss

D. *Regularization Loss*

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

Regularization Loss vs Number of Steps is:

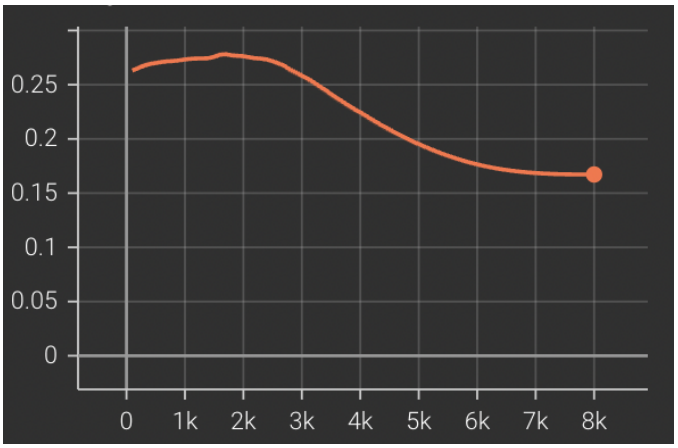

Fig. 4. Regularization Loss

E. *Learning Rate*

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

This is visualised from the below plot.

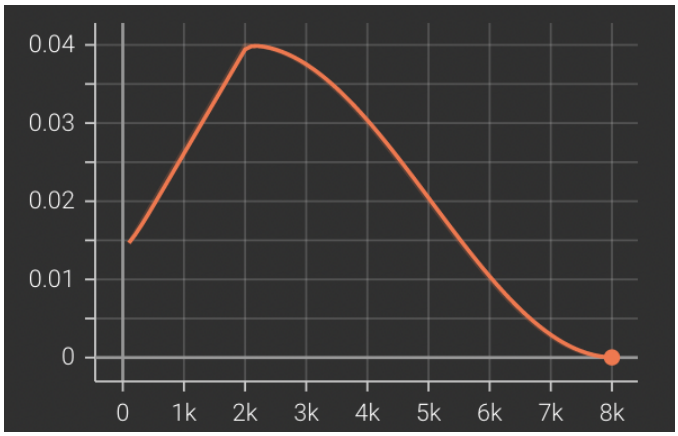

Fig. 5. Learning Rate

IX. Testing Results

After the training we tested the Model from our collected Test Data. The Results are as follows:

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.

Fig. 6. Car 1

It predicted the below car with 90% accuracy.

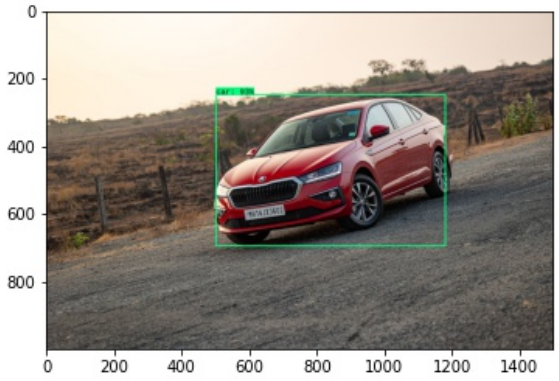

Fig. 7. Car 2

The Model predicted the below Cat Image with 95% accuracy.

Fig. 8. Cat 1

These type of cats are less in Training Data. So it predicted the below Image as Cat with 60% accuracy.

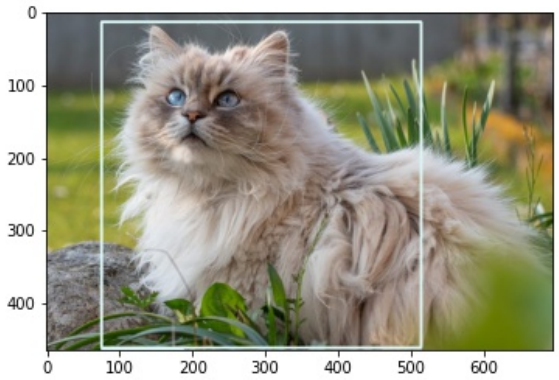

Fig. 9. Cat 2

Similar to Cars, Motor Bike does not have many variations, so it predicted with 89% accuracy.

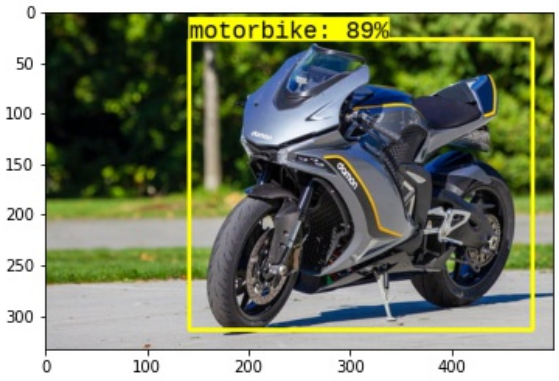

Fig. 10. Motor Bike 1

It predicted the below Image as Motor Bike with 98% accuracy.

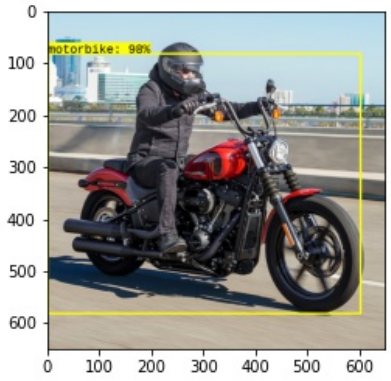

Fig. 11. Motor Bike 2

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.

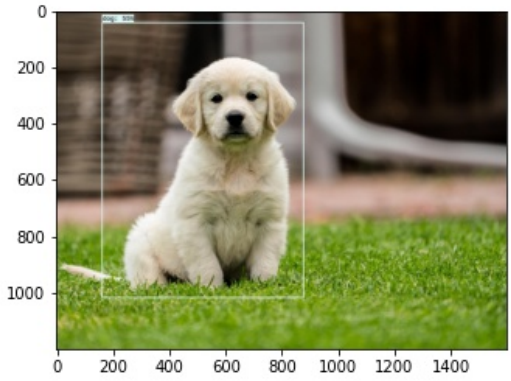

Fig. 12. Dog 1

X. 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.

References

[1] A Comprehensive Survey on Transfer Learning by Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Senior Member, IEEE, Hui Xiong, Fellow, IEEE, and Qing He.

[2] Object Detection and Tracking for Community Surveillance using Transfer Learning by Gayatri Sasi Rekha Machiraju, K.Aruna Kumari and Shaikh Khadar Sharif.

[3] VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION by Karen Simonyan and Andrew Zisserma.

[4] Going deeper with Convolutions by Christian Szegedy, Wei Liu, Yangqing Jia.

[5] Deep Residual Learning for Image Recognition by Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun Microsoft Research.