Traffic Sign Recognition using CNN with Learned Color and Spatial Transformation

Capstone Project for Machine Learning Engineer Nanodegree

Liangyuan Tian
March 24th, 2017

I. Definition

This project has provided a novel approach for traffic sign imgage recognition utilizing the newest advancements in computer image recognition. A deep learning model which is capable of learning color and spatial variations of image objects is proposed. As a result, the model has higher performance than human without data preprocessing (except cropping) nor augmentation.

Project Overview

Recent advancement in autonomous vehicles has shown the once fictional technology is gradually becoming a reality in the near future. In the foreseeable future, vehicles of various automation levels will coexist on the road. Thus it is crucial for automated driving machines able to recognise and interpret the meaning of traffic signs and follow traffic rules.

The GTSRB [4] is a multi-class, single-image dataset consists of 50,000+ traffic sign images with 43 classes collected in Germany. Some of the images samples are shown in Figure 1.

Figure 1: GTSRB sample image

This project use the images in the dataset as input, and the class category as output. Data can be downloaded from the official website http://benchmark.ini.rub.de/?section=gtsrb&subsection=dataset. These large amount of images are collect in real life, thus very suitable to be used as training data for our learning model. To save time on cropping images from the original dataset, this project uses the already processed GTSRB data from Udacity, which can be downloaded from here.

Traffic sign recognition topics have been relatively well studied in research, notable publications include [1], which used nerual network to classify signs; and [2] uses pattern matching; with the advances in convolutional nerual network; [3] is able to achieve human level accuracy using muti-scale convolutional nerual network. The winner of the original competition [5] used a committee of convolutional neural networks achieved the best accuracy of 99.46%.

Despite the success in convolution nerual networks (CNN), conventional CNNs are unable to handle color and spatial variation well. Image preprocessing (Histogram equlization, convert to gray scale) and augmentation (rotation, translation, sheering, augment brightness) are common practices to give the network more synthetic variations to learn from. This complicates training process and requires addtional processing before feeding data into the detection model.

Problem Statement

For this multi-class classification problem, an automatic traffic sign recognition model will be built to categorize various traffic sign images. A deep convolutional neural network will be trained on various traffic sign images, its performance should exceed human accuracy to provide reliable decision support for autonomous vehicles.

In order to enable the network to learn color and spatial variance, 2 learnable modules will be introduced.

Spatial transformer network

Spatial transformer network [12] (STN) is a learnable module which explicitly allows the spatial manipulation of data within the network. This differentiable module can be inserted into existing convolutional architectures, giving neural networks the ability to actively spatially transform feature maps, conditional on the feature map itself, without any extra training supervision or modification to the optimisation process. An example of how STN transforms an image is given in Figure 2. On the left is the input image. In the middle illustrates where STN is sampling. On the right is the output of the transformation.

Figure 2: STN example

Learned color transformation

[13] has systematically compared many techniques of the recent advancements in image recognition. One of them is learned color transformation using 1 by 1 convolution modules. This project adopts the best practice using RGB -> 10 1 by 1 convolution -> 3 1 by 1 convolution architecture with VLReLu activation.

Aside from these 2 learnable plugin modules, other techniques such as improved feature extraction architecture, convolutional classfier architecture and batch normalizaion techniques are implemented into the module. All techniques will discussed in detail in the following chapters.

Metrics

Many prior researches have been conducted on this dataset. The best published results are listed in the table below.

Method	Accuracy
Committee of CNNs[5]	99.46%
Color-blob-based COSFIRE filters for object recogn6	98.97%
Human Performance[7]	98.84%
Multi-Scale CNNs[3]	98.31%
Random Forests[8]	96.14%
LDA on HOG 2[9]	95.68%
LDA on HOG 1[10]	93.18%
LDA on HOG 3[11]	92.34%

Our model will be benchmared against these published results on percentage accuracy.

II. Analysis

Data Exploration

In this project, we have in total 34799 training images, 4410 validation images and 12630 testing images with 43 labels in every dataset. The label distribution of the training set is illustrated below. As its shown in Figure 3, label distribution is unbalanced, some classes have significantly larger quantity than others. We will not attempt to balance the dataset as the intention is to test the model performance without augmentation.

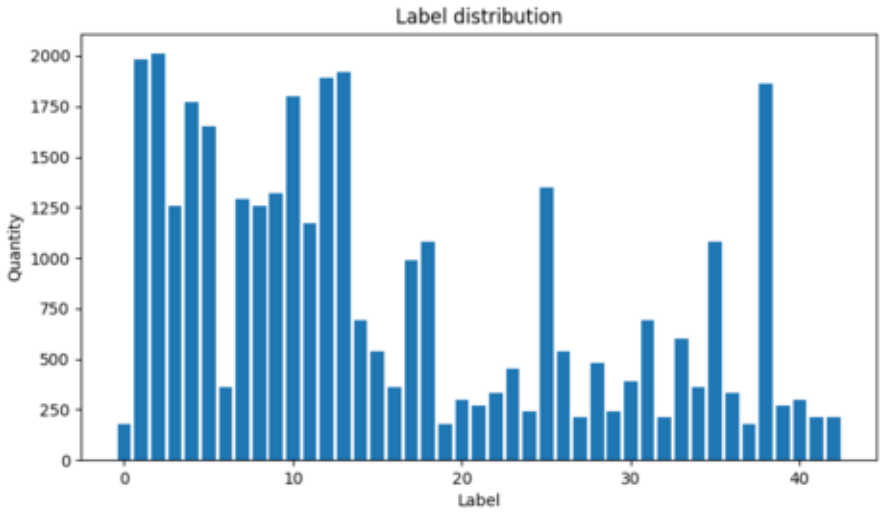

Figure 3: Training label distribution

As for images features, Figure 4 shows some of the charactoristics of the dataset. As we can see, images haven taken under many lighting conditions, from different viewing angle, with different resolutions. Addtionally, there might be artifacts such as shadow, flare and motion blur.

Algorithms and Techniques

Spatial transformer network

The goal of spatial transformers [12] is to add to the base network a layer able to perform an explicit geometric transformation on an input. The parameters of the transformation are learnt thanks to the standard backpropagation algorithm, meaning there is no need for extra data or supervision. The structure of STN is shown in Figure 5.

The input feature map U is passed to a localisation network which regresses the transformation parameters θ. The regular spatial grid G over V is transformed to the sampling grid $T_\theta(G)$, which is applied to U using differentiable image resampling, producing the warped output feature map V . The combination of the localisation network and sampling mechanism defines a spatial transformer.

The 3 elements of STN is decribed in detail in the following passages.

Localisation Network

Figure 4: Features

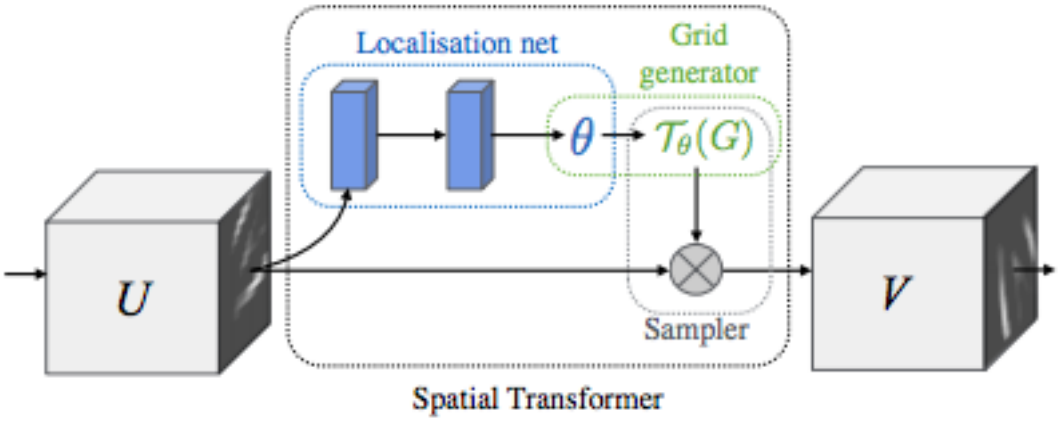

Figure 5: Spatial Transformer Network structure

The localization network takes the original image as an input and outputs the parameters of the transformation we want to apply. It takes image as an input, and output affine trasformation matrix with 6 dimensions. The localization network structure for this project is shown in Figure 6.

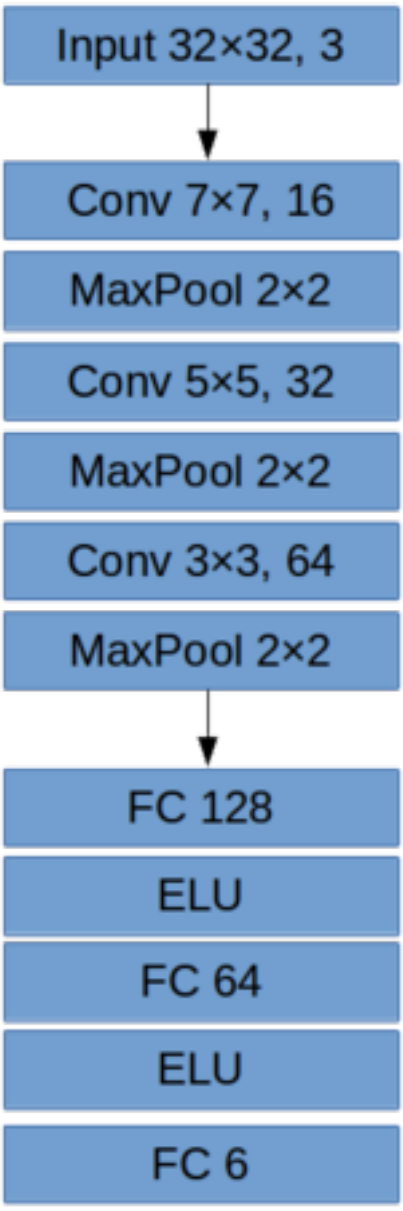

Figure 6: Localization network structure

The output of the last fully connect is then reshaped into affine transformation matrix:

$$A_\theta = \begin{bmatrix} \theta_{11} & \theta_{12} & \theta_{13} \\ \theta_{21} & \theta_{22} & \theta_{23} \end{bmatrix}$$

Parameterised Sampling Grid

The grid generator generates a grid of coordinates in the input image corresponding to each pixel from the

output image.

$$\begin{pmatrix} x_i^s \\ y_i^s \end{pmatrix} = T_\theta(G_i) = A_\theta \begin{pmatrix} x_i^t \\ y_i^t \\ 1 \end{pmatrix} = \begin{bmatrix} \theta_{11} & \theta_{12} & \theta_{13} \\ \theta_{21} & \theta_{22} & \theta_{23} \end{bmatrix} \begin{pmatrix} x_i^t \\ y_i^t \\ 1 \end{pmatrix}$$

where (x_i^t, y_i^t) are the target coordinates of the regular grid in the output feature map, (x_i^s, y_i^s) are the source coordinates in the input feature map that define the sample points.

An example of affine transformation is given in Figure 7.

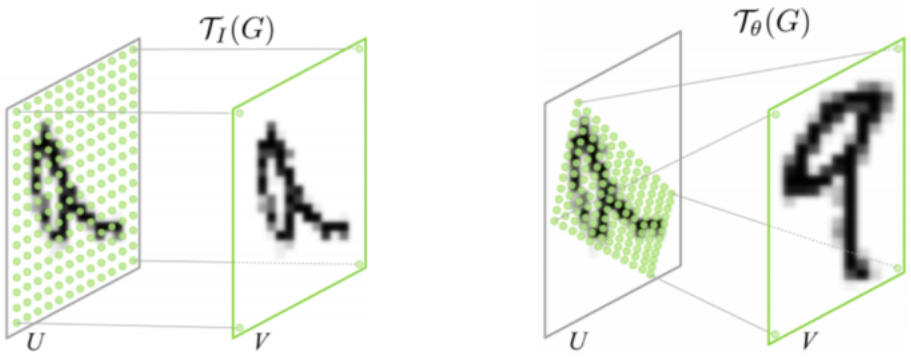

Figure 7: Affine Transformation

Differentiable Image Sampling

The sampler generates the output image using the grid given by the grid generator. In this project we use bilinear interpolation, the output value for pixel i at location (x_i^t, y_i^t) in channel c is

$$V_i^c = \sum_n^H \sum_m^W U_{nm}^c max(0, 1 - |x_i^s - m|) max(0, 1 - |y_i^s - n|)$$

An example of bilinear interpolation is shown in Figure 8:

In this geometric visualisation, the value at the black spot is the sum of the value at each coloured spot multiplied by the area of the rectangle of the same colour, divided by the total area of all four rectangles.

Since bilinear sampling kernel is Differentiable, errors can be back propagated to the weights in localisation network.

Learned color transformation

[13] has compared various learned image pixel trasformations via 1 by 1 convolution. The results from the original paper is listed in the following table.

Architecture	Non-linearity	Accuracy change
RGB -> conv1x1x10->conv1x1x3	tanh	-0.008%
RGB	-	0%
RGB -> conv1x1x3->conv1x1x3	VLReLU	+0.009%
RGB -> conv1x1x10-> conv1x1x3 + RGB	VLReLU	+0.011%
[RGB; log(RGB)] -> conv1x1x10-> conv1x1x3	VLReLU	+0.011%
RGB -> conv1x1x16->conv1x1x3	VLReLU	+0.012%
RGB -> conv1x1x10->conv1x1x3	VLReLU	+0.014%

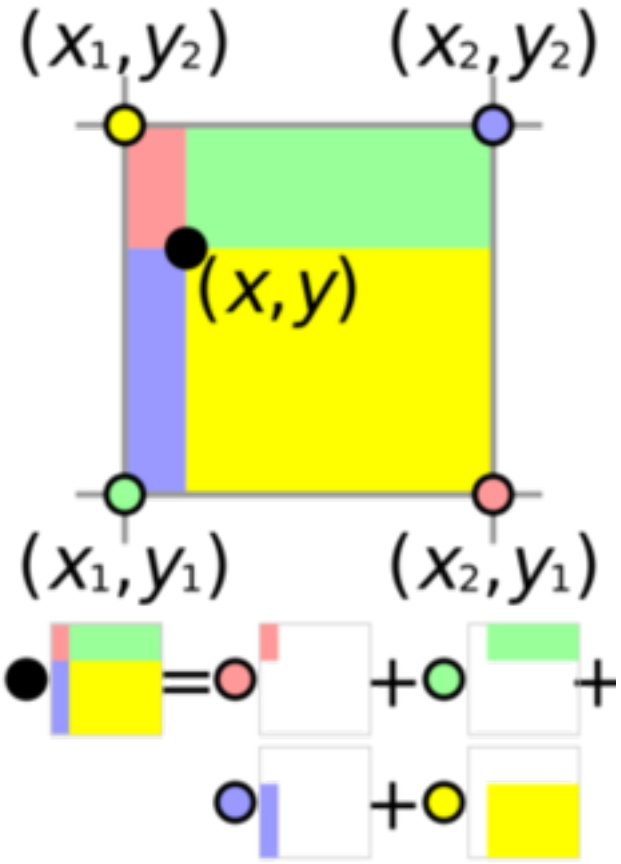

Figure 8: Bilinear interpolation

The best performing color transformer RGB -> conv1x1x10->conv1x1x3 is selected as the learning module.

CNN network design

[17] has proposed an empirical CNN architecture design for image recognition, this project adopts the authors network design for 32x32 image classification.

Feature extraction network

Layer type	Kernel size	Feature map size	No. of filters
convolution	5x5	32×32	16
convolution	5x5	32x32	32
convolution	5x5	32x32	64
convolution	5x5	32x32	96
convolution	5x5	32x32	128
convolution	5x5	32x32	192
MaxPool	2x2	-	-
convolution	5x5	16x16	256
MaxPool	2x2	-	-

Classifier network

Layer type	Kernel size	Feature map size	No. of filters
convolution	5x5	8x8	128
convolution	5x5	8x8	64
MaxPool	8x8	-	-

Batch normalization

Batch Normalization [18] allows for faster learning and higher overall accuracy by normalizing layer inputs. This project implement BN for each convolution layer to improve model performance.

Benchmark

The the performance of our model will be tested against the human performance [7] on both overall accuracy as well as categorical accuracy.

Human performance accuracy

Overall	Blue	Danger	End of	Red round	Red other	Speed	Special
98.84%	99.72%	98.67%	98.89%	98.00%	99.93%	97.63%	100%

III. Methodology

Data Preprocessing

Since the model structure is able to learn the color and spatial variance, new test images are no longer needed to be precisely cropped nor augmented. Thus training process is greatly simplified, we will then only focus on adjusting hyper parameters in the network to get the best performance.

Implementation

Our project is composed of 3 main components from coding perspective: - Model definition in conv_model.py, defines model architecture. - Spatial transformer network in spatial_transformer.py, defines spatial transformer network class. - Training in train_keras.py, defines data handling and training process.

Each component is disscussed in detail in the following sections.

Model definition

Our model is implement in Keras [21] with Tensorflow[22] backend. The final CNN architecture is illustrated in the Figure 10.

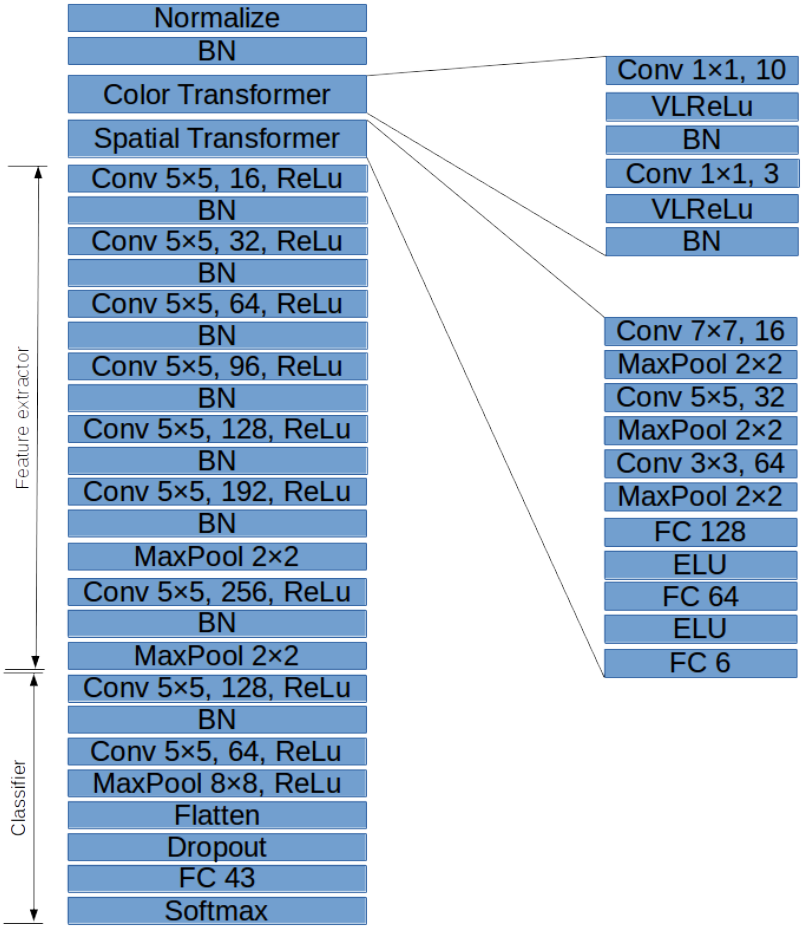

Figure 9: Full network

Spatial transformer network

The STN class is defined in this component as a customized Keras Layer.

To initilize STN, one needs to specify localization net and output size (and input size if STN is the first layer). An example is given below:

```
import spatial_transformer

# Define localization network
def locnet():
    # ...
    return locnet

# Create new STN object, use localization network locnet() and output size of 32 * 32
stn = SpatialTransformer(localization_net=locnet(),
                         output_size=(32, 32))

# ...

# Add STN as as a customized Keras layer
model.add(stn)
```

Training

Training module loads in all data, train, validate and test the model. Model weights with the least validation is kept during training.

Hyper parameters for training is listed in the table below.

Name	Value
Batch size	128
Epochs	150
Optimizer	Adam
Learning rate	0.001
Learning rate decay	0.01
L2 regularization lamda	0.05
Drop out	0.6

Refinement

Various hyper parameters and network architecture has been tested for the designed model.

The first attempt for the project was to use transfer learning to extract bottleneck features from Inception model [19], however the accuracy plateaued around 80% accuracy. Then an attempt to fine tune the Inception model failed due to limited computation power. Even with all inception modules' weights freezed, the final layers takes more than 10 minutes to train one epoch on a GTX1080.

Then the model proposed in [5] with spatial transformer network is implemented and training the model from ground up, without preprocessing and augmentation, the model failed to exceed human Performance.

With better network architecture, color transformer and batch normalizaion implemented, the model was able to ahieve accuracy above 99.00%. Tested with multiple hyper parameter sets, model performance reached 99.59% on our testing dataset.

IV. Results

Model Evaluation and Validation

After training the model, it is tested on the testing dataset and achieved an accuracy of 99.59%, which is higher than the highest published accuracy[5]. The model is able to predict all kinds of labels with high accuracy and confidence. An interactive testing environment is provided in jupyter notebook.

`Usage:`

```
jupyter notebook test_model.ipynb
```

The high accuray has shown the model is able to generalizes well on unseen data and give a precise prediction.

A robustness test is conducted using a dozen of new images downloaded from internet. The following Figure11 and Figure 12 shows the result of the predictions.

Speed limit (70km/h)
Stop
No passing
Speed limit (80km/h)
Speed limit (30km/h)

Speed limit (120km/h)
Bicycles crossing
Yield
Priority road
End of all speed and passing limits

Bumpy road
No entry
General caution
Pedestrians
Beware of ice/snow

Speed limit (30km/h)
Yield
End of all speed and passing limits
End of no passing
Priority road

Keep left
Turn left ahead
Roundabout mandatory
Ahead only
Go straight or right

No entry
Turn right ahead
Speed limit (80km/h)
Ahead only
Keep right

General caution
Priority road
Stop
No entry
Speed limit (80km/h)

No passing
Roundabout mandatory
Keep left
Turn left ahead
Go straight or right

Speed limit (30km/h)
Bumpy road
General caution
No passing
No vehicles

No vehicles
Speed limit (70km/h)
Speed limit (80km/h)
Yield
Priority road

No entry
Dangerous curve to the left
Go straight or right
Roundabout mandatory
Stop

Speed limit (120km/h)
Speed limit (60km/h)
Roundabout mandatory
Vehicles over 3.5 metric tons prohibited
Speed limit (80km/h)

As we can see, the model performs very well on trained labels with very high confidence. However, it is unable to predict labels which aren't trained. Such as the "end of speed limit 60km/h" and "pedestrains only" are miscategorized since thet are not in the training label set.

Overall, given the accurate and highly confident prediction results, I think the model is a robust enough solution for autonomous vehicle systems. With more data set trained, this model could be implemented for production.

Justification

Compared to human performance given by [7], we hereby compare the categorical accuray of the two models in the following table.

Type	Overall	Blue	Danger	End of	Red round	Red other	Speed	Special
Our model	99.59%	99.60%	99.53%	100%	99.81%	99.87%	99.69%	99.17%
Human performance	98.84%	99.72%	98.67%	98.89%	98.00%	99.93%	97.63%	100%

As it is shown above, except "Red other" and "Special" category, our model has higher accuray than human performance.

Here we use McNemar's test to judge if the improvement is significant:

H_0: the two marginal probabilities for each outcome are the same, $p_b = p_c$

H_1: $p_b \neq p_c$

```
> oldclassif <- c (rep ("correct", 12483), rep ("wrong", 147))
> newclassif <- c (rep ("correct", 12578), rep ("wrong", 52))
> table (oldclassif, newclassif)
          newclassif
oldclassif correct wrong
   correct   12483     0
   wrong        95    52
> mcnemar.test (oldclassif, newclassif)

    McNemar's Chi-squared test with continuity correction

data:  oldclassif and newclassif
McNemar's chi-squared = 93.011, df = 1, p-value < 2.2e-16
```

Since p is less than typical threshold 0.05, we reject the null hypothesis. Thus the model performance improvement is significant.

V. Conclusion

Free-Form Visualization

Figure 13 shows all images that the model failed to predict.

These images are subject to severe flaws such as strong shadow/flare, obstructions, low illumination, motion blur etc. Even humen struggle to recognize.

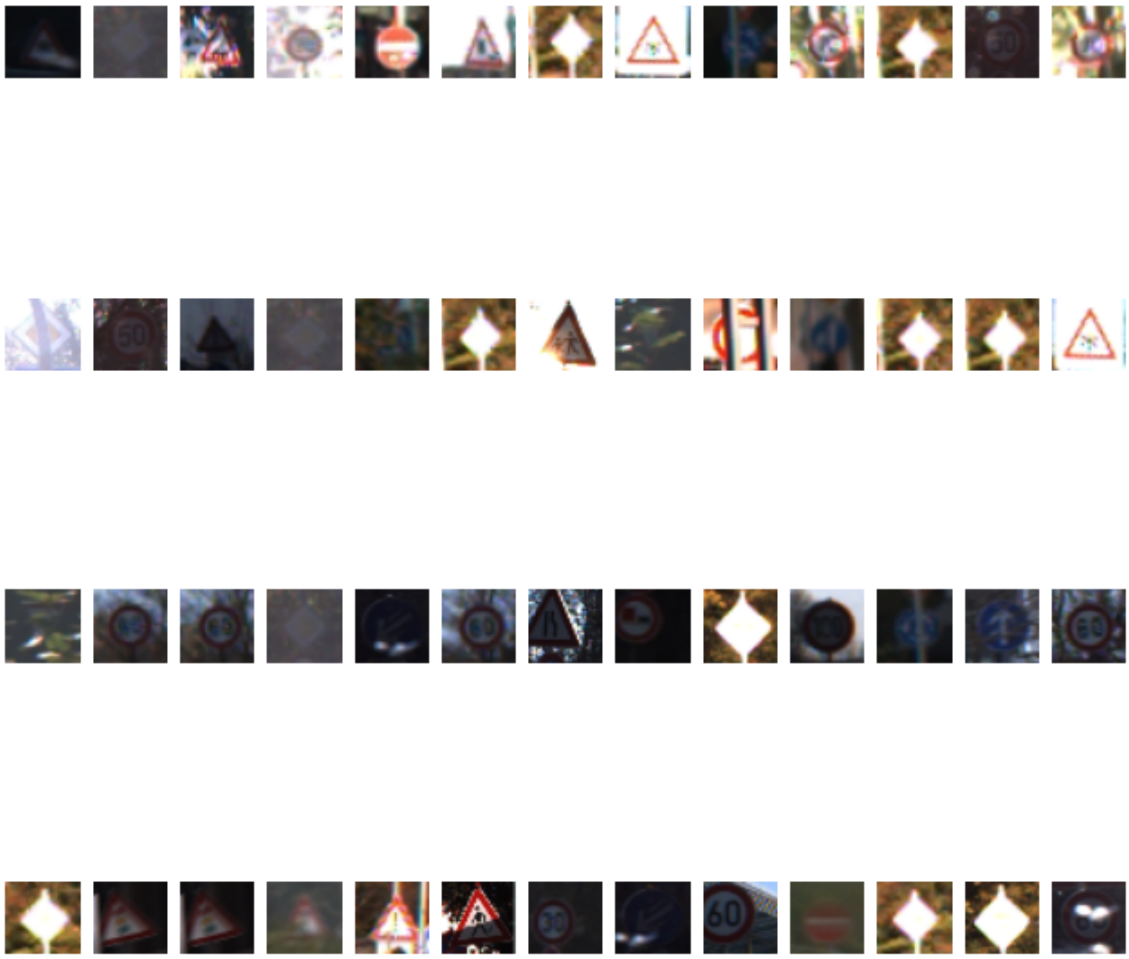

Figure 10: Failed images

Reflection

This project has provided a novel apporach for traffic sign recognition with simplified pipline while its performance still exceeds the best published results. With learned color and spatial transformation, the model is able to adapt to image's charactoristics without external support. This essentially enables model to achieve better model performance with less data. Since the transformers can be inserted into networks as an standard layer, these methods are also applicable for any other image recognition models.

Improvement

Due to limited computation power, parameter sweeping was not conducted. It is very possible this model could yield higher performance with other hyper parameter sets. Furthermore, The model architecture can be further optimized.

In [12], the author has demonstrated using multiple spatial transformer networks could improve performance. A better sptial transformer model is proposed in [20], which has prevented information loss during multiple affine trasformations.

This project aims to test the highest performance without preprocessing or augmenting the dataset. However, it is expected that model performance could be further improved with preprocessing and augmentation.

Current model is trained on a very limited dataset only, it is unable to categorize any other traffic signs which it is not trained on. With more data, we can expect the model become able to classify more sign images.

References

[1]: De la Escalera, Arturo, J. Ma Armingol, and Mario Mata. "Traffic sign recognition and analysis for intelligent vehicles." Image and vision computing 21.3 (2003): 247-258.

[2]: Miura, Jun, et al. "An active vision system for on-line traffic sign recognition." IEICE TRANSACTIONS on Information and Systems 85.11 (2002): 1784-1792.

[3]: Sermanet, Pierre, and Yann LeCun. "Traffic sign recognition with multi-scale convolutional networks." Neural Networks (IJCNN), The 2011 International Joint Conference on. IEEE, 2011.

[4]: J. Stallkamp, M. Schlipsing, J. Salmen, C. Igel, Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition, Neural Networks, Available online 20 February 2012, ISSN 0893-6080, 10.1016/j.neunet.2012.02.016. (http://www.sciencedirect.com/science/article/pii/S0893608012000457) Keywords: Traffic sign recognition; Machine learning; Convolutional neural networks; Benchmarking

[5]: Multi-column deep neural network for traffic sign classification, Multi-column deep neural network for traffic sign classification, D. Ciresan, U. Meier, J. Masci, J. Schmidhuber, August 2012, Neural Networks (32), pp. 333-338

[7]: Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition, Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition, J. Stallkamp, M. Schlipsing, J. Salmen, C. Igel, August 2012, Neural Networks (32), pp. 323-332

[8]: Traffic sign classification using K-d trees and Random Forests , Traffic sign classification using K-d trees and Random Forests , F. Zaklouta, B. Stanciulescu, O. Hamdoun, August 2011, International Joint Conference on Neural Networks (IJCNN) 2011

[9]: Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition, Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition, J. Stallkamp, M. Schlipsing, J. Salmen, C. Igel, August 2012, Neural Networks (32), pp. 323-332

[10]: Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition, Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition, J. Stallkamp, M. Schlipsing, J. Salmen, C. Igel, August 2012, Neural Networks (32), pp. 323-332

[11]: Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition, Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition, J. Stallkamp, M. Schlipsing, J. Salmen, C. Igel, August 2012, Neural Networks (32), pp. 323-332

[12]: Jaderberg, Max, Karen Simonyan, and Andrew Zisserman. "Spatial transformer networks." Advances in Neural Information Processing Systems. 2015.

[13]: Mishkin, Dmytro, Nikolay Sergievskiy, and Jiri Matas. "Systematic evaluation of CNN advances on the ImageNet." arXiv preprint arXiv:1606.02228 (2016).

[14]: Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." arXiv preprint arXiv:1502.03167 (2015).

[17]: Xudong Cao, "A practical theory for designing very deep convolutional neural networks", 2015

[18]: Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." arXiv preprint arXiv:1502.03167 (2015).

[19]: Szegedy, Christian, et al. "Rethinking the inception architecture for computer vision. CoRR abs/1512.00567 (2015)." (2015).

[20]: Lin, Chen-Hsuan, and Simon Lucey. "Inverse Compositional Spatial Transformer Networks." arXiv preprint arXiv:1612.03897 (2016).

[21]: Chollet, François. "Keras." (2015).

[22]: Abadi, Martín, et al. "Tensorflow: Large-scale machine learning on heterogeneous distributed systems." arXiv preprint arXiv:1603.04467 (2016).