CIS 520 Spring 2018, Project Proposal

Recognizing German Traffic Signs

Team Members:

Sandeep Dcunha (PennKey: `sdcunha`; Email: `sdcunha@upenn.edu`)
Anton Khabbaz (PennKey: `akhabbaz`; Email: `akhabbaz@seas.upenn.edu`)
Connor Callahan (PennKey: `cocal`; Email: `cocal@sas.upenn.edu`)
Semih Canturk (PennKey: `canturks`; Email: `canturks@seas.upenn.edu`)

Assigned Project Mentor: Nikolaos Kolotouros

Team Member Contributions:

Team Member	Contributions
Sandeep Dcunha	Wrote sections for project proposal and report Introduced FCN code and added relevent functionality, as well as debugging Wrote analysis code as well as code for preprocessing and for FCN-CNN pipeline
Anton Khabbaz	Running the FCN data Some Debugging of the FCN code Analyzed some of the FCN data for problems Wrote sections of the Report and presented
Connor Callahan	Wrote code for CNN Built CNN Wrote sections for project proposal Wrote sections for project report
Semih Canturk	Helped write code for CNN Wrote sections of project proposal Wrote sections of project report

Code Submission: `https://github.com/csc2168/CIS_520`

1 Introduction

Our group was interested in learning more about computer vision and, more specifically, the development of convolutional neural networks (CNN) to solve these problems. The detection and recognition of traffic signs is an area where this approach has a lot of traction and practical importance in the development of mobile phone applications and automated cars.

We view this project as an opportunity to gain experience in computer vision and use of machine learning in automation. Using a publicly available benchmark dataset, we attempted to learn sign recognition using a variety of CNN architectures.

2 Related Work

Many institutions have attempted to make quick, reliable algorithms for detecting traffic signs. Particular difficulties arise when natural variations from lighting and weather arise that may not have had a strong signal in a training set. Computational speed is another barrier in that a car must be able to make decisions such as adjusting speed or braking quickly after encountering a sign [2]. Here we disregard the limitations on any given sensor or frame rate.

There has been a variety of work going into creating reliable CNN algorithms for traffic sign detection with recent results able to pass 99% accuracy in both detection and recognition in standard weather conditions when specifically observing signs with red contours [1]. The problem is often formulated as one of two types. The first is direct detection problem where the position of the sign is found with regression while the identity of the sign is determined through classification, with each of these mechanisms in convolutional layers. The second is a regional proposal method where the model first generates several candidate regions regardless of their classes, and then predicts object positions and class for each. Generally, the former is faster while the latter is more accurate[3].

3 Data Set

Our dataset was the German Traffic Sign Detection/Recognition Benchmark (GTSDB/GTSRB) [5]. Both sets include 43 classes of signs labeled 0..42. The GTSBD included 900 images taken from a sensor that include at least one, if not several, images of traffic signs all of size 1360 by 800 pixels with annotations and identifications in .ppm format. Each image has identified a set of regions where the traffic signs are found together with a integer representing the class These are located in a file called `gt.txt`. The GTSRB included 50,000 images of individual traffic signs of size 15 by 15 to 250 by 250 pixels along with annotations and class identifications.

4 Problem Formulation

We used two different approaches with convolutional neural networks to detect and identify signs: a fully convolutional network (FCN) to identify and classify the images and an FCN to identify the regions within an image where there was a sign, and then classify the pixels as either a sign or not. Those regions were then cropped and sent to a classification CNN to detect sign type. Performance in both cases was measured using mean intersection over union (IoU) that measures the ratio of intersecting pixels that are the same in both predicted and annotated images and the union of all those pixels. The IoU (Equation 1) measure applies only to pixels of the same class. The mean IOU (Equation 2) for each image is then calculated by taking the mean of the IOU for every class in To accomplish this,

$$\text{IOU}(c) = \frac{\sum_{i,j} \mathbf{1}(\text{pred}[i,j] = c \wedge \text{gt}[i,j] = c)}{\sum_{i,j} \mathbf{1}(\text{pred}[i,j] = c \vee \text{gt}[i,j] = c)} \tag{1}$$

Figure 1: Intersection over Union calculation given two images, pred and gt and class c.

$$\begin{aligned} S &= \{pred[i,j] \forall i,j\} \cap \{gt[i,j] \forall i,j\} \\ \text{Mean IOU} &= \frac{1}{|S|} \sum_{c \in S} \text{IOU}(c) \end{aligned} \tag{2}$$

Figure 2: Mean Intersection over Union

the classification of the CNN is used to label the pixels previously identified as having a sign to now being of a specific class.

Preprocessing included image rescaling, file format conversion (from .ppm to .png), assigning traffic sign boundaries a uniform pixel for their class, and rotations and reflections of images.

5 Algorithms

5.1 CNN

Convolutional Neural Networks (CNN) were chosen as our main machine learning algorithm (given that FCNs are a specific type of CNN) due to their ability to distinguish patterns in images - hence their popularity in computer vision problems.

The main practical difference between a standard ANN (Deep MLPs, for example) and CNNs is that CNNs are shift-invariant: To a standard ANN trying to classify an image, for example an object with a white background, where the object is placed within the image matters: two images where an object is on the opposite sides in each respective image would be less likely to be classified into the same class. In the meantime, it would be much easier (aka less computational power and training data) for a CNN to recognize that both images represent the same object in different locations. CNNs are also good at learning spacial information due to the convolution operations which they learn.

A CNN usually consists of groups of convolutional layers followed by a pooling layer (this grouping could be repeated multiple times depending on desired depth), and the final layer a fully connected layer. The convolutional layers consist of groups of neurons which classify only within their receptive layers by being tightly connected to each other and sharing parameters - this allows them to take advantage of useful regions in images. The pooling layers, on the other hand, are used for downsampling before moving on to new layers. Finally, the higher-level classifications are done using the fully connected layer at the end.

5.2 FCN

The Fully Convolutional Network (FCN) is a convolutional network, much like the CNN. Unlike the CNN, the FCN is fully convolutional, that is it doesn't have any fully connected layers at the end. The CNN uses the fully connected layers to squash its featurespace into a multiclass prediction. On the other hand, the FCN can produce a result of any size, generally (as in this case), one of the same size as the label. This allows for class segmentation in images rather than multiclass predictions.

6 Experimental Design and Results

6.1 Data Preparation

We were provided two datasets, German Traffic Sign Recognition Benchmark (GTSRB) and the German Traffic Sign Detection Benchmark

(GTSDB). We performed preprocessing on both the datasets.

6.1.1 FCN

For the FCN, the GTSRB (detection) dataset was used. Each image was converted into a .png format. Furthermore, the labels provided as bounding box coordinates in a CSV file was converted to single channel images of the same size as its corresponding color image with the pixels inside the bounding box set to its category and background pixels set to 43 (as pixel values 0-42 represent classes). We also created a binary version of these annotations where background pixels are 0 and sign pixels are 1. Note that, while there might be multiple bounding boxes for an image, the labels for each image were combined into one corresponding annotation image.

The 900 files of the GTSDB data with full labels was divided into three sections: 20% or 149 images was reserved for testing; 64% was used for training and 16% was used for validation.

Essentially two models were made: the ClassFCN model consisted of 44 classes, the 43 sign classes and one class that consisted of background counts; the BinaryFCN model consisted of a sign class and a background class.

Each model was trained for over 4000 iterations. The loss function was the softmax cross entropy function and in each case the losses were about 0.001. The models each took some two hours to train using a NVidia 1080 Ti GPU.

6.1.2 CNN

The CNN was able to leave training data into a .ppm format. When data was read in, options were set to enable width shift, height shift, shear, and zoom ranges of 20% as well as horizontal flips. Data was standardized to 32 by 32 pixels, though the original dataset had a large range of potential sizes.

The data was split into training, validation, and test datasets. There were 87807 training images, 7842 validation images , and 12,630 test images. These were identified with their class both by a directory structure (except the test data) and truth information in a .txt file.

6.2 Model Architecture and Selection

6.2.1 FCN

In our FCN we have five layers; the first consists of two pairs of convolution, ReLu rectification followed by pooling. The second layer has the same stages as the first. The third layer consists of four convolution, ReLu pairs followed by a pooling stage. The fourth layer is identical to the third layer. The fifth layer consists of the four convolution, ReLu stages.

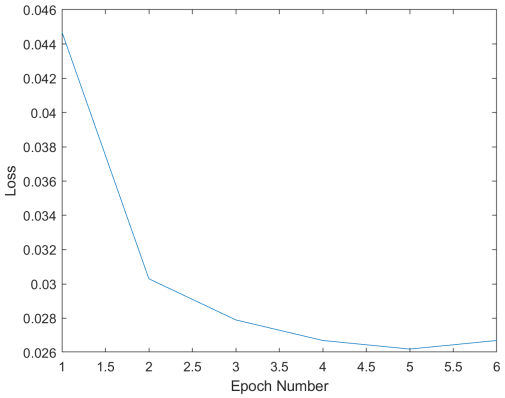
The model was pretrained using the FCN used in [4]. The model used was `http://www.vlfeat.org/matconvnet/models/beta16/imagenet-vgg-verydeep-19.mat`.

6.2.2 CNN

The CNN has three convolutional layers as well as a full connection. All layers ReLU units with 32 outputs in the first layer, 64 outputs in the second layer, and 128 outputs in the second layer, with each layer followed by pooling. The model is then flattened before reaching the full connection, composed first of 1,024 ReLU units followed by 43 units with a soft maximum activation. Of the 1,024 ReLU units in this full connect, half are ignored by dropout.

The model is compiled using the Adam optimizer for stochastic optimization. It used accuracy for its performance metric and binary cross-entropy for its loss function. The training used batch sizes of 32, 5 epochs, 10,000 training steps per

epoch, and 2,000 validation steps per epoch. We saw that the loss started to go back up after the 5th epoch. The model took about four hours to train.



6.3 Model Performance and Evaluation

The advantage of the binary class is that the sign class has about 40 times as much data as the average sign class in the ClassFCN. Also there are many fewer parameters since there are fewer classes

The final CNN model was able to correctly identify 9,830 test images as their true class of 43 classes, giving about 78% accuracy. Given that this model was trained from scratch and there is a large number of classes, this was a reasonably predictive model for the scale of this project.

The Multiclass FCN had a mean IOU of 0.19 and the Binary FCN had a mean IOU of 0.81. The histogram in Figure 3 shows the IOU of each image. Most do very well, but there are images for which the prediction was not good. However, that figure is misleading and biased, as there is a large class imbalance in the data. Therefore, the per class IOU, seen in Figure 4 displays a more interesting result. This demonstrates that for most signs, the Binary FCN identified the sign correctly. Note that the shapes of the two histograms is described mostly by the sign class, but

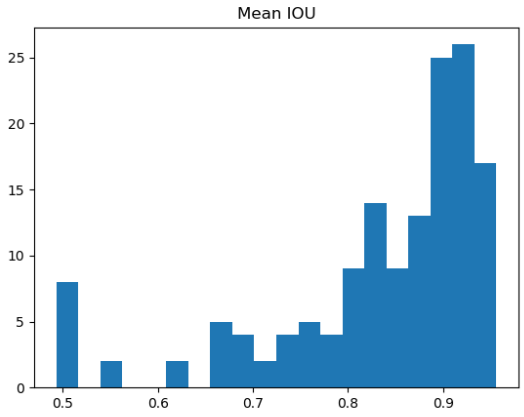


Figure 3: Binary FCN Mean IOU histogram

the background class increases the mean IOU's value.

7 Conclusion and Discussion

Although our general predictive power is not quite as high as we had hoped, we do see that our FCN model is able to detect appropriate boundaries for sign images and that its classification is significantly better than guessing amongst the 43 classes. Our CNN model shows strong performance under 0-1 loss for multiclass classification, correctly identifying the right sign type, amongst 43 classes 78% of the time.

One reason that the Binary FCN classification was not optimal was that the signs accounted for usually a very small part of the image, often less than 1%. This is the class imbalance issue in object recognition and there are CNN's and re-weighting that minimize this.

If we were to continue this project further, we would try to rectify the class imbalance and improve the integration between the FCN and the CNN. We would also try to apply more thorough preprocessing to handle varying sign image sizes more adequately. Finally, we would try to find

solution methods able to handle the relatively small amount of training data for our FCN.

Acknowledgments

Thanks goes out to the tutorial GitHub repository maintained by Venkatesh Tata for using Keras and outlining a simple binary image classification CNN architecture [6].

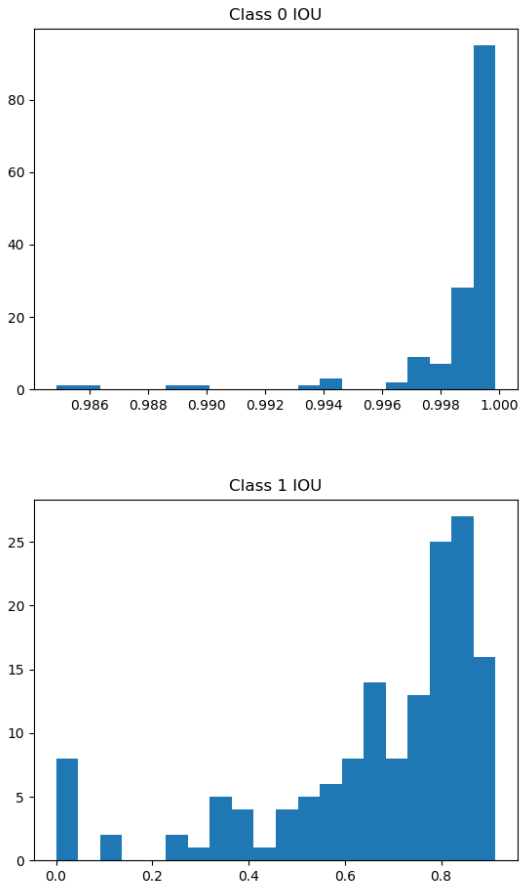


Figure 4: Binary FCN Per Class IOU

References

[1] Cnn design for real-time traffic sign recognition. *Procedia Engineering*, 201:718 – 725, 2017. 3rd International Conference Information Technology and Nanotechnology.

[2] Laura Bliss. How to teach a car a traffic sign. *CITYLAB*, 2017.

[3] Hee Seok Lee and Kang Kim. Simultaneous traffic sign detection and boundary estimation using convolutional neural network. 2018.

[4] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[5] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural Networks*, (0):–, 2012.

[6] Venkatesh Tata. Convolutional-Neural-Network. `https://github.com/venkateshtata/-Convolutional-Neural-Network`. Accessed: 2018-04-20.