Capstone Project

Machine Learning Engineer Nanodegree

Sarita Navuluru
10/24/2017

Definition

Project Overview

The increased focus and interest in atonomous road vehicles and driver assistant systems research in the last few years has created a wide interest in the problem of image classification of Traffic Signals [Udacity-Self Driving Car NN]

As seen in the above moving autonomous car, the system captures Traffic signals using a camera mounted (typically) on its roof. The movement could cause the captured image to be skewed in its orientation, rotated, blurred due to speed variation and deformed in its scale and translation. Environmental conditions such as rain, fog, bursty sunlight can cause distorted illumination, hazy, vague images. This makes the problem of correctly identifying Traffic Signal images a complex task.

Today Systems can learn to identify Traffic Signals using various AI techniques. Concepts, methods, and algorithms constituted from a combination of Deep Learning, Computer Vision and Optimization have emerged from their

math-research-oriented-highly-theoritical papers to full-fledged products developed by the prominent technology giants, given the increased computation power aka GPU and the availability of stored giant data that feed the systems. Deep learning methods particularly today superseed traditional computer vision and machine learning methods and are known to perform better than a human to correctly classify a Traffic Signal.

The “German Traffic Sign Recognition Benchmark” is a multi-category classification competition held at IJCNN 2011. It remains one of the most popular databases for exploring and classifying traffic signals. A comprehensive, lifelike dataset of more than 50,000 traffic sign images has been collected. It reflects the strong variations in visual appearance of signs due to distance, illumination, weather conditions, partial occlusions, and rotations. The images are complemented by several precomputed feature sets to allow for applying machine learning algorithms without background knowledge in image processing. The dataset comprises 43 classes with unbalanced class frequencies.[http://ieeexplore.ieee.org/document/6033395/]

Current work

Several techniques and algorithms have been proposed in this area - Traffic Signal Classification, such as Linear Discriminant Analysis -LDA , Support Vector Machines(SVM), histograms of oriented gradients (HOG) [Dalal and Triggs (2005)], Haar-like features, and color histograms [Viola, P., and Jones], perceptron networks, deep neural networks, ensemble classifiers, random forests, and Convolutional Neural Networks [http://benchmark.ini.rub.de/]

The technique used in the paper Spatial Transformer Networks **[Spatial Transformer Networks] won the competition.**

Method	Accuracy
two spatial transformers with idsia-like network (1)	99.67
single spatial transformer with idsia-like network (2)	99.46

winner of original contest: idsia network	99.46
single idsia-like network (3)	98.85
human performances (corresponding paper)	98.84

Convolutional Neural Network techniques, probabilistic latent semantic analysis based upon traditional handcrafted features extraction, and other algorithm based on k-d trees and random forest [Traffic sign classification using K-d trees and Random Forests , Human Performance, INI-RTCV] achieved significant accuracy as can be seen from the past competition results in 2011.

The final competition session that was held at IJCNN 2011. For results of the first phase of the competition, please see the IJCNN 2011 Competition result table.

TEAM	METHOD	TOTAL	SUBSET All signs
[3] IDSIA	Committee of CNNs	99.46%	99.46%
[155] COSFIRE	Color-blob-based COSFIRE filters for object recogn	98.97%	98.97%
[1] INI-RTCV	Human Performance	98.84%	98.84%
[4] sermanet	Multi-Scale CNNs	98.31%	98.31%
[2] CAOR	Random Forests	96.14%	96.14%
[6] INI-RTCV	LDA on HOG 2	95.68%	95.68%
[5] INI-RTCV	LDA on HOG 1	93.18%	93.18%
[7] INI-RTCV	LDA on HOG 3	92.34%	92.34%

Problem Statement

The problem is to to recognize the traffic sign from the images in the GTSRB dataset using Convolutional Neural Networks and the learnable component Spatial Transformer Network after treating the original images to a series of preprocessed transformations. This is a typical ‘Image Classification’ problem. Solving this problem is essential for self-driving cars to operate on roads.

Convolutional Neural Networks

In machine learning, a **convolutional neural network**(CNN, or ConvNet) is a class of deep, feed-forward artificial **neural networks** that has successfully been applied to analyzing visual imagery[wiki]

Further discussion of what a convolution is and the various building blocks of the network is performed in the Algorithm section.

The model is inspired by studying the following standards

Layer	Cireşan *et al.* [6]	Sermanet & LeCun [5]	Jin *et al.* [26]
0	Input (48×48, RGB)	Input (32×32, grayscale)	Input (47×47, RGB)
1	Conv (7×7, 100 maps)	Conv (5×5, 108 maps)	Conv (5×5, 70 maps)
2	Nonlinearity (not specified)	Nonlinearity (rectified *tanh*)	Max-pooling (3×3, stride 2)
3	Max-pooling (2×2)	Max-pooling* (2×2)	Nonlinearity (ReLU)
4	Conv (4×4, 150 maps)	Conv (5×5, 108 maps)	Local normalisation
5	Nonlinearity (not specified)	Nonlinearity (rectified *tanh*)	Conv (3×3, 110 maps)
6	Max-pooling (2×2)	Max-pooling (2×2)	Max-pooling (3×3, stride 2)
7	Conv (4×4, 250 maps)	Fully-conn.* (100 units)	Nonlinearity (ReLU)
8	Nonlinearity (not specified)	Nonlinearity (rectified *tanh*)	Local normalisation
9	Max-pooling (2×2)	Fully-conn. (100 units)	Conv (3×3, 180 maps)
10	Fully-conn. (300 units)	Nonlinearity (rectified *tanh*)	Max-pooling (3×3, stride 2)
11	Nonlinearity (not specified)	Fully-conn. (43 units)	Nonlinearity (ReLU)
12	Fully-conn. (43 units)	Softmax	Local normalisation
13	Softmax		Fully-conn. (200 units)
14			Nonlinearity (ReLU)
15			Fully-conn. (43 units)
16			Softmax

Metrics

The following metrics are used to judge the performance of the model. Please note, further details such as the type of loss and how it compares with other losses etc., are described in the algorithm and the implementation sections.

Loss

When the model makes a prediction,the loss function is computed. It measures the compatibility between a prediction (e.g. the class scores in classification) and the ground truth label. The data loss takes the form of an average over the data losses for every individual example.

There are different kinds of loss functions such as mean_squared_error, hinge etc.

The one chosen for this algorithm is the "Categorical Cross-Entropy". This has been ubiquotous with Convolutional networks used for image classification

I choose this loss function over others such as mean_squared_error because the ln() function in cross-entropy takes into account the closeness of a prediction and is a more granular way to compute error.

Accuracy

Test accuracy is the number of labels the model correctly classified by the total number of labels.

Learning Curve

The learning curve (training accuracies plotted over epochs/iterations) allows us to know if the model is overfitted or underfitted and whether there is redundancy in epochs.

Analysis

Data Exploration

The Dataset

The German Traffic Signal database can be downloaded from the official website http://benchmark.ini.rub.de/?section=gtsrb&subsection=dataset.

This project however uses the resized and cropped version of the original GTSRB dataset, provided by Udacity in pickle files here.

The dataset consists of three sets of images, one for training, one used for validation and one for testing. More details can be found in the jupyter notebook that implements the model.

Image and the set sizes

Each image is a 32×32×3 array of pixel intensities, represented as [0, 255] integer values in RGB color space. Class of each image is encoded as an integer in a 0 to 42 range.

The number of images in the training set are 39209. Validation set consists of 4410 images, and the test set consists of 12630 number of images.

Classes

There are 43 different classes of labels. The following table lists all the class names and the number of images in each class in the training set, validation set, and test set.

The following lists the classes, their train, test, and valid statistics in ascending order or Training data for each class.

Class ID	Sign Name	#Train	#Test	#Valid	%Train	%Test	%Valid
0	Speed limit (20km/h)	210	60	30	0.54%	0.48%	0.68%
37	Go straight or left	210	60	30	0.54%	0.48%	0.68%
19	Dangerous curve to the left	210	60	30	0.54%	0.48%	0.68%
32	End of all speed and passing limits	240	60	30	0.61%	0.48%	0.68%
27	Pedestrians	240	60	30	0.61%	0.48%	0.68%
41	End of no passing	240	60	30	0.61%	0.48%	0.68%
42	End of no passing by vechiles over 3.5 metric tons	240	90	30	0.61%	0.71%	0.68%
24	Road narrows on the right	270	90	30	0.69%	0.71%	0.68%

29	Bicycles crossing	270	90	30	0.69%	0.71%	0.68%
39	Keep left	300	90	30	0.77%	0.71%	0.68%
21	Double curve	330	90	60	0.84%	0.71%	1.36%
40	Roundabout mandatory	360	90	60	0.92%	0.71%	1.36%
20	Dangerous curve to the right	360	90	60	0.92%	0.71%	1.36%
36	Go straight or right	390	120	60	0.99%	0.95%	1.36%
22	Bumpy road	390	120	60	0.99%	0.95%	1.36%
6	End of speed limit (80km/h)	420	150	60	1.07%	1.19%	1.36%
16	Vechiles over 3.5 metric tons prohibited	420	150	60	1.07%	1.19%	1.36%
34	Turn left ahead	420	120	60	1.07%	0.95%	1.36%
30	Beware of ice/snow	450	150	60	1.15%	1.19%	1.36%
23	Slippery road	510	150	60	1.3%	1.19%	1.36%
28	Children crossing	540	150	60	1.38%	1.19%	1.36%
26	Traffic signals	600	180	60	1.53%	1.43%	1.36%
15	No vechiles	630	210	90	1.61%	1.66%	2.04%
33	Turn right ahead	689	210	90	1.76%	1.66%	2.04%
14	Stop	780	270	90	1.99%	2.14%	2.04%
31	Wild animals crossing	780	270	90	1.99%	2.14%	2.04%
17	No entry	1110	360	120	2.83%	2.85%	2.72%
18	General caution	1200	390	120	3.06%	3.09%	2.72%
35	Ahead only	1200	390	120	3.06%	3.09%	2.72%
11	Right-of-way at the next intersection	1320	420	150	3.37%	3.33%	3.4%
3	Speed limit (60km/h)	1410	450	150	3.6%	3.56%	3.4%
8	Speed limit (120km/h)	1410	450	150	3.6%	3.56%	3.4%
7	Speed limit (100km/h)	1440	450	150	3.67%	3.56%	3.4%
9	No passing	1470	480	150	3.75%	3.8%	3.4%
25	Road work	1500	480	150	3.83%	3.8%	3.4%
5	Speed limit (80km/h)	1860	630	210	4.74%	4.99%	4.76%
4	Speed limit (70km/h)	1980	660	210	5.05%	5.23%	4.76%
10	No passing for vechiles over 3.5 metric tons	2010	660	210	5.13%	5.23%	4.76%
38	Keep right	2070	690	210	5.28%	5.46%	4.76%

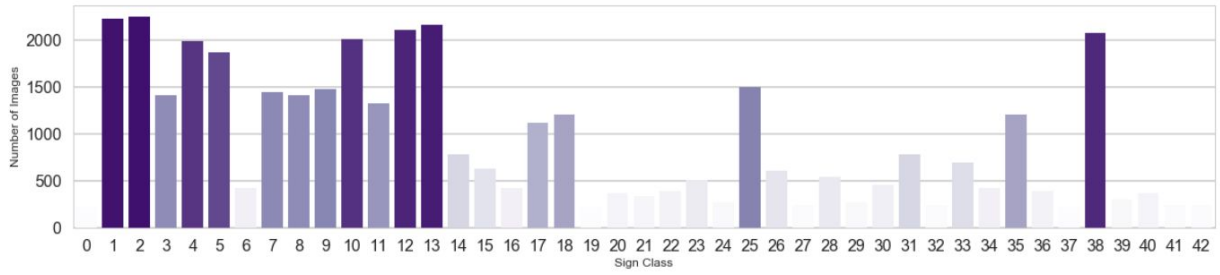
12	Priority road	2100	690	210	5.36%	5.46%	4.76%
13	Yield	2160	720	240	5.51%	5.7%	5.44%
1	Speed limit (30km/h)	2220	720	240	5.66%	5.7%	5.44%
2	Speed limit (50km/h)	2250	750	240	5.74%	5.94%	5.44%

The percentage of images in each class is equivalent in all three sets. Train and Test are more proportional to each other compared to the valid, which varies very slightly.

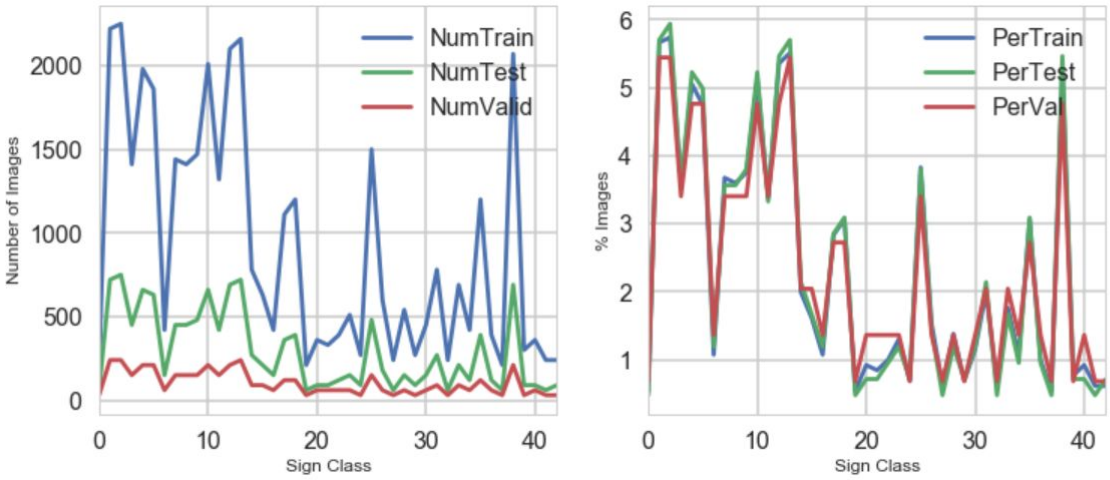
The sets are unbalanced across the classes. We see given the heat markings and the numbers that some classes are represented significantly better than the others.

Classes 1,2,4,5,10,12,13,25,38 relatively have high number of images
Classes 0,16,19-24, 27,29,32,39-42 have relatively very low number of images

Histogram Visualization of Images per Class in training set



Number and Percentage Plots for Train, Test, Validation sets



The above figure shows number of images for each sign class for each of the datasets. If we disregard the scale, the pattern of crests and troughs is the same in all the three sets.

The second graph shows the percentages across the three sets for each class. Between classes 19 to 24(approx), the validation differs slightly from the test and train plots. This shows that the test and train data are proportionally more similar compared to the validation data.

First Image of each class from the dataset

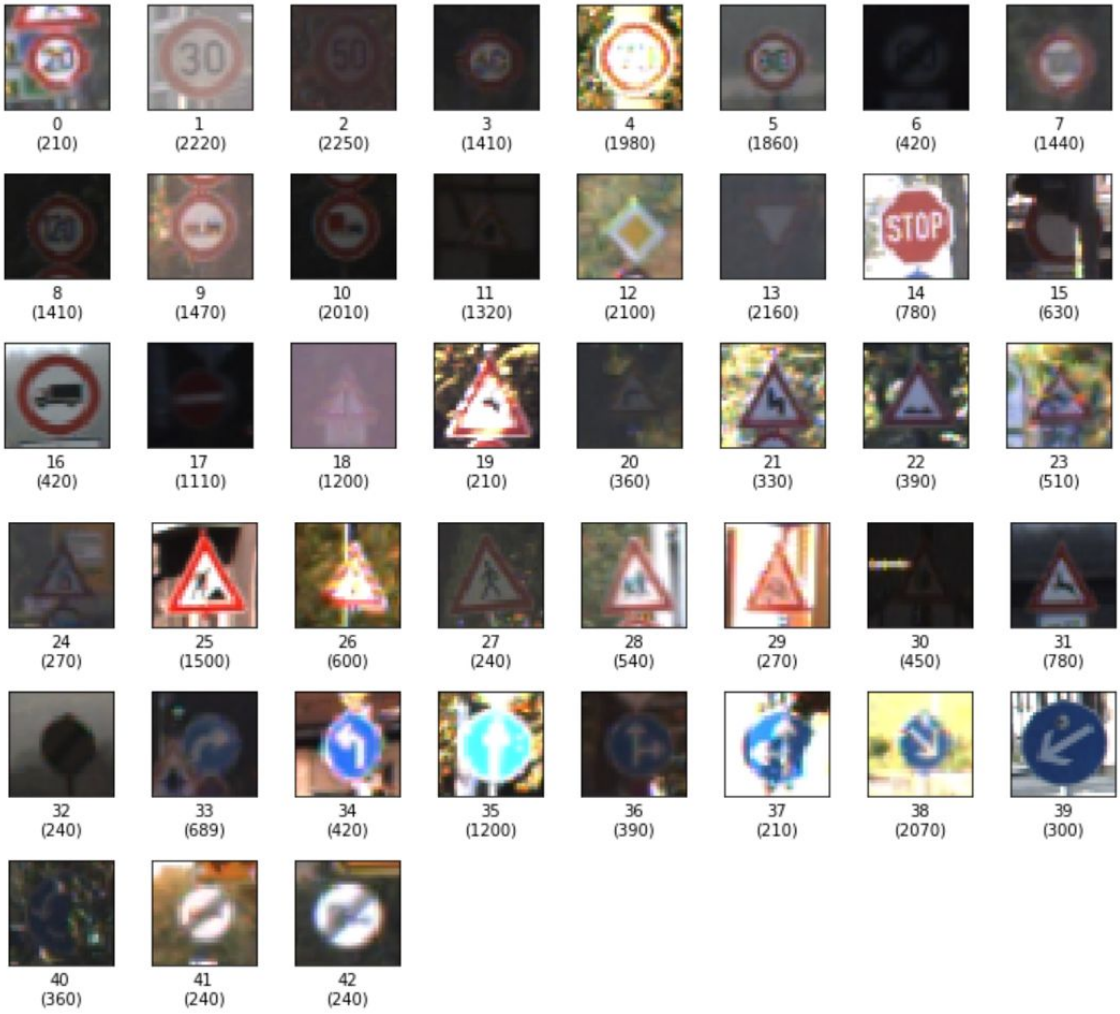
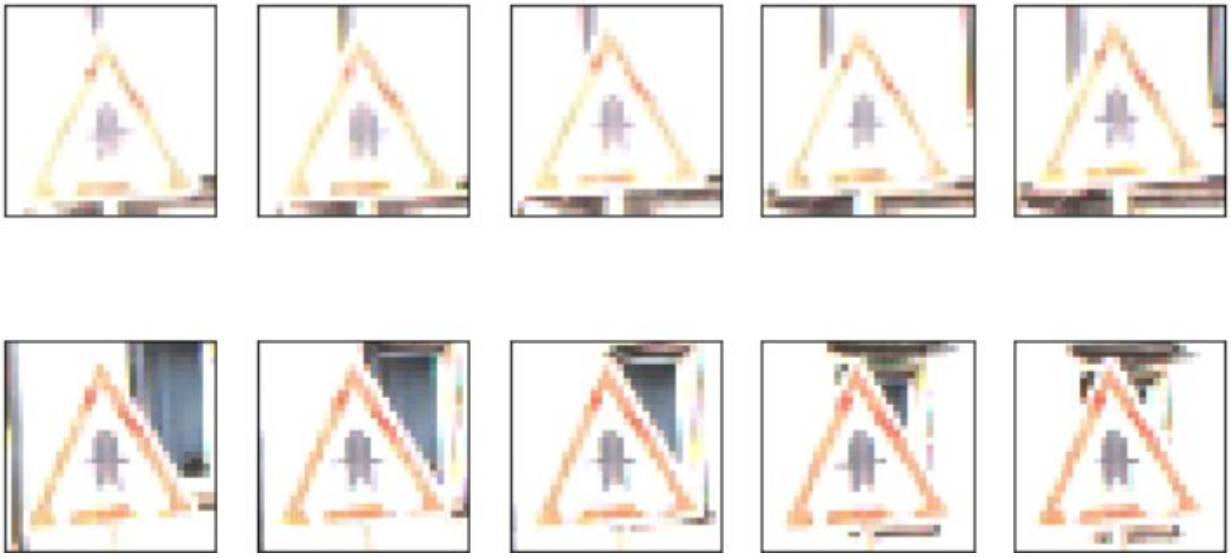


Fig shows the first image from each of the classes. Some of the images are very dark.

Exploratory Visualization

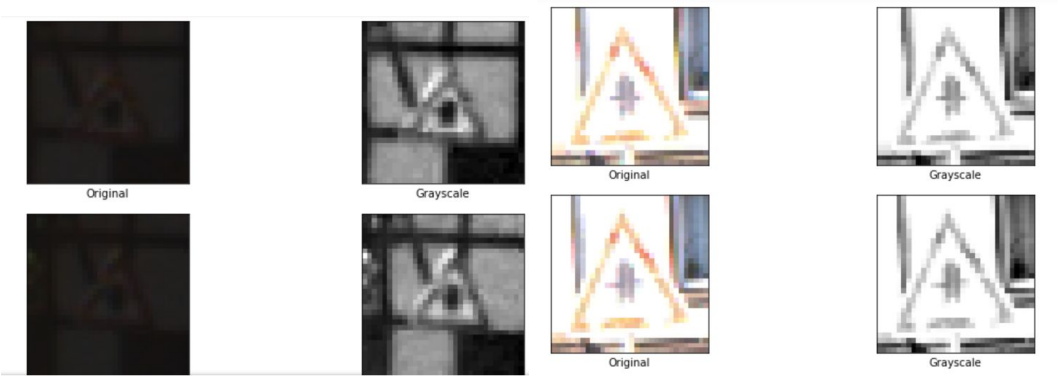
Fig shows a very dark image for class 11. Lets explore some images in class 11.

The followng are some really bright images form the same class



Converting to Grayscale.

Lets see what happens if we convert the dark ones and the bright ones to gray scale



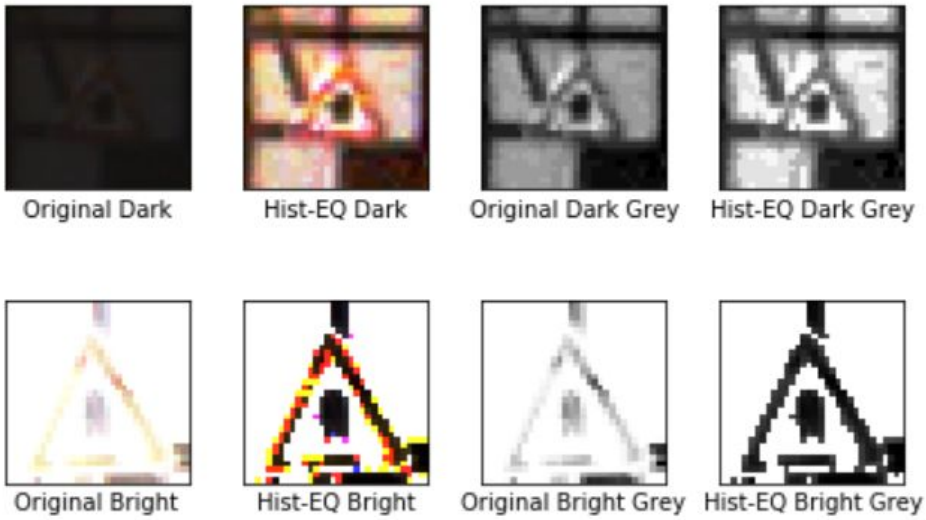
The dark ones seem to benefit from the grayscale conversion.

Lets explore some contrast enhancing techniques and apply them to the images to see how they transform.

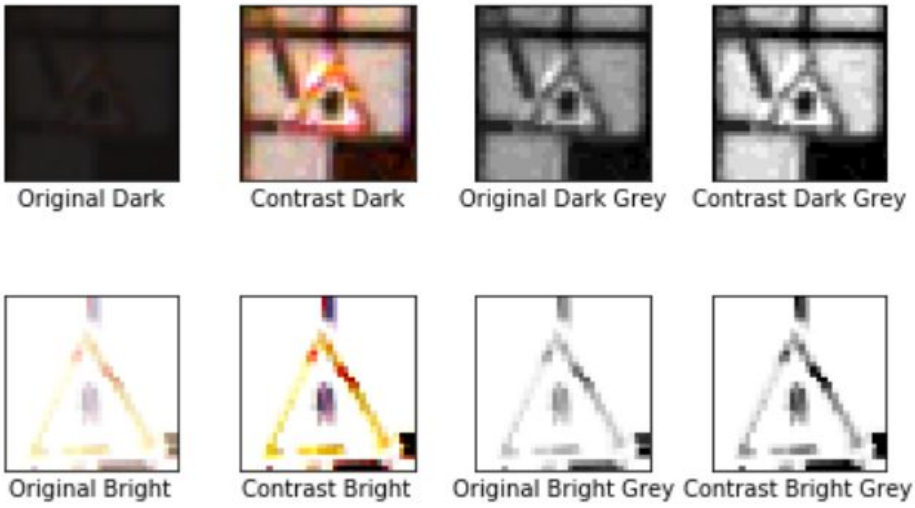
Image enhancing techqnieus

The notes on the following image enhancing technqiues have been taken from [wikipedia]

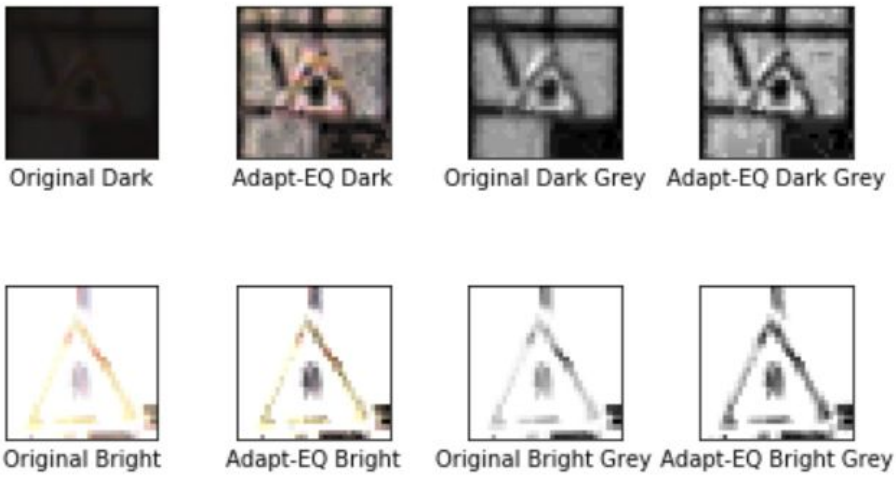
Histogram equalization is a method in image processing of contrast adjustment using the image's histogram. This method usually increases the global contrast of many images, especially when the usable data of the image is represented by close contrast values. Through this adjustment, the intensities can be better distributed on the histogram. This allows for areas of lower local contrast to gain a higher contrast. Histogram equalization accomplishes this by effectively spreading out the most frequent intensity values.



Contrast stretching (often called normalization) is a simple image enhancement technique that attempts to improve the**contrast** in an image by `**stretching**' the range of intensity values it contains to span a desired range of values, e.g. the the full range of pixel values that the image type concerned allows.

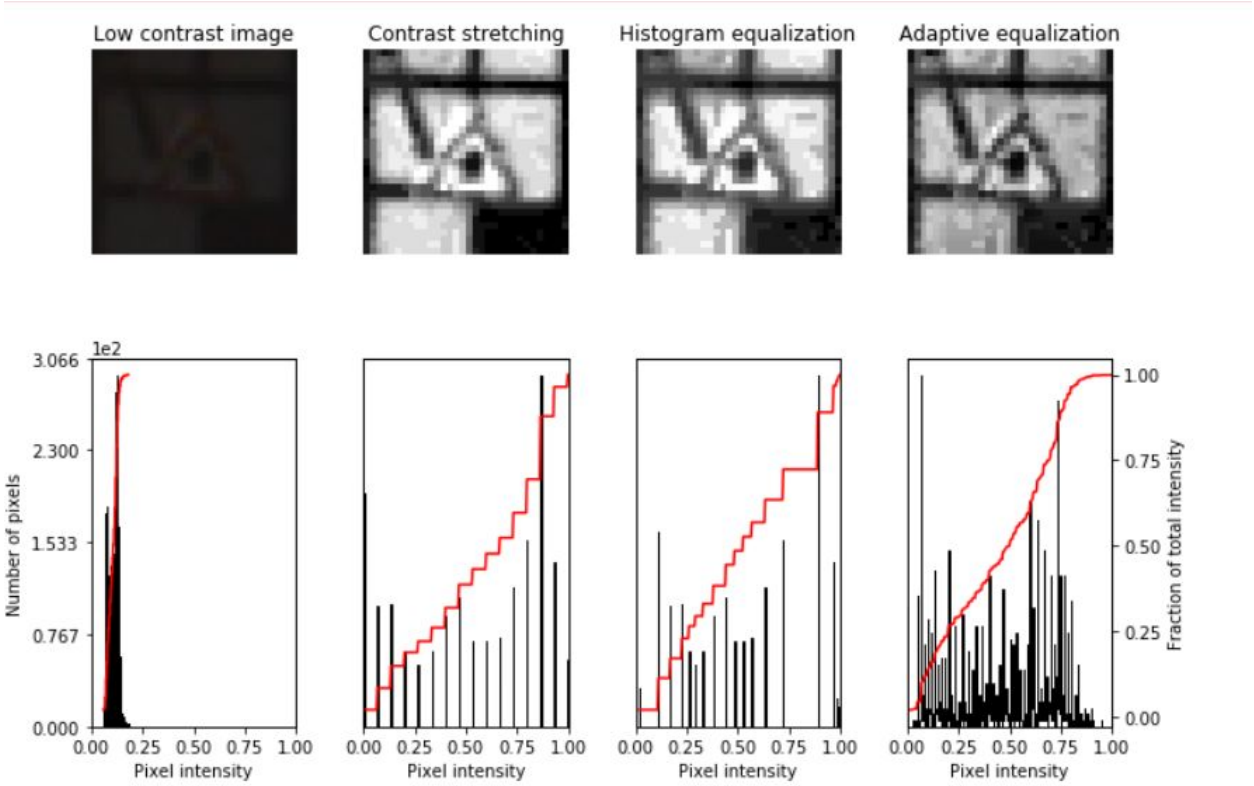


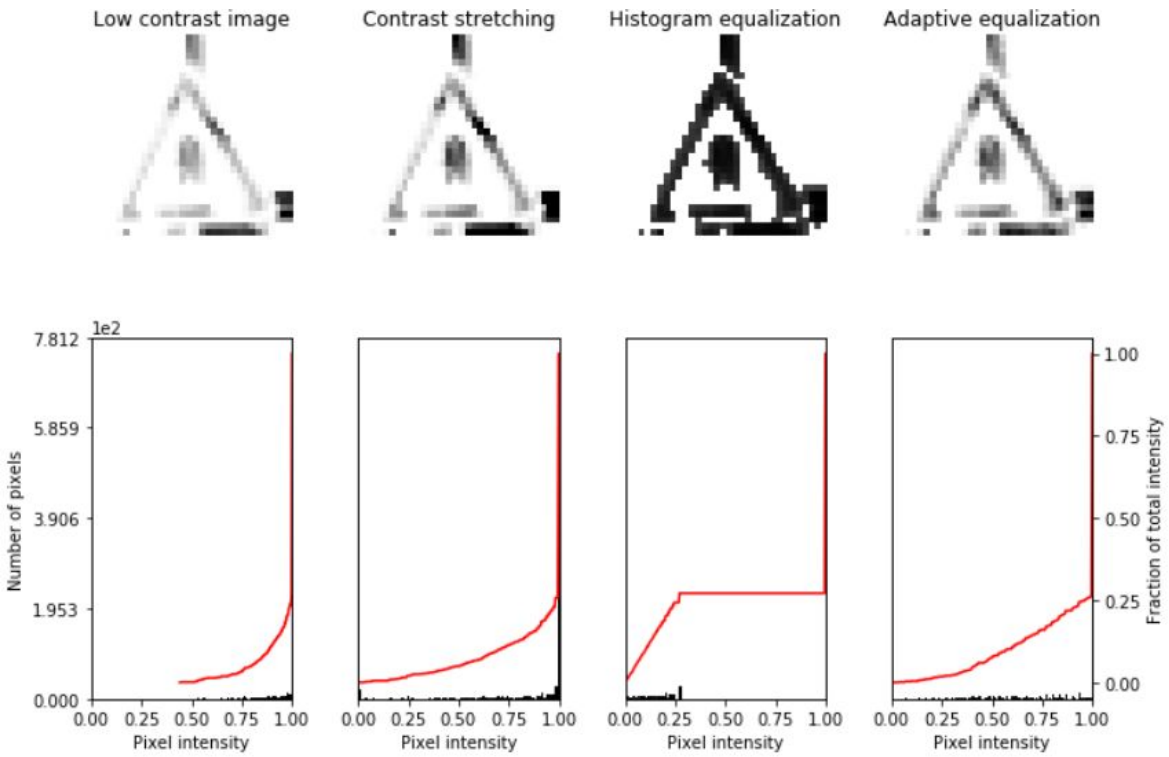
Adaptive histogram equalization (AHE) is a computer image processing technique used to improve contrast in images. It differs from ordinary histogram equalizationin the respect that the adaptive method computes several histograms, each corresponding to a distinct section of the image, and uses them to redistribute the lightness values of the image. It is therefore suitable for improving the local contrast and enhancing the definitions of edges in each region of an image.



Eyeballing the figures above, the Histogram equalization offers maximum clarity to the human eye. Adaptive Equalization sharpens the edges.

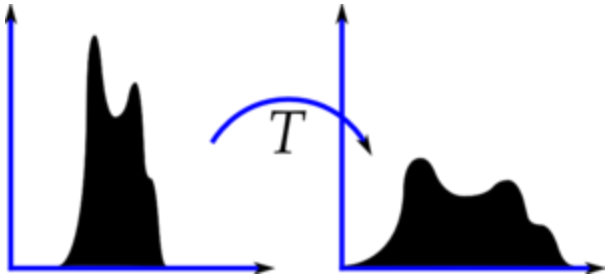
the pixel intensities are more balanced after the preprocessing of the dark and the bright images.





The pixel intensity histograms show a more balanced spread after applying the transformations.

Consider an image whose pixel values are confined to some specific range of values only. For eg, brighter image will have all pixels confined to high values. But a good image will have pixels from all regions of the image. So you need to stretch this histogram to either ends (as given in below image, from wikipedia) and that is what the above techniques do (in simple words). This normally improves the contrast of the image.[OpenCV]



The original dataset would benefit from preprocessing. A combination of these techniques are used to preprocess and augment data.

Algorithms and Techniques

Image Classification

Image classification is the task of taking an input image and outputting a class (say Stop sign or Children Crossing sign) or a probability of classes that best describes the image.

Technique

I use a Supervised Multi-Class Classification model that uses a special kind of neural network called Convolutional Neural Network. These networks have been some of the most influential innovations in the field of computer vision.

Methodology

1. Preprocess and Augment Data (Preprocess techniques have been discussed in the above Data Explore section)
2. Build Model with default hyperparameters
3. Train model on Train Data, Validate on Validate Data after each epoch. Study train-loss, validation-loss, train-accuracy, validation-accuracy and finetune the parameters.
4. Test all variations of models to see which gives the best Test Results
5. Compare the learning curves of the top two models, and then evaluate the training loss and accuracy graph of the top model.
6. Use the top rated model to visualize intermittent output as transformed by the Spatial Transformer Network and also the convolutions of the top rated model during Training.
7. Use the top rated model to evaluate test results, error analysis.
8. Visualize misclassification percentages, precision, accuracy/class, confusion matrices, plots, histograms.
9. Use new images found on Web to test with the top rated model and study its performance.

Algorithm

Repeat the following three steps until benchmark accuracy and loss are obtained. Finetune the parameters as the training happens and retrain. Fix the number of epochs; run the following steps until epoch reached.

1. Forward Pass (Convolutions, Feature identification) : The image passes through a series of convolution layers, each convolution builds on a feature map with the help of filters and identifies more complex features. In the beginning, the filter values are random and the process produces nonsensical feature maps. However with each iteration, when learning takes place and filters are corrected, the feature maps start identifying valid low level and high level features.

2. Loss Function E : At the end of an iteration, the final prediction is made by the fully connected layer and the loss is computed. The first pass generally produces the max error.

3. Backpropagation (Learning) : We go back into network from the nth layer and compute gradients of each layer with respect to the Loss function. All modules of a convolutional network are differentiable. Partial derivatives and chain rule may be applied to compute gradients with respect to given variable(loss). Gradient Descent can then be used to adjust weights.

At the end of n epochs, we should typically have a low loss, a probable accurate prediction.

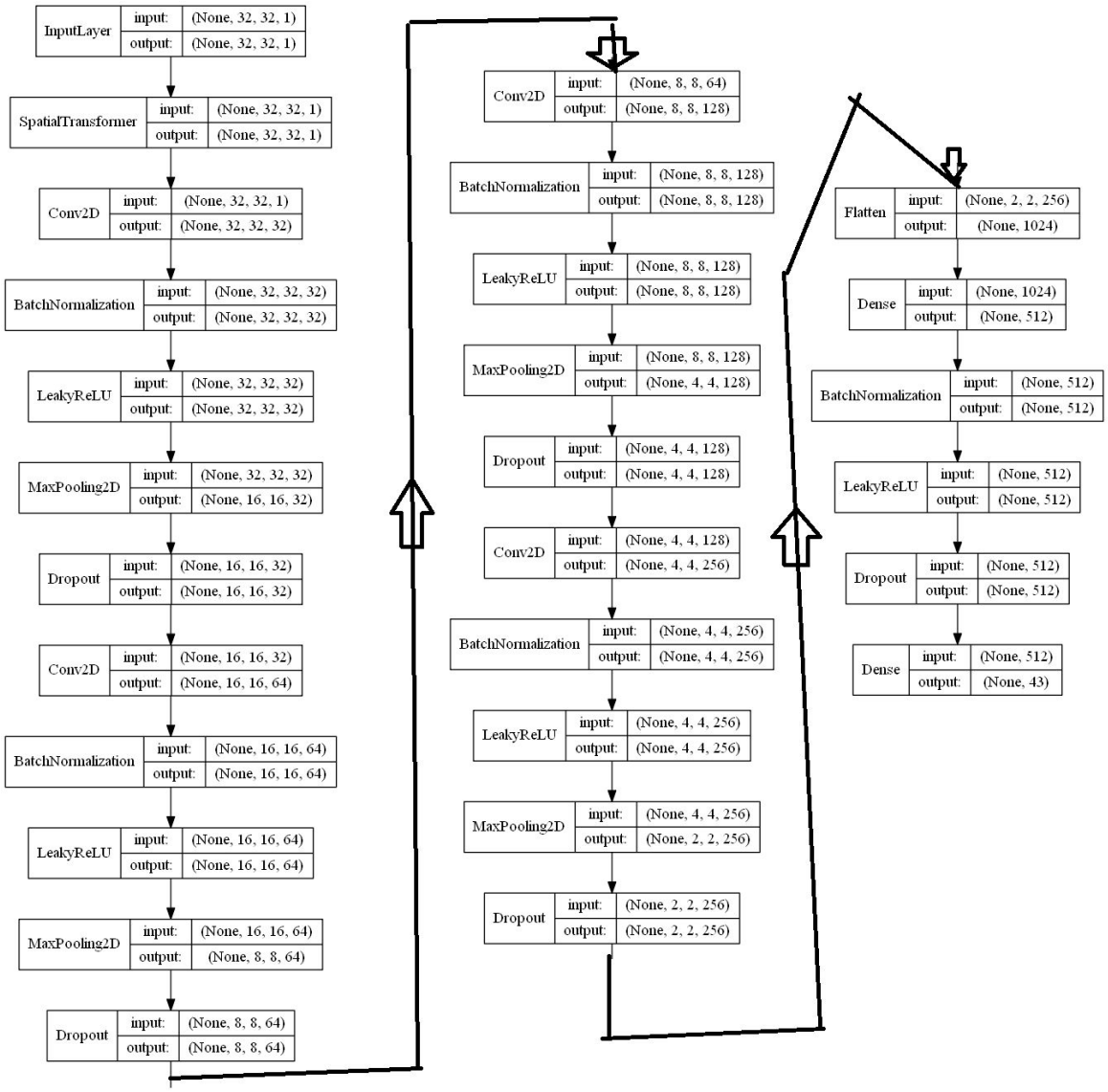
Model (Pipeline)

The model is inspired from the works of Yan Lecun's LENET [http://yann.lecun.com/exdb/lenet/], IDSIA Net [Multi-Column Deep Neural Network for Traffic Sign Classification Dan Cire¸san, Ueli Meier, Jonathan Masci and J¨urgen Schmidhuber] and Spatial Transformer Network[https://arxiv.org/abs/1506.02025]

The model has 4 convolution layers followed by a fully connected layer and a fully connected output layer. Batch normalization, Maxpooling, Dropout are used to provide faster convergence, generalization and regulate overfitting.

A learnable component called Spatial Transformer Network is injected at the very beginning so the network learns about spatial invariance. The inputs are rotated, scaled sheared and zoomed to provide for the invariance learning.

The following is a structure of the proposed model.



Building Blocks of the Model

1. Inputs

Input : An image in computer vision is represented as a matrix of digits representing its pixel values. A 32x32 colored image in RGB 3-channels is represented as a 3-d matrix of 32x32x3 numbers where each number is a value indicating the color intensity at that point ranging from 0-255.

Output: The image transformed by a layer. For instance, the STN may shear and zoom the image. The Convolutional layer performs matrix multiplication on the image and outputs feature maps.

The original image during preprocessing is transformed to grayscale. The input shape is 32x32x1

2. Spatial Transformer Network

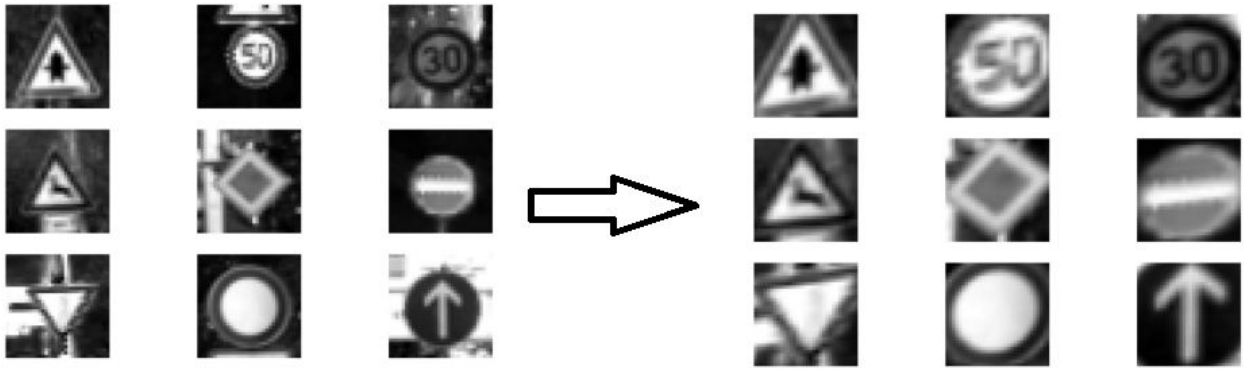
The first layer of the model is the Spatial Transformer Network. A transformer is a differentiable transformation module used to translate, rotate, shear and rescale an input image or an activation map.

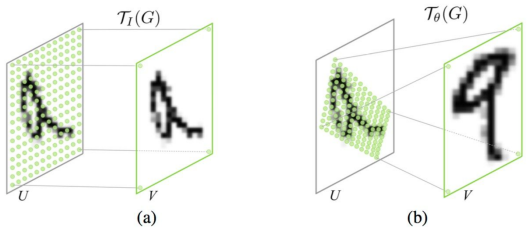
I use a transformer to provide invariance to deformations in the input images.

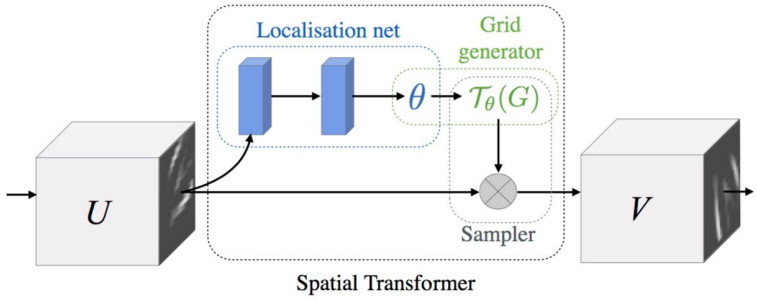
The notable quality of an STN is unlike static data augmentation, it is a differentiable module that learns to predict an accurate transformation for the deformed input. Because it is differentiable,standard backpropagation can be applied and the transformation coordinates (thetas) can be learnt. They can be injected anywhere in the model as they facilitate backprop.

(Input) -> (Spatially transformed by the STN)

The number 30 is zoomed in, the background of the signal is taken out. The circle with the number 30 on it is zoomed in. It also appears sheared.





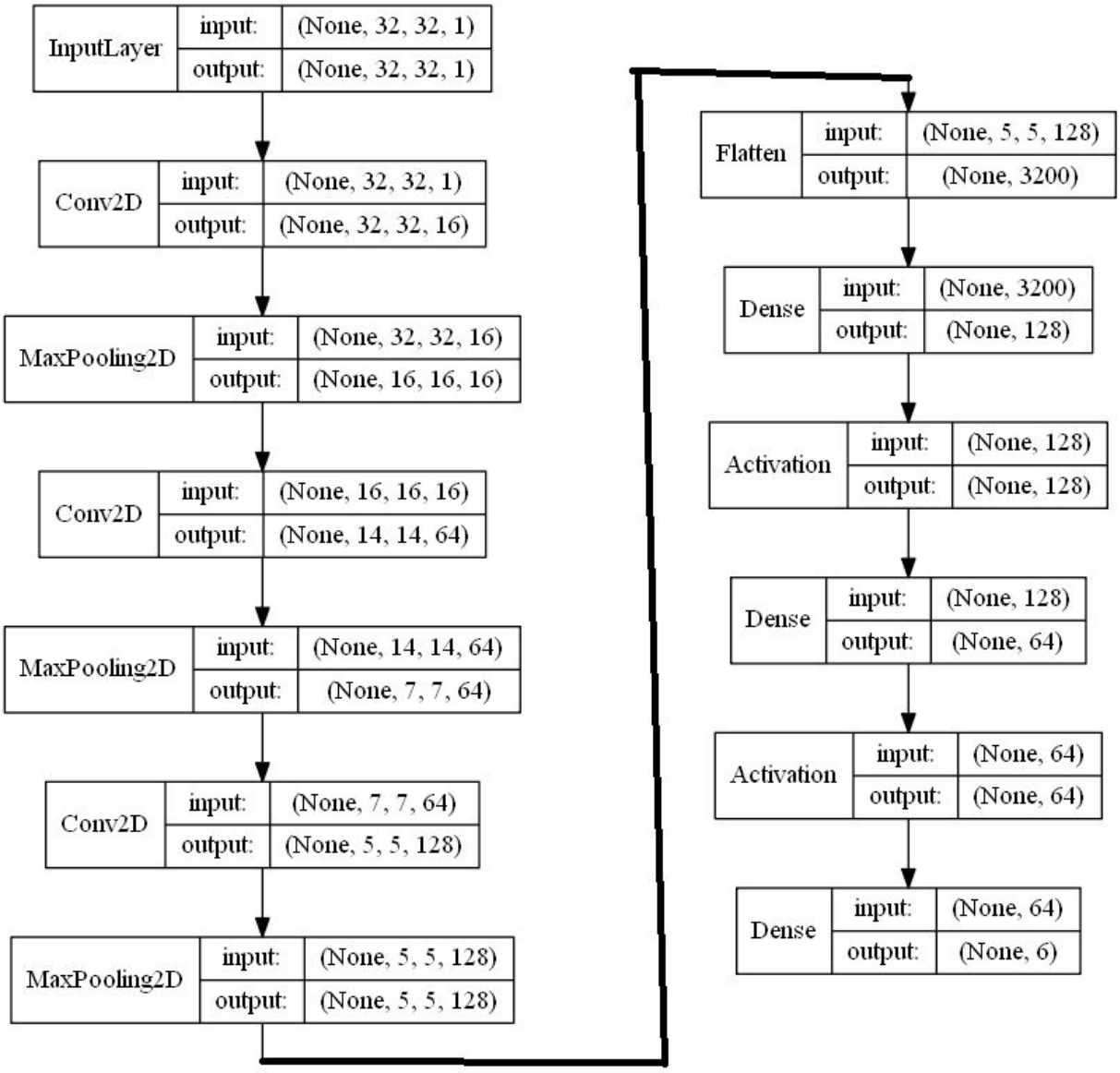


$$\begin{pmatrix} x_i^s \\ y_i^s \end{pmatrix} = \mathcal{T}_\theta(G_i) = \mathtt{A}_\theta \begin{pmatrix} x_i^t \\ y_i^t \\ 1 \end{pmatrix} = \begin{bmatrix} \theta_{11} & \theta_{12} & \theta_{13} \\ \theta_{21} & \theta_{22} & \theta_{23} \end{bmatrix} \begin{pmatrix} x_i^t \\ y_i^t \\ 1 \end{pmatrix}$$

The spatial transformer consists of a localization network, a grid generator and a sampler. The localization network predicts the 6 thetas required for the trasformation. The other parts render build and render the transformation on the input to produce the transformed ouput.

Architecture of the locnet

The locnet contains three convoluiton layers and one dense layer, followed by the output layer. This design was chosen by trial and error.

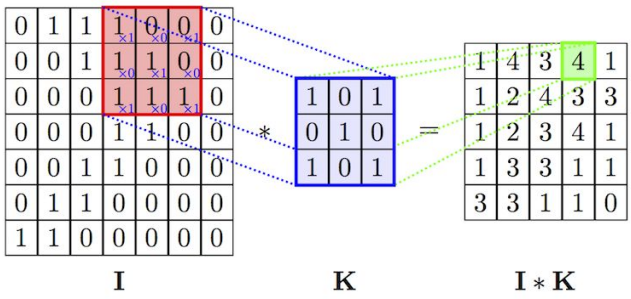


The final classifier in the locnet returns the predictions for the 6 thethas that determine the transformation

2. Convolution

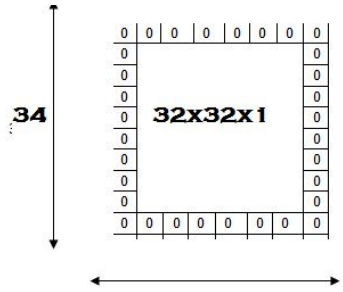
The Spatially transformed image has the same dimensions as the input image. The transformed image passes through a convolution layer. The convolution layer produces 32 feature maps as specified in the algorithm.

A convolution consists of a filter sliding over the input image looking for low level features. This is anonymous to a torch light moving over an image. As the **filter** is sliding, or **convolving**, around the input image, it is multiplying the values in the filter(filter weights) with the original pixel values of the image (aka computing **element wise multiplications**). These multiplications are all summed up. we repeat this process for every location on the input volume.This produces a smaller matrix called **activation map** or **feature map**. Each of these filters can be thought of as **feature identifiers**. Low level features are identified in the initial layers of a CNN like straight edges, simple colors, and curves. These feature maps become inputs to the next convolution layer which produces higher level feature maps that possibly identify circles, squares and other shapes.



K is the filter/kernel in the above image.

So as to have the same output height and width, I pad the input image with zeroes on the side.

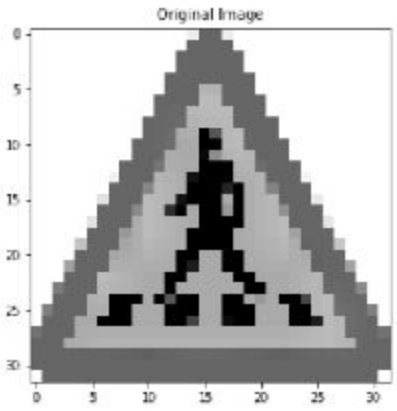


Size of the filter used in the algorithm : 3x3

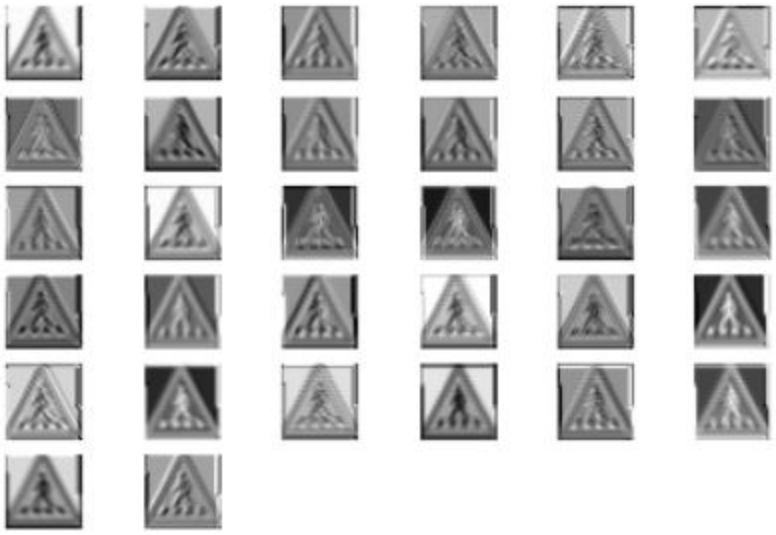
Strides : (The number of pixes the filter moves) 1

The depth of the filter should always match the depth of the input.

Output of the Convolutional Layer 1



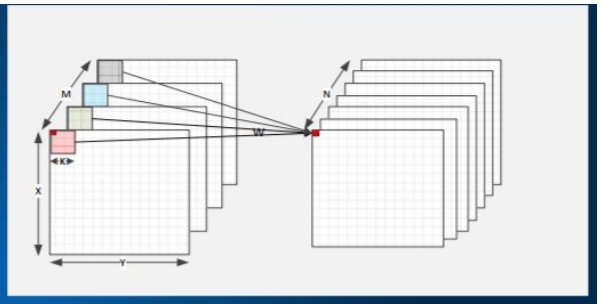
Transformations in Convolution Layer 0



As can be seen 32 filtermaps have been generated as specified in the output of the convolutional function. They identify the edges, curves and other low level features.

Chain rule is used during the learning of the weights of the filter that happens during the backpropagation phase. To start with, the weights are random. Partial derivatives using chain rule are computed for each weight with respect to the loss function E. The weights are then readjusted using Gradient Descent.

```
for (n = 0;  n < N;  n ++)
  for (m = 0;  m < M;  m ++)
    for(y = 0; y<Y; y++)
      for(x = 0; x<X; x++)
        for (p = 0; p< K; p++)
          for (q = 0;  q< K; q++)
            y_L (n; x, y) +=  y_L-1(m, x+p, y+q) * w (n ,m;  p, q);
```



Let' s use the chain rule for convolutional layer

$$\frac{\partial E}{\partial y_{l-1}} = \frac{\partial E}{\partial y_l} \times \frac{\partial y_l(w, y_{l-1})}{\partial y_{l-1}};$$

$$\frac{\partial E}{\partial w_l} = \frac{\partial E}{\partial y_l} \times \frac{\partial y_l(w, y_{l-1})}{\partial w_{l-1}}$$

[reference:link]

3. Batch Normalization

Despite normalization in the beginning, data is shifted and adjusted by the weights and parameters as it flows down the network.

Batch normalization avoids this problem of the internal covariate shift and improves convergence rates by dynamically normalizing the training data at the intermediate hidden layers, not just at the beginning of the pipeline.

The last line in the following algorihtm is what makes Batch Normalization really powerful - teh two learnable parameters thetha and gamma

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

[Fig: Reference]

We initialize the BatchNorm Parameters to transform the input to zero mean/unit variance distributions but during training they can learn that any other distribution might be better. The gamma and beta parameters can be learnt during backpropagation.

4. Activation

Without a *non-linear* activation function in the network, a Convolutional Neural Network, no matter how many layers it has, would behave just like a single-layer network, because summing these layers would give us just another linear function.

There are several kinds of non-linear activation functions such as Sigmoid, ReLU, Tanh. *I use LeakyRelu here.* **Leaky** ReLUs are one attempt to fix the “dying **ReLU**” problem. Instead of the function being zero when x < 0, a **leaky ReLU** will instead have a small negative slope (of 0.01 or so).

Leaky Reulu function is a differentiable function.

$$f(x) = \begin{cases} 0.01x & \text{for} \quad x < 0 \\ x & \text{for} \quad x \geq 0 \end{cases}$$

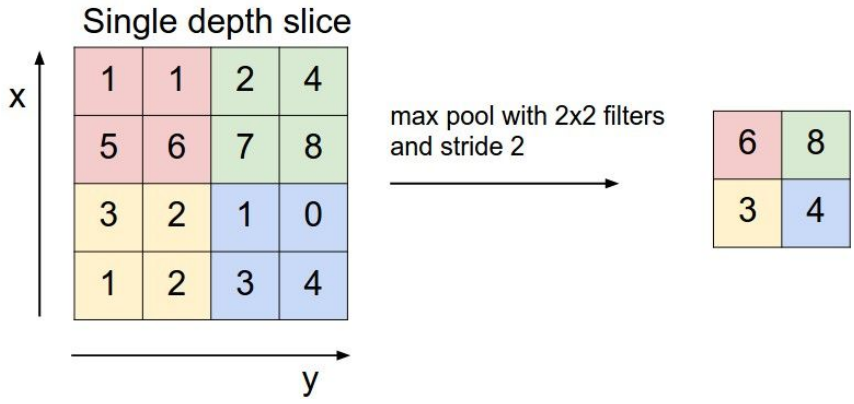
5. Pooling

Once a feature has been found, its exact location isn't as important as its rough location relative to other features. The function of the pooling layer is to progressively reduce the spatial size of

the representation to reduce the amount of parameters and computation in the network, and hence to also control overfitting.

Maxpooling is a popular function with image classifiers which I use in this model.

Pool size : 2x2



6. Dropout

Dropout is a regularization technique for reducing overfitting in **neural networks**. It adds noise to its hidden units thus preventing overfitting.

I use a dropout percentage of 0.2 in the convolution layers and 0.5 in the final fully connected layer.

7. Fully Connected Layers

Fully connected layers take the high-level filtered images and translate them into votes/probabilities. They classify the image.

The algorithm uses 2 fully connected layers. The final fully connected layer predicts the class of the input image. The output layer uses the Softmax activation.

Softmax activation is basically the normalized exponential probability of class observations represented as neuron activations. In a multi-class classification, 'Cross-Entropy' is cut out for usage along with Softmax.

8. Loss function

When the fully connected layer makes a prediction,the loss function is computed. It measures the compatibility between a prediction (e.g. the class scores in classification) and the ground truth label. The data loss takes the form of an average over the data losses for every individual example.

There are different kinds of loss functions such as mean_squared_error, hinge etc.

The one chosen for this algorithm is the "Categorical Cross-Entropy". This has been ubiquotous with Convolutional networks used f or image classification

I choose this loss function over others such as mean_squared_error because the ln() function in cross-entropy takes into account the closeness of a prediction and is a more granular way to compute error.

,

9. Backpropagation

The backpropagation equations provide us with a way of computing the gradient of the cost function. It uses chain rule in partial derivatives to compute how each layer and its weights have contributed to the loss.

The weights are readjusted to minimze the loss function.

Mathematically.,[reference]

1. **Input a set of training examples**
2. **For each training example** x**:** Set the corresponding input activation $a^{x,1}$, and perform the following steps:
    - **Feedforward:** For each $l=2,3,\ldots,L$ compute $z^{x,l}=w^l a^{x,l-1}+b^l$ and $a^{x,l}=\sigma(z^{x,l})$.
    - **Output error** $\delta^{x,L}$**:** Compute the vector $\delta^{x,L}=\nabla_a C_x \odot \sigma'(z^{x,L})$.
    - **Backpropagate the error:** For each $l=L-1,L-2,\ldots,2$ compute $\delta^{x,l}=((w^{l+1})^T\delta^{x,l+1})\odot\sigma'(z^{x,l})$.

3. **Gradient descent:** For each l=L,L−1,…,2l=L,L−1,…,2 update the weights according to the rule wl→wl−ηm∑xδx,l(ax,l−1)Twl→wl−ηm∑xδx,l(ax,l−1)T, and the biases according to the rule bl→bl−ηm∑xδx,lbl→bl−ηm∑xδx,l.

Training and Testing

The above model is run over the training data; after each epoch, the learned model is tested on validation data. This gives us a good indication of its performance. Any tuning, tweaking of parameters is done during this phase.

The final model is tested on unseen test data.

Benchmark

My proposed benchmark consists of the following :

The human performance on the German Traffic Signal Database is 98.84%.[http://benchmark.ini.rub.de/?section=gtsrb&subsection=results]. Given my limited resources, I would target 98% on accuracy.

Methodology

Data Preprocessing

After several trails, I decided to use the following preprocessing -

- Apply Grayscale conversion of all sets
- Preprocess Train data
    1. Histogram Equalization
    2. Supplement Train data with Histogram Equalization + Contrast Stretched (apply both transoformations) versions of the Grayscale.
    3. Further supplement Train data with Histogram Equalization + Adaptive Equalized (apply both transoformations) versions of the Grayscale. So the train data now has 3X more times than its original quantity.
- Preprocess Valid and Test data
    1. Histogram Equalization + Adaptive Equalization on the grayscaled version. Note, no supplementation. The original data is transformed.

- Data Augment Train data : The transformed data in Train set is further augmented by Data augmentation techniques that shear, rotate, zoom etc teh data. 200 images per class are added.

Model Training

I train on Train data, validate on each epoch on the validation data. This allows us to know how the model is performing. I then tuned the hyperparameters to obtain better performance.

3. Model Testing

The final model is tested on test data and loss/accuracy is measured.

Implementation

Platform : Keras on tensorflow, python, Amazon AWS

Parameters

Input image size : 32x32x1
Weights initialization : Random
Learing Rate:
Trial 1 : 0.001
Trial 2: Reduce on plateau (when validation loss doesn't change for 3 epochs, I reduce the learning rate by a factor of 0.5

Trial 3: Scheduled decay of learning rate (Every 5 epochs, I reduce the learning rate by a factor of 0.5

Optimizer : Adam Optimizer
Used default settings
$\alpha = 0.001$ (Learning Rate)

Loss : Categorical Cross-Entropy
Kernel size : 3x3

Strides : 1
Maxpool : yes, after each convolution
Pool size : 2x2
Dropout % : 0.2 in convolutional layers and 0.5 in the fully connected layer.
BatchNormalization : yes, after each maxpool

Model Summary

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
spatial_transformer_1 (Spati (None, 32, 32, 1)         501542
_________________________________________________________________
conv2d_16 (Conv2D)           (None, 32, 32, 32)        320
_________________________________________________________________
batch_normalization_1 (Batch (None, 32, 32, 32)        128
_________________________________________________________________
leaky_re_lu_1 (LeakyReLU)    (None, 32, 32, 32)        0
_________________________________________________________________
max_pooling2d_16 (MaxPooling (None, 16, 16, 32)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 16, 16, 32)        0
_________________________________________________________________
conv2d_17 (Conv2D)           (None, 16, 16, 64)        18496
_________________________________________________________________
batch_normalization_2 (Batch (None, 16, 16, 64)        256
_________________________________________________________________
leaky_re_lu_2 (LeakyReLU)    (None, 16, 16, 64)        0
_________________________________________________________________
max_pooling2d_17 (MaxPooling (None, 8, 8, 64)          0
_________________________________________________________________
dropout_2 (Dropout)          (None, 8, 8, 64)          0
_________________________________________________________________
conv2d_18 (Conv2D)           (None, 8, 8, 128)         73856
_________________________________________________________________
batch_normalization_3 (Batch (None, 8, 8, 128)         512
_________________________________________________________________
leaky_re_lu_3 (LeakyReLU)    (None, 8, 8, 128)         0
_________________________________________________________________
max_pooling2d_18 (MaxPooling (None, 4, 4, 128)         0
_________________________________________________________________
dropout_3 (Dropout)          (None, 4, 4, 128)         0
_________________________________________________________________
conv2d_19 (Conv2D)           (None, 4, 4, 256)         295168
```

```
_________________________________________________________________
batch_normalization_4 (Batch (None, 4, 4, 256)         1024
_________________________________________________________________
leaky_re_lu_4 (LeakyReLU)    (None, 4, 4, 256)         0
_________________________________________________________________
max_pooling2d_19 (MaxPooling (None, 2, 2, 256)         0
_________________________________________________________________
dropout_4 (Dropout)          (None, 2, 2, 256)         0
_________________________________________________________________
flatten_6 (Flatten)          (None, 1024)              0
_________________________________________________________________
dense_16 (Dense)             (None, 512)               524800
_________________________________________________________________
batch_normalization_5 (Batch (None, 512)               2048
_________________________________________________________________
leaky_re_lu_5 (LeakyReLU)    (None, 512)               0
_________________________________________________________________
dropout_5 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_17 (Dense)             (None, 43)                22059
=================================================================
Total params: 1,440,209
Trainable params: 1,438,225
Non-trainable params: 1,984
```

Trials

Several trials were attempted to better understand the data and the training parameters.

Initial trial was a CNN model with one less Convolutional layer and no batch normalization or spatial transformer. This was performed on the Grayscale dataset without data augmentation. It gave me an test accuracy of 92%

I added Batch Normalization and found the test accuracy barely increased.

I then added the Spatial Transformer Network, which increased the accuracy by one point to 93%.

I then performed data augmentation of 400 images per class consisting of rotation, shearing, zooming. This gave me a test accuracy of 95%.

I then preprocessed the data with histogram, adaptive equalization and contrast stretching. I tried out several combinations of preprocessing techniques. The best combination gave me an accuracy of about 96%.

I added an additional Convolutional layer amounting to the above described model and was able to secure an accuracy of 97%.

The final model with batch normalization, data augmentation, spatial transformer network, additional convolutional layer with static learning rate 0.001 gave me an accuracy of around 98.2%.

I then reduced the LR by a factor of 0.5 whenever thevalidation loss plateaued and it gave me better performance. The final accuracy is close to 98.49%.

I also tried reducing LR in a scheduled way every 5 eopochs by a factor of 0.5, which gave me an accuracy score of 98.44%

I chose the best model (reduce LR on a plateau of validation loss over 3 epochs by a factor of 0.5)

I can further experiment with the outputs of the convolution layers, the kernel sizes, dropout probabilities. However given the limited resources, I stop here.

Refinement

Spatial Transformer Network impoves accuracy.

Reducing the Learning Rate on plateau and also scheduled every n epochs, improves accuracy - 98.49% and 98.44% respectively.

Metrics

[Loss, Accuracy]

1. Model with Static LR : 0.001 : [0.078798514442264503, 0.97885985750106241]
2. Model with Reduce on Plateau LR : [0.055269072866941486, 0.98495645293713752]
3. Model with Scheduled LR reduce : [0.060815478878839174, 0.98440221699097752]

Other technical issues encountered

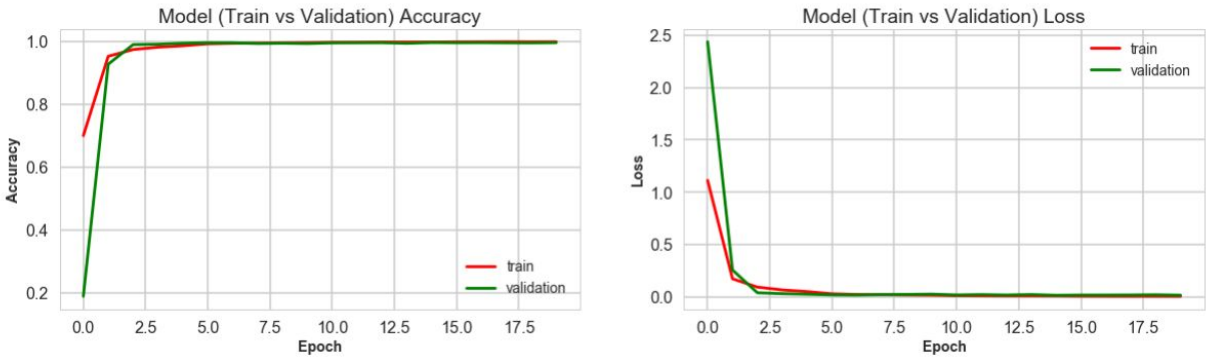
I encountered issues while processing Histogram equalization, Contrast Stretching and Adaptive Equalization on Grayscale (1 channel) pictures. I have been able to successfully resolve them.

Results

(approximately 2 - 3 pages)

Model Evaluation and Validation

Training Curve

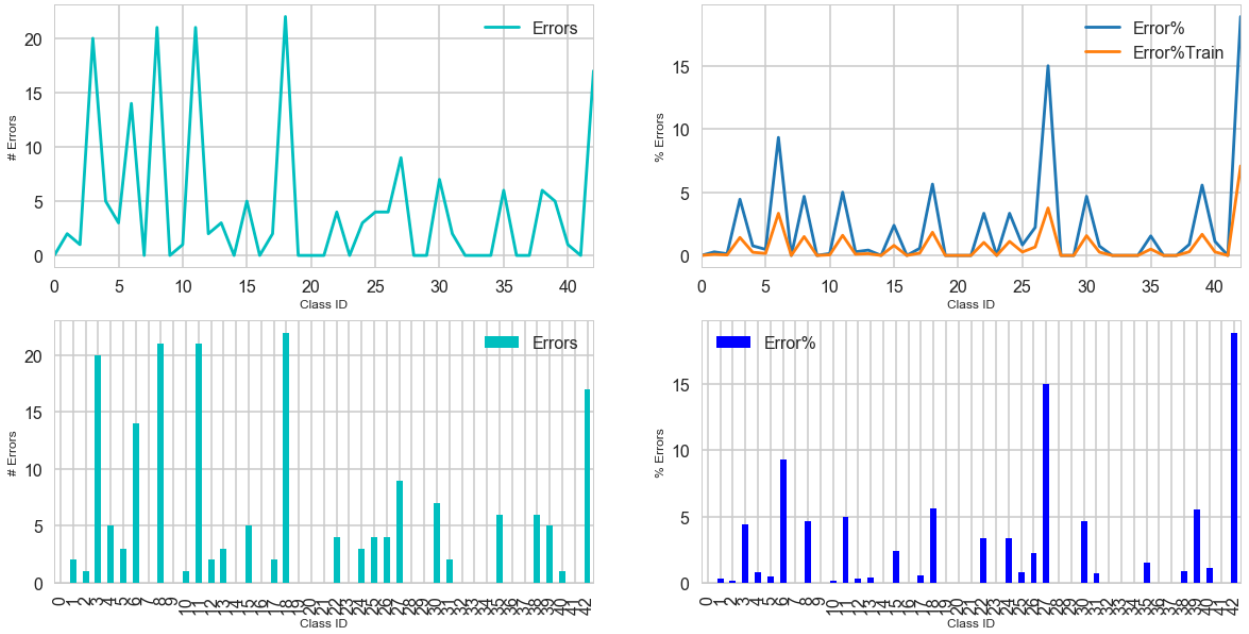


The training and validation curves show a steep rise in accuracy and steep fall in loss within the first 3 epochs. This could indicate the need for an initial lower learning rate than the current 0.001. As an experiment, I conducted the same test with 0.0001 starting learning rate, though the learning was not as steep, it did converge to a lower score by epoch 20. Given the limited resources, I had to stall the experiment after epoch 20. But possibly running the model for a greater number of epochs with the lower learning rate would enhance the accuracy/loss. The long plateau in learning shows a certain bias. More experimentation is needed.

Test Set evaluation

Total number of errors on the test set : 190

The following graphs show the spread of test errors over the classes

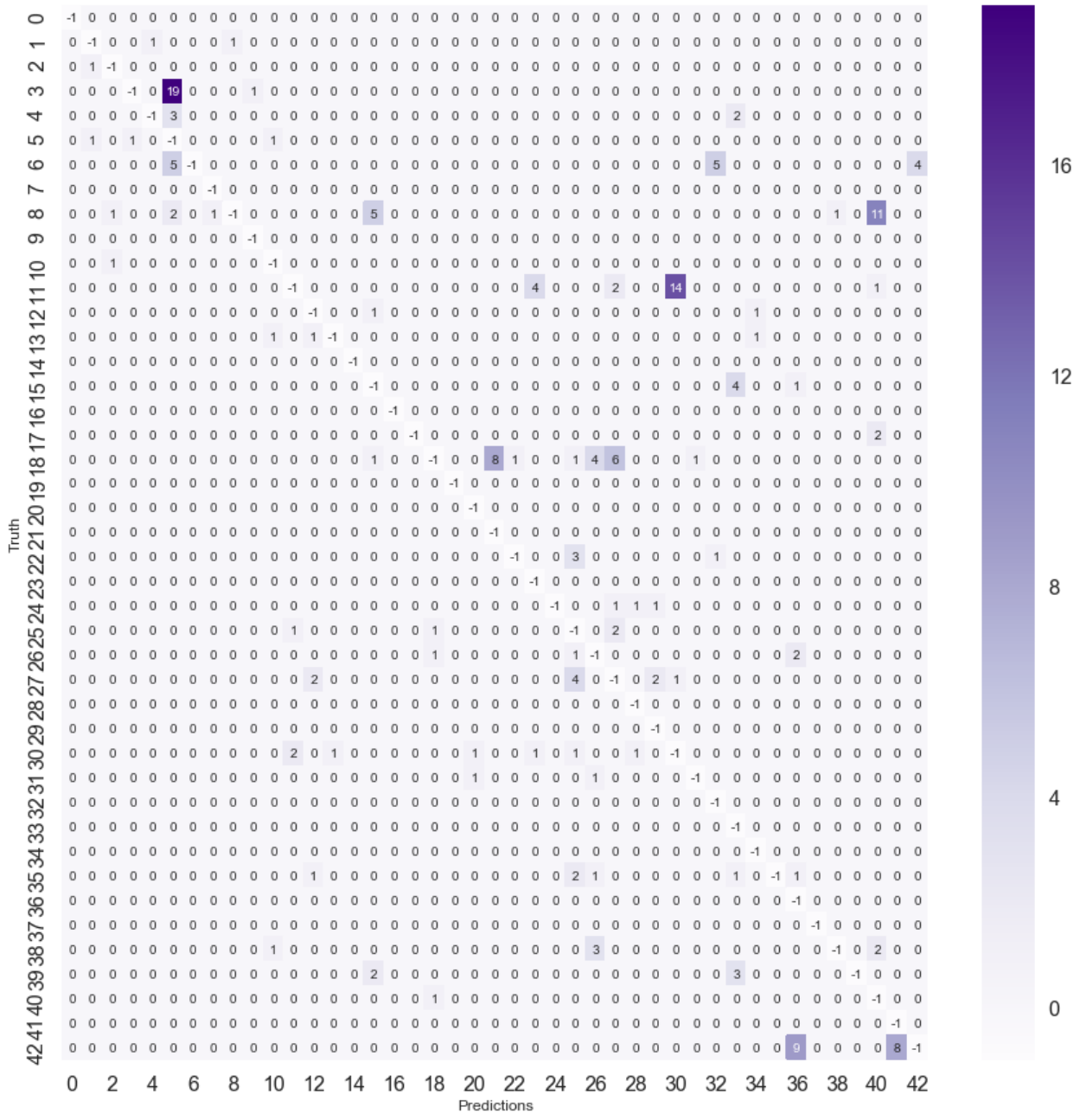


The above shows errors and error percentages on the test data.

Classes 6,27,42 have the maximum percentage of errors. I have listed out the maximum misclassified class for each of the error class in the following sections.

Confusion matrix of test errors

The following shows the Confusion matrix. Please note, given 98% accuracy, I have nulled the diagonal so we can see the actual misrepresentations.



3 has been maximum misrepresented as class 5.
Class 42 is highly misclassified as class 36 and class 41.

The following is a table of each class, its accuracy, precision and what class it has been maximum misrepresented as and what percentage of the misclassification happens to be the identified mispresentation class.

	Sign	Accuracy(F1)	Precision	Max Misclassified as	%Max_Misclassified_Class
0	Speed limit (20km/h)	1	1	-1	nan%
1	Speed limit (30km/h)	1	1	4	0.14%
2	Speed limit (50km/h)	1	1	1	0.13%
3	Speed limit (60km/h)	0.98	1	5	4.42%
4	Speed limit (70km/h)	1	1	5	0.46%
5	Speed limit (80km/h)	0.98	0.96	1	0.16%
6	End of speed limit (80km/h)	0.95	1	5	3.68%
7	Speed limit (100km/h)	1	1	-1	nan%
8	Speed limit (120km/h)	0.98	1	40	2.56%
9	No passing	1	1	-1	nan%
10	No passing for vechiles over 3.5 metric tons	1	1	2	0.15%
11	Right-of-way at the next intersection	0.97	0.99	30	3.51%
12	Priority road	1	0.99	15	0.15%
13	Yield	1	1	10	0.14%
14	Stop	1	1	-1	nan%
15	No vechiles	0.97	0.96	33	1.95%
16	Vechiles over 3.5 metric tons prohibited	1	1	-1	nan%

17	No entry	1	1	40	0.56%
18	General caution	0.97	0.99	21	2.17%
19	Dangerous curve to the left	1	1	-1	nan%
20	Dangerous curve to the right	0.99	0.98	-1	nan%
21	Double curve	0.96	0.92	-1	nan%
22	Bumpy road	0.98	0.99	25	2.59%
23	Slippery road	0.98	0.97	-1	nan%
24	Road narrows on the right	0.98	1	27	1.15%
25	Road work	0.98	0.98	27	0.42%
26	Traffic signals	0.96	0.95	36	1.14%
27	Pedestrians	0.84	0.82	25	7.84%
28	Children crossing	0.99	0.99	-1	nan%
29	Bicycles crossing	0.98	0.97	-1	nan%
30	Beware of ice/snow	0.93	0.91	11	1.4%
31	Wild animals crossing	0.99	1	20	0.37%
32	End of all speed and passing limits	0.95	0.91	-1	nan%
33	Turn right ahead	0.98	0.95	-1	nan%
34	Turn left ahead	0.99	0.98	-1	nan%

35	Ahead only	0.99	1	25	0.52%
36	Go straight or right	0.95	0.9	-1	nan%
37	Go straight or left	1	1	-1	nan%
38	Keep right	0.99	1	26	0.44%
39	Keep left	0.97	1	33	3.53%
40	Roundabout mandatory	0.91	0.85	18	1.12%
41	End of no passing	0.94	0.88	-1	nan%
42	End of no passing by vechiles over 3.5 metric tons	0.87	0.95	36	12.33%

In [72]:

```
1

2
```

From the above, we see that pedestrians has been misrepresented as Road Work. The following shows samples from original class and its wrongly predicted class.

Pedestrians :

<matplotlib.figure.Figure at 0x1e11156eb70>

was misclassified as:
Road work :

Right-of-way at the next intersection :

was misclassified as:
Beware of ice/snow :

End of no passing by vechiles over 3.5 metric tons :

was misclassified as:
End of no passing :

Benchmark

The model meets the benchmark specified. The goal was to meet 98% accuracy and it does so.

Test on Web Data

The following are some images gathered from the web. I preprocessed the images and tested them with the model.


Original


Processed


Original


Processed


Original


Processed


Original


Processed

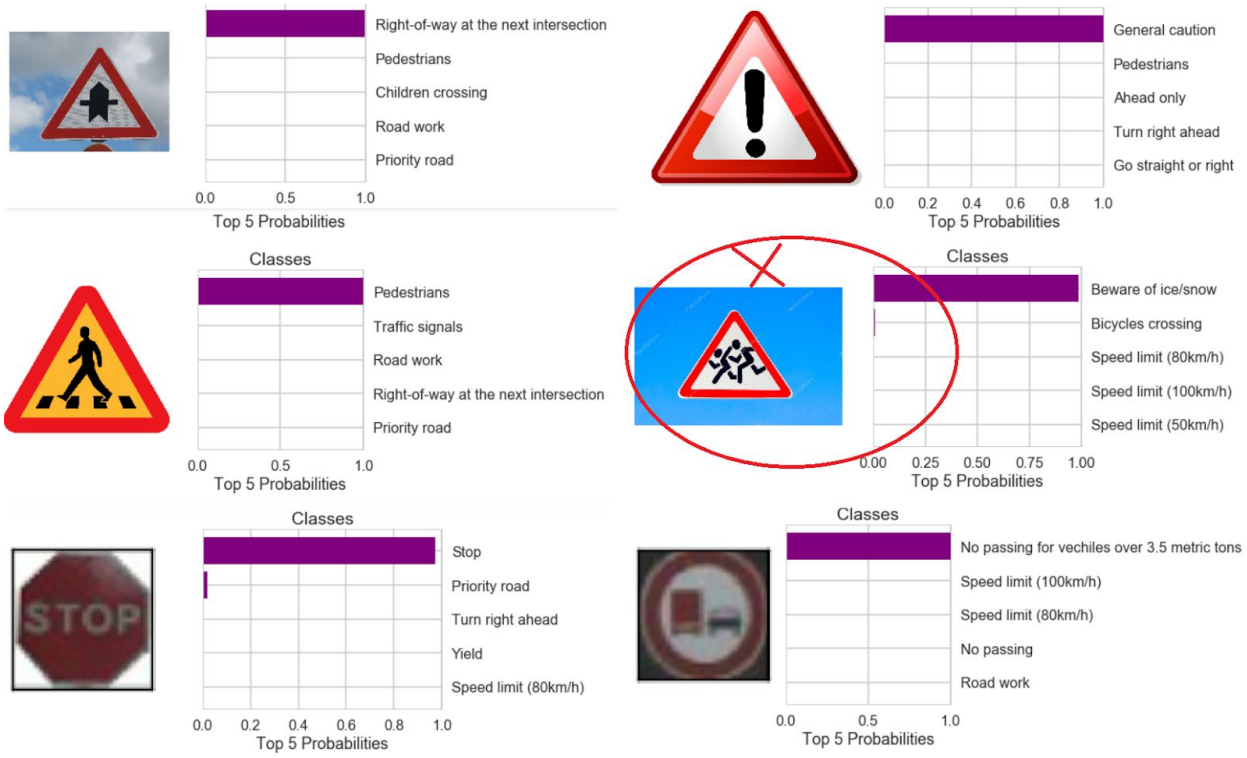
Original

Processed


Original


Processed

Predictions



The model gets only one image wrong. The children cross sign that the model wrongly predicted has a different pattern in the dataset.

Conclusion

Traffic Signal Classification is an important and a complicated task that has gained importance due to advances in self-driving car research.

Convolutional networks make use of Computer Vision, machine learning (Deep learning), Mathematical optimization (Gradient Descent) to classify images. The training part consists of three stages-The forward pass, where features are detected, the LOSS function computation, adn the backpropagation or the backward pass where gradients of each layer are computed with respect to the overall loss, and the weights and other learning parameters are adjusted to facilitate better feature identification in the next forward pass.

Figuring out the correct parameters of a machine learning algorithm for a specific dataseet requires tuning of its parameters and hyperparameters. Its performance depends on its intrinsic and tuned bias and variance.

This project makes use of the Keras Platform to train a model, tune a model. It makes use of several state-of-art research elements such as Batch Normalization, Spatial Transformer Network, Leaky Relu to attain a relatively higher accuracy score.

Several factors drive the metrics of performance. More experimentation can be performed as follows

1. The locnet may be downsized in the number of its convolutions
2. Outputs of the main model can be changed in each layer
3. Additional Convolutions may be added to the model to see if its intrinsic bias may be corrected.
4. Maxpool follows all convolutions. Maxpool adds noise, and donwsamples the intermittent feature maps. It may be removed from some of the convolutions or fully connected layers.
5. Dropouts diminish the processing power of layers. Some dropouts may be dropped or their percentages may be tweaked.
6. Lower learning rate and a higher epoch max may be used to see if a gradual but more fine-tuned learning curve occurs leading to a higher accuracy and lower loss.

References

Most of the references are inline.