Traffic Sign Detection and Recognition

Ankit Rokde (153059002)
Ashish Rawat (163050011)
Minali Upreti (16305R002)
Saket Bhojane (163050063)

April 28, 2017

Chapter 1

Objective

The objective of our project is to detect and recognize street signs present images using machine learning. This project can be an important utility for intelligent transport system (e.g. self driving cars). The images captured from car cameras are generally blurred and corrupted because of motion of vehicle and atmospheric turbulence.

Our problem statement clearly consists of two parts namely:

1. Detecting presence of traffic signs in the input image
2. Recognizing the detected traffic signs if any

Challenges

- Image blurring and Gaussian noise corruption due to vehicle motion.
- Weather and different geographic conditions.
- Varying types of street signs.
- Street sign can be disoriented, faded or damaged.
- Objects with same color and shape in surroundings of the street sign.

Chapter 2

Dataset

We are using dataset from **German Traffic Signs Recognition Benchmark**[3]. This data set has 43 classes corresponding to street signs in Germany. It has 39209 images in total. These images are just of the street signs without any surroundings or background. However these signs are captured from various atmospheric and lighting conditions and various places in Germany making it challenging to predict the street sign when given to a trained model. The number of images in each class vary from 210 to 2250. The images are in PPM format of varying sizes from 15x15 pixels to 250x250 pixels not necessarily in square shape. The images have some space between boundary and the actual sign so that edge based approaches can be used. The sign may not be exactly at the center of the image. The annotation corresponding to an image is provided as the class of the sign present in the image and relative co-ordinates of the bounding box enclosing the street sign.

Chapter 3

Approaches

Since we are dealing with image classification problem CNNs are the way to go. Here are few approaches that we tried for our problem statement.

3.1 Custom Built CNN

We implemented our own custom CNN to solve this problem. We normalized our input images to size 32x32 to feed it to our network. We used one hot encoding for our classes. In our architecture we used 6 convolution layers and 2 fully connected layers. We used maxpooling after every 2 convolution layers. We used stochastic gradient descent with loss function as categorical cross entropy.

3.2 YOLO

Figure 3.1: YOLO architecture[4]

In a single evaluation the YOLO[2] neural network finds out the class probablities and the associated bounding boxes in an image. YOLO is capable of learning a general classifier that not only classifies the genre of the training data but is also capable of finding other classes of similarly associated images beacuse it sees not only the local region but also the entire image and store the contextual information about classes.

The given image is divided into SxS grid cells. A cell is responsible for identifying an object if its center lies in it. Each cell predicts some bounding boxes and associated confidence levels. Each bounding box gives 5 predictions namely x,y,w,h and the confidence. The pair (x,y) constitutes the co-ordinates represent the top left corner of the bounding box which lies relative to the grid cell. Here w and h corresponds to the width and hieght of the box.

3.2.1 Network Design

The first few convoluational layers extract out the features of the image while the layers near the end which are fully connected predict the probability and the co-ordinates of the location of the object. This network has 24 convolutional layers followed by two fully connected layers. The final output of the network is the 7x7x30 tensor of predictions. It is pre-trained on Image-Net 1000 class competition dataset.

3.2.2 Preparing the dataset for YOLO

We first converted the ppm images to jpg images using the PIL module in python. Then annotation data was extracted from csv file and saved into separate text files for each image. The annotation dataset contains the location coordinates of the traffic sign in the image. There were 43 classes, but some of the classes did not have quite enough images for YOLO to learn anything useful, so we also trained the YOLO network by combining these 43 classes into 4 different classes.

3.3 Inception architecture[4]

Figure 3.2: Inception architecture v3[4]

Google introduced its own architecture for image recognition and named it Inception v3. We have used this Inception v3 network in our model available from tensorflow[1] library. The training of Inception v3 is done on ImageNet data from 2012 which have about 1000 odd classes.

The common approach when using a standard model is transfer learning. In transfer learning, which means we are starting with a model that has been already trained on another problem. We then retrained it on a similar problem. Deep learning from scratch can take days, but transfer learning can be done in short order. We used the same network, but train it again to recognize small number of classes based on our own examples.

The script used loads the pre-trained Inception v3 model, removes the old final layer, and trains a new one on the traffic sign images.

Inception was not trained on any of these street signs, originally. However, the kinds of information that make it possible for it to differentiate among 1,000 classes are also useful for distinguishing other objects. By using this pre-trained network, we are using that information as input to the final classification layer that distinguishes our street sign classes.

Convolution, max-pooling, and dropout layers are repeatedly applied to the tensors, and the result is the logits variable which gives the predictions.

This model achieves 5.64% top-5 error while an ensemble of four of these models achieves 3.58% top-5 error on the validation set of the ImageNet whole image ILSVRC 2012 classification task.

Chapter 4

Results

4.1 Custom Built CNN

We did not obtain very satisfactory results with our custom built CNN for our 42 classes. It was performing satisfactorily for small number of classes but performed poorly with our 43 class data set.

4.2 YOLO

We trained the YOLO framework with our own data set for 80000 iterations. In this data set we trained real life images obtained from images captured from German Traffic Sign Detection and Recognition benchmark. We tried prediction on test images from the same data set. For many weights it was not able to predict the signs in the image. Here is a descent prediction we could obtain from it.

Figure 4.1: A successful traffic detection of traffic sign from YOLO

4.3 Inception architecture

This was the best model we could obtain for our purpose. We trained it with the benchmark data set that we described before and obtained the following results.

Figure 4.2: Training accuracy for 4000 iterations

Figure 4.3: Cross Entropy for 4000 iterations

Figure 4.4: Validation accuracy for 4000 iterations

Figure 4.5: Training accuracy for 8000 iterations

It could also predict traffic signs present in images captured from car cameras. However it was not performing very well when there were multiple traffic signs in the same image.

Figure 4.6: Cross entropy for 8000 iterations

Figure 4.7: Validation accuracy for 8000 iterations[4]

Figure 4.8: Training accuracy for 16000 iterations

Figure 4.9: Cross entropy for 16000 iterations

Figure 4.10: Validation entropy for 16000 iterations

Chapter 5

Conclusion

Our goal was to learn CNN from a practical perspective. Traffic sign detection and classification is a non trivial and challenging problem. We learned about state of the art techniques and CNN architectures though this project. Designing our own CNN architecture was found out to be quite challenging. We got to know why fine tuning is a better approach while doing image classification.

Bibliography

[1] Tensorflow: An open-source software library for machine intelligence. https://www.tensorflow.org/. [Online; accessed 20-April-2017].

[2] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 779–788, 2016.

[3] Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. The German Traffic Sign Recognition Benchmark: A multi-class classification competition. In *IEEE International Joint Conference on Neural Networks*, pages 1453–1460, 2011.

[4] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 2818–2826, 2016.