MACHINE LEARNING ENGINEER NANODEGREE

Capstone Proposal

Anuja Tupe

October 12th, 2018

German Traffic Sign Classification Using Keras

DOMAIN BACKGROUND

Self Driving Cars are disrupting the automobile and transportation industry. It is an active ongoing research field. Lot of companies like Google, Uber, Volkswagen etc. are working in this field. The area of study for self driving cars is broadly applicable to several industries like basic computer science, computer vision, mechanical engineering, machine learning, robotics. Self driving cars pose many real world problems that can be solved using computer vision and machine learning. As I work in the silicon valley, solving these problems can have a real impact in this field.

Following is one of the links to an academic paper where machine learning was applied to solve this problem - http://yann.lecun.com/exdb/publis/pdf/sermanet-ijcnn-11.pdf

PROBLEM STATEMENT

Traffic signs are one of the most essential component for regulating traffic. They assist people in navigating the vehicle with discipline thus avoiding any chaos or accidents. One such problem that I am interested in solving is classifying traffic signs. If traffic signs are not correctly classified by driverless cars, it can pose a safety threat to people in and around the driverless car. It can lead to accidents and possibly can be fatal.

DATASETS AND INPUT

We will be using the dataset provided by German Traffic Sign Recognition Benchmark. The dataset can be downloaded from here - http://benchmark.ini.rub.de/?section=gtsrb&subsection=dataset#Downloads.

This dataset includes 50000 images. The images are colored images and are provided in the ppm format. The dimensions of all the images are not the same.

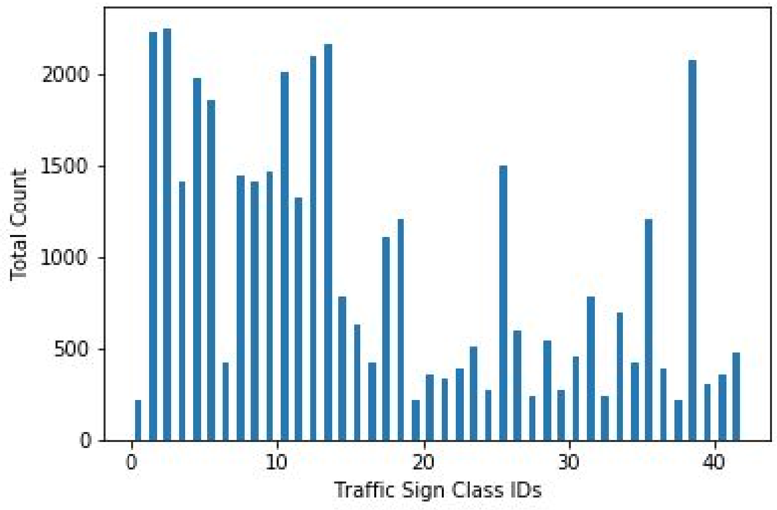
These images are split into train dataset and test dataset. There are 39209 images in the train dataset and 12630 images in the test dataset. There are 43 types of traffic signs that we will be training on. The images vary in lot of aspects - brightness, sizes, color. Each traffic sign image has a class ID (integer from 0 to 42) associated with it. This information is stored in the csv files.

The train dataset has folders for every type of traffic sign (43 folders). Each folder has the images of the respective traffic sign and a CSV file which maps the image name with a class ID.

In order to get meaningful information about what each traffic sign means, we will create a CSV file with the mapping between class IDs and class labels. We will use the CSV files from the 43 folders to create our class labels.
http://www.gettingaroundgermany.info/zeichen.shtml - We will use this website to get meaningful english names for each of the 43 traffic signs. For example - Class ID 1 is for traffic sign “Speed Limit 30”.

The test dataset has assorted images of traffic signs. It also has a CSV file which maps the image name with the correct class ID.

The dataset that we will be using is imbalanced. The plot below represents the frequency of the every traffic sign. The X axis represents the class IDs for the traffic sign images. The Y axis represents the total count of each of the traffic sign class ID.



Class 2 has the highest frequency and the frequency is 2250. Class 0 has the lowest frequency and the frequency is 210

http://benchmark.ini.rub.de/?section=gtsrb&subsection=dataset#Codesnippets - We will use the sample code snippet provided by the GTSRB website to read the images. I will modify this code to preprocess the image after we read the image. We will store this image in an array.

SOLUTION STATEMENT

The traffic sign recognition problem is a classification problem. Since, it involves images, Convolutional Neural Network (CNN) would be a great fit. To summarize, the solution will be to analyse the images and decide the preprocessing techniques that we would be using for the images. These preprocessed images will then be used for training the CNN model. Along with preprocessing the images, we will also augment our images to increase the size of our dataset. This will help train our model better and it will also reduce overfitting. Finally, we will test it on the test dataset provided.

We will also test the model by picking some german traffic sign images from Internet. We will print out the label predicted by our model for the test image.

BENCHMARK MODEL

In 2011, the IJCNN (International Joint Conference on Neural Networks) competition took place for traffic sign recognition. The IJCNN 2011 competition results can be a benchmark model.

http://benchmark.ini.rub.de/index.php?section=gtsrb&subsection=results&subsubsection=ijcnn - This link has the results of the competition. It has a table of teams that participated and their score with the highest score on top.

Additionally, we will be trying out different CNN models and we will try to beat the performance every time we try a new model.

EVALUATION METRICS

Since, our dataset is imbalanced, we will be using F-beta score as the evaluation metrics. F-beta score is the weighted harmonic mean of precision and recall. The value of beta that we will be using is 1.0. This means that precision and recall are equally important for the model. Precision is the exactness or quality of the model and recall is the completeness of the model. Precision is true positives divided by the sum of true positives and false positives. Recall is number of true positives divided by the sum of true positives and false negatives.

Confusion matrix is another metric that can be used to summarize the performance of the classification algorithm. We will calculate the confusion matrix. However, since, the number of labels are high (43 class IDs) it can be a little cumbersome to present the confusion matrix and make sense of it.

We will get some random german traffic sign images and check if the predicted class label is correct.

PROJECT DESIGN

Data Split for Training and Validation

Since, our dataset is imbalanced we have to make sure the dataset split for training and validation is done in such a way that a considerable amount of images for each class ID is given for training the model. For example - In our case, Class 2 has the highest frequency and the frequency is 2250. Class 0 has the lowest frequency and the frequency is 210. During the test train split, it can happen that the train dataset has say only 10 samples of the class 0 images and has 1100 images of the class 2 images. This can lead to an inaccurate model, since the model is not trained well on the class 0 images. To solve this issue, we will use the StratifiedShuffleSplit method provided by sklearn. This method will help us create a train validation set which is equally balanced in terms of the classes.

Preprocessing

There are many preprocessing techniques available for images. We will use the histogram equalization technique. If you notice the images in the dataset, they seem to be very dark and blurry. Histogram equalization is used for improving the contrast of the image. In order to do this, we will first convert our image from RGB to HSV. We will then use the equalize_hist() method of skimage to equalize each channel in the image.

Cropping an image is another preprocessing step that we will execute. This helps in concentrating on the important part of the image and reduce the noise. We will crop the image to be in the square shape. The center part of the image is the important part. Hence, we will crop the image to be a square image considering the center of the image. The logic to do this is - Select the smaller side of the image. For example - If the dimension of the image is 40x50, we will select the smaller side i.e. 40. We will crop the image to be 40x40. In order to consider the center of the image, we will select 20 on the left side of the center and 20 on the right side of the center. Similarly, we will select 20 on the top of the center and bottom of the center.

Lastly, we will resize the images. All images are not of the same size. The dimensions of the images will vary. For example - Say one image can be of size 60x60 and the other image can be of size 80x80. We will resize the image. We will resize it to be of size 48x48.

Data Augmentation

There can be a lot of inconsistencies in images. For example - In one image, the traffic sign can be to the left in the image. In some other image, the traffic sign can be at an angle. There can be many such variations with respect to size, angles, contrast. Hence, it is a good idea to use data augmentation to train our model. In augmentation, we produce various versions of the same image and thereby increase the amount of information learnt by the model. This avoids the overfitting problem by training the model with different versions of the same image. Keras has an ability to generate variations of the images and we will use that for augmenting our data.

Model To Use

Since, our problem definition revolves around classifying images, we will use a neural network called Convolutional Neural Network (CNN). We will be using Keras for building CNNs. Keras is a neural network API written in Python. We will be trying out different CNNs for this problem statement. We can try and experiment with different convolutional neural network models for this project and compare the accuracies of each.

I am planning to use the following models and compare the performance -

1. A simple CNN model - I will know the details of which layers and how many layers I will be using, once I start working on it.
2. LeNet Model - This was one of the very first CNN which powered the field of neural networks. It was used to recognise hand written digits and was initially used in Banks.
3. A modified version of the LeNet model - Basically, I will try to improve/modify the lenet model performance for the given dataset by introducing additional layers or changing the input parameters for the model.

REFERENCES

1. http://benchmark.ini.rub.de/?section=gtsrb&subsection=news
2. http://www.gettingaroundgermany.info/zeichen.shtml
3. https://machinelearningmastery.com/metrics-evaluate-machine-learning-algorithms-python/
4. https://towardsdatascience.com/learning-rate-schedules-and-adaptive-learning-rate-methods-for-deep-learning-2c8f433990d1
5. https://becominghuman.ai/image-data-pre-processing-for-neural-networks-498289068258
6. https://stackoverflow.com/questions/47117179/histogram-equalization-skimage?rq=1
7. https://www.tutorialspoint.com/dip/histogram_equalization.htm
8. https://chsasank.github.io/keras-tutorial.html
9. https://www.quora.com/Why-are-CNNs-used-more-for-computer-vision-tasks-than-other-tasks
10. https://machinelearningmastery.com/custom-metrics-deep-learning-keras-python/
11. http://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html
12. https://stackoverflow.com/questions/35572000/how-can-i-plot-a-confusion-matrix
13. http://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html
14. https://towardsdatascience.com/accuracy-paradox-897a69e2dd9b
15. https://en.wikipedia.org/wiki/F1_score
16. https://machinelearningmastery.com/classification-accuracy-is-not-enough-more-performance-measures-you-can-use/
17. http://text-analytics101.rxnlp.com/2014/10/computing-precision-and-recall-for.html
18. https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/
19. https://medium.com/@sidereal/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5