Computer Vision, Fall 2021

Evaluation Of Multiple CNN Approaches for German Traffic Sign Recognition Benchmark

Ankit Sati – as14128

New York University

Abstract – We explore multiple approaches for the German Traffic Sign Recognition (GTSRB) problem. We start testing with slightly modified versions of Lenet5, Alexnet, Googlenet models. Lastly, we implement two Convolution Neural Networks (CNNs), one of them augmented with Spatial Transformer Networks (STNs) to boost accuracy making the model insensitive to large variations in the input images. These two models report an accuracy of 98% and 99% respectively on GTSRB dataset.

GitHub Link:
https://github.com/Satiankit96/Data-Science.git

Keywords: deep neural networks, traffic signs image classification, spatial transformer networks, image preprocessing.

1. Introduction

Traffic sign recognition are important in many real-world applications like selfdriving cars, road network management and traffic surveillance. The traffic signs are designed in such a way that each one is distinct, easily visible in all weather conditions and easily differentiable from each other because they are fundamental to road safety. This makes the recognition problem at hand very constrained. The development of not only an accurate system but also a very robust one that can translate to 100% confidence in practical applications is imperative. The GTSRB dataset presents a number of challenges due to real-world variabilities such as variations in point of views, physical degradation, unpredictable or harsh weather, abrupt lighting conditions, motion blur, sun glare, fading signposts, graffiti, stickers. Moreover, the dataset has image of resolutions as low as 15x15. Any traffic sign recognition model must be able to handle such issues. This project is an attempt to solve the problem and build upon the previous works by [1] and [2]. We test 3 preexisting models LeNet5, AlexNet and GoogleNet (slightly modified to suit the problem). We then implement a simple CNN model that we use as reference for incorporating the Spatial transformer network (STN) modules. The models are tested on the GTSRB dataset. The CNN architecture with STN outperforms all the other models.

The rest of the report is organized as follows: section 2 presents the preprocessing methods used for all models except the one that incorporates STN. Sections 3 to 7 present each of the models in order along with their individual results. Section 8 briefly compares the results of all the models. Section 9 draws conclusions and proposes future work.

2. Data preparation for LeNet5, AlexNet, GoogleNet and CNN model:

Pre-processing: All the images are up sampled or down sampled to 32x32 and shuffled. We shuffle the training data to increase randomness and variety in the training dataset and create a more stable model. During out first testing with the models we found out that the images not being successfully classified were the ones with high clutter, degradation, abrupt changes in contrast(shadow) and big translations. According to the findings in [1] we then convert the images to YUV space as turning the images to grayscale improved accuracy. The U (blue projection) and V (Red projection) are left unchanged. The local histogram equalization and normalization spreads out the most frequent intensity values and changes the range of pixel intensity values normalizing the global and local contrast of the images. For all the models (except CNN model with STN) the input images are randomly rotated ([-20, +20] degrees) and randomly changed in scale ([0.9, 1.1] ratio) and then passed through the network. We don't perturb the images too much since it will change the meaning of the image (traffic sign) and the model learns to associate labels to wrong signs. The results were improved across all models.

Pre-processed Images

Model	Perturbed	Non perturbed
LeNet5	95.1	94.2
AlexNet (with Adam)	96.9	95.2
GoogleNet	98.1	97.4
CNN w/o STN	98.8	97.1

Table 1: Validation Accuracy for Perturbed and Non-Perturbed Data

3. LeNet5

[2] architecture was first proposed for handwriting recognition, but some changes can help us make it more efficient. We use mini batch gradient descent as out optimizer and dropout for regularization. The higher model update frequency allows for more robust convergence avoiding local minima. Batch size of 16 and 32 yield best results with lower size giving diminishing returns. The dropout layer randomly deactivates neurons from the forward/backward pass modifying the network itself rather than the cost function. The model shows slightly higher accuracy when trained with randomly perturbed examples.

LeNet5	Validation set	Test set
Accuracy	95.1	94.2

Table 2: Accuracy Table for LeNet5

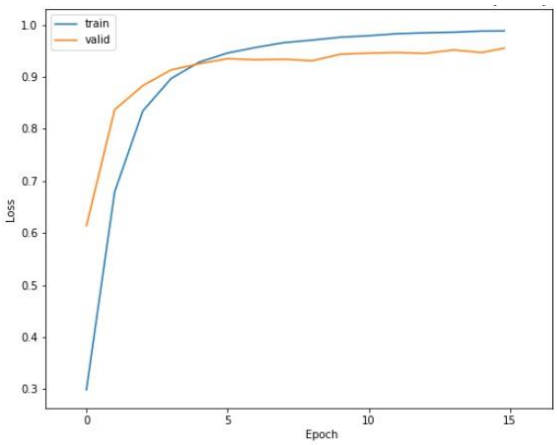
4. Alexnet

The input dimension for out dataset is 32x32x3 hence the model is modified as shown in table 3. We experiment with two different techniques to increase the accuracy of the model. First was to decay the learning rate. Exponential learning rate decay with

Layer	(Kernel, stride, pad)
Conv	(5, 1, valid)
Pool	(2, 2)
Conv	(3, 1, valid)
Pool	(2, 2)
Conv	(3, 1, same)
Conv	(3, 1, same)
Conv	(3, 1, same)
Pool	(2, 2)
Flat	-
FC	-
FC	-
FC	O/P 43 probabilities

Table 3: AlexNet Model as Implemented

decay steps 150 and decay rate 0.95 posed to be an effective in increasing the accuracy of the model. For the second test we implemented the Adam optimizer that simulates the learning rate decay without us having to tune the two extra hyperparameters like in the case of the exponential learning rate decay. L2 regularization with regularization parameter lambda value of 1e5 to prevent overfitting gave us the best accuracy of 97.2%.



Accuracy History for AlexNet (with Adam) over 15 Epochs

AlexNet	Validation set	Test set
Exp decay	95.3	93.1
Adam	96.9	96.8

Table 4: Accuracy Table for AlexNet

5. GoogleNet

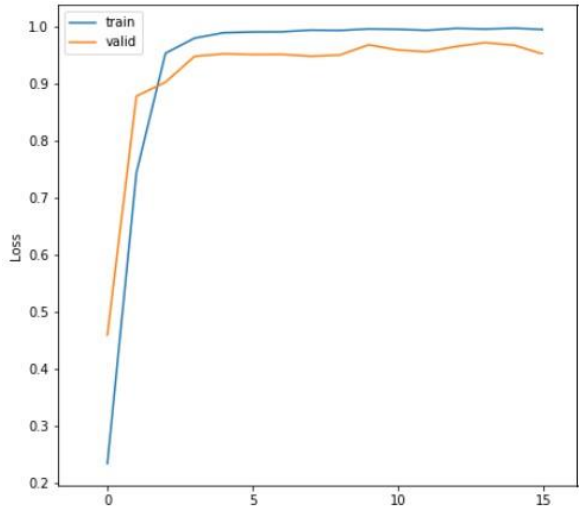
We reduce the model size to make the training more manageable. We introduce two inception modules with dimension reductions after the first layer(conv), second layer (max pool) and third layer (max pool), followed by an avg pooling and a fully connected layer. As found in [1] using overlapping pooling can reduce error rates and make the model more difficult to overfit. With this model we get out highest validation accuracy yet of 97.9%.

Layer	Params
Conv (3, 2)	1792
Inception	137072
Inception	388736
Max pool (3, 2)	
Inception	433792
Inception	449160
Max pool (3, 2)	
Inception	859136
Inception	1444080
Avg Pool (3, 1)	
Flatten (864)	
Full (43)	44032

Table 5: GoogLeNet architecture

GoogleNet	Validation set	Test set
Accuracy	98.1	97.3

Table 6: Accuracy Table for GoogleNet



Accuracy for GoogleNet

6. CNN model

The aim was to set and test a simple CNN architecture that could perform well and be used as a benchmark/reference for the CNN+STN architecture (in the section below).

Layer	Output
Conv2d	28,28,32
Conv2d	24,24,32
Max pool	12,12,32
Dropout	12,12,32
Conv2d	10,10,64
Conv2d	8,8,64
Max pool	4,4,64
Dropout	4,4,64
Flatten	1024
Full conn	256
Dropout	256
Full conn	43

Table 7: CNN Model

Using ReLU as activation function and Adam as out optimizer we get the following results:

Validation set	Test set
98.8	98.2

Table 8: Accuracy with the CNN model for test and validation sets.

7. CNN with STN modules

Even though all the above models were given randomly perturbed images, the system did not prove to be very robust for testing data.

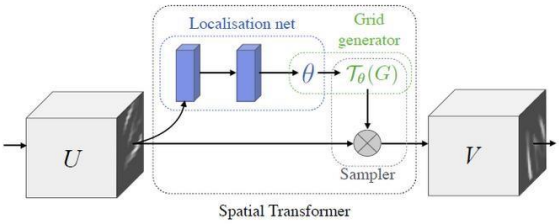
The models also required heavy preprocessing for them to be able to classify the images accurately. To train them to be able to classify such images we had to alter the dataset with random but manual perturbations. With these models many of the distorted images were still incorrectly labeled. Hence instead of having some of the data manually perturbed or modified it could be better to have a network learn to ignore those distortions and then classify the image.

Examples of wrong predictions by previous models

CNNs are locally translation invariant because of the inherent nature of the max pooling operations. But they are local so if the translations are too big the CNN model will get confused. STNs help to get focus around the area of interest and ignore the

perturbation. The resulting output is easier to classify.

Inspired by the approach by [1] the implement method is also a single column CNN combined with spatial transformer, an ReLU activation function, local contrast normalization and max pooling. We experiment with two different optimization algorithms (non-adaptive method Stochastic gradient descent SGD and adaptive method Adaptive Moment Estimation Adam) to compare their effectiveness. We finally present the final model that gives the highest accuracy.



a. Data Preprocessing:

For this approach we don't convert the dataset to YUV space. We up sample or down sample the 48x48 size and carry a local contrast normalization with gaussian kernels. It centers the input image around its mean value and enhances the edges. The dataset is also not perturbed or jittered in any form, since STN will map its output with its input image/tensor. Spatial transformer modules perform a geometric transformation (affine transformations) on the input (tensor or image). One of the objectives of using an STN is to not have to perturb or heavily preprocess the dataset and still have the model take in account the possible clutter, scale difference, rotations etc. in the input images. STN modules can be inserted into CNN architectures since the parameter for transformation (A sub theta - A_Θ) applied to the feature map are learnt by back propagation. x_i^t and y_i^t are known to us since they are the known target location of the pixel (G_i is the i^{th} grid point in the target space). V_c gives us the value at that location.

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

Affine transformation

It gets output with bilinear interpolation or sampling. We do summation over U for the channel 'c' of pixel 'n, m'. If x_i^s is equal to m, the value of $[1 - |x_i^s - m|]$ will be equal to 1 and similarly for y_i^s if equal to n, $[1 - | y_i^s - n|]$ will be 1. So essentially, we are copying U_{nm} of that particular channel to the current pixel V_c. We ignore the summation since the function is maximum once but most of the times it is 0 except in the neighborhood of x_i^s and y_i^s. Most of the time we just copy the input to the output.

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

Bilinear sampling

Theoretically it should be enough to have an STN module in the beginning of the Convolutional layers so that we can transform the image and then do the processing, but from the findings of we decide to implement the model where multiple STN modules are placed in between the convolutional layers. The architecture is as shown in table 9 with localization network of STNs; STN1, STN2, STN3 shown in table 10, 11, 12 respectively.

Layer	Description	Kernel
0	Input	
1	STN1	
2	Conv1 + ReLU	7x7
3	Max Pool	2x2
4	LCN	
5	STN2	
6	Conv2 + ReLU	4x4
7	Max Pool	2x2
8	LCN	
9	STN3	
10	Conv3 + ReLU	4x4
11	Max Pool	2x2
12	LCN	
13	Full + ReLU	
14	Fully connected	
15	Softmax(43 classes)	

Table 9: CNN architecture with STN modules

No.	Layer
1	MaxPool2d (2, stride=2)
2	Conv2d (3, 250, kernel size=5, stride=1, padding=2)
3	ReLU
4	MaxPool2d (2, stride=2)
5	Conv2d (250, 250, kernel size=5, stride=1, padding=2)
6	ReLU
7	MaxPool2d (2, stride=2)
8	Linear (9000, 250)
9	ReLU
10	Linear (250, 6)

Table 10: ST1 in Table 9

8	Linear (800, 300)
9	ReLU
10	Linear (300, 6)

Table 11: ST2 in Table 9

No.	Layer
1	MaxPool2d (2, stride=2)
2	Conv2d (250, 150, kernel size=5, stride=1, padding=2)
3	ReLU
4	MaxPool2d (2, stride=2)
5	Conv2d (150, 200, kernel size=5, stride=1, padding=2)
6	ReLU
7	MaxPool2d (2, stride=2)
8	Linear (200, 300)
9	ReLU
10	Linear (300, 6)

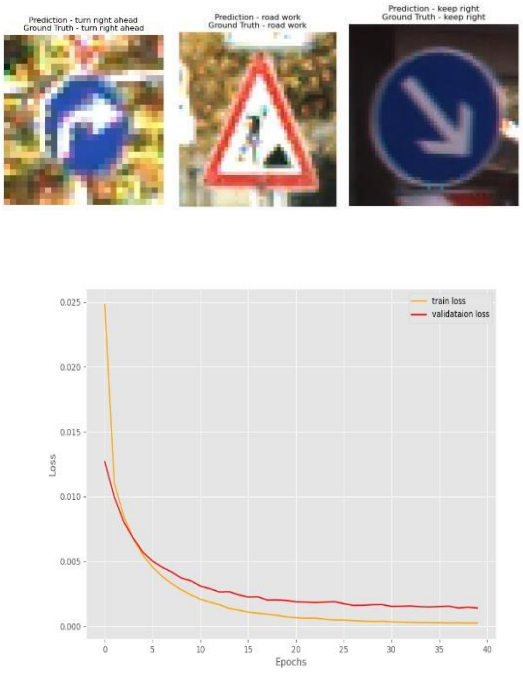
Table 12: ST3 in Table 9

The architecture is tested with SGD and Adam. Adam combines RMSProp and momentum by storing both the individual learning rate of RMSProp and the weighted average of momentum. Adam estimates the gradient as the momentum parameter to improve training speed. The model war trained with Adam with learning rate of (0.0001), exponential decay rate for first moment estimate (0.9), exponential decay rate for second moment estimate (0.999). The model was trained with SGD with no momentum or weight decay and learning rate of 0.01.

	Accuracy	Loss
Adam	98.6	0.0002
SGD	99.1	0.0001

No.	Layer
1	MaxPool2d (2, stride=2)
2	Conv2d (200, 150, kernel size=5, stride=1, padding=2)
3	ReLU
4	MaxPool2d (2, stride=2)
5	Conv2d (150, 200, kernel size=5, stride=1, padding=2)
6	ReLU
7	MaxPool2d (2, stride=2)

perform worse than non-adaptive SGD. Adam also required two more hyperparameters to be tuned which amplifies the hyperparameter tuning problem.



Correct Predictions and Loss over 40 epochs

8. Final Results

Out of all the models trained and experimented with, the CNN architecture with STN modules gave us the highest accuracy with minimal training and testing loss. The model is relatively immune to adversarial examples and can classify them with a higher accuracy than compared to other models presented in this project.

Table 13: Accuracy of CNN + STN model with Adam and SGD methods.

The results of this project coincide with the findings of [1] in that although the adaptive optimization methods like Adam have empirically shown to give better accuracy with minimal training time, in this case they

	Accuracy	Data perturbed
LeNet5	95.1	Yes
AlexNet	96.9	Yes
GoogleNet	98.1	Yes
CNN	98.8	Yes
CNN + spatial transformers	99.1	No

9. Conclusions and future work

This project implements various CNN models for automatic recognition of traffic signs on the German Traffic Sign Benchmark. The LeNet5, AlexNet, GoogleNet and the basic simple CNN architectures in this project give a fairly high recognition accuracy but are not enough for practical applications. The CNN model with STN modules implemented outperforms all the other models in this project and achieves a recognition rate accuracy of 99.1% in the GTSRB dataset. The experiments report that the inherent ability of STN to transform input images prove to be vital in achieving higher recognition rate accuracy of the traffic signs. The model requires no augmentation or

manual perturbation of the dataset like in other CNN models.

Future work possibilities include but are not limited to study/development of a single neural network that could provide even higher recognition rate accuracy for country with similar traffic signs. The scope of more robust models that give unparalleled accuracy and are completely immune to heavily perturbed, adversarial examples is present Its practical application in, say, selfdriving cars require 100% confidence since they may pose security concerns that may cause negative effects and may endanger drivers and pedestrians alike.

REFERENCES

[1] Arcos-García, Álvaro; Álvarez-García, Juan A.; Soria-Morillo, Luis M. Deep neural network for traffic sign recognition systems: An analysis of spatial transformers and stochastic optimisation methods. Neural Networks, 2018.

[2] Sermanet, P., & LeCun, Y. Traffic sign recognition with multi-scale Convolutional Networks. The 2011 International Joint Conference on Neural Networks, 2011.

[3] LeCun, Y., Bottou, L, Bengio, Y, and Haffner, P. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11):2278–2324, 1998

[5] Robbins, Herbert E. "A Stochastic Approximation Method." Annals of Mathematical Statistics 22 (2007): 400-407, 2004

[6] Kiefer, J., and Wolfowitz, J. Stochastic Estimation Of the maximum of a regression function, Cornell university.

[7] Rad, Mahdi & Roth, Peter M. & Lepetit, Vincent (2020). ALCN: Adaptive Local Contrast Normalization, 2020.

[8] J. Stallkamp, M. Schlipsing, J. Salmen, C. Igel. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition, Neural Networks, Volume 32, 2012, ISSN 0893-6080, 2012.

[9] J. Stallkamp, M. Schlipsing, J. Salmen and C. Igel, "The German Traffic Sign Recognition Benchmark: A multi-class classification competition," The 2011 International Joint Conference on Neural Networks, 2011, pp. 1453-1460, doi: 10.1109/IJCNN.2011.6033395.

[10] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincant Vanhoucke, Andrew Rabinovich, Going Deeper with Convolutions, 2014.

[11] Krizhevsky, A., Sutskever, I. & Hinton, G. E. ImageNet Classification with Deep Convolutional Neural Networks. In F. Pereira, C. J. C. Burges, L. Bottou & K. Q. Weinberger (ed.), Advances in Neural Information Processing Systems 25 (pp. 1097--1105), 2012.

[12] Diederik P. Kingma, Jimmy Ba. Adam: A Method for Stochastic Optimization, 2017.

[13] Zhu, Z., Liang, D., Zhang, S., Huang, X., Li, B., & Hu, S. Traffic-sign detection and classification in the wild. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 2110–2118), 2016.

[14] Shadeed, W., Abu-Al-Nadi, D., & Mismar, M. Road traffic sign detection in color images. 10th IEEE International Conference on Electronics, Circuits and Systems, 2003.

[15] Szegedy, C., Ioffe, S., Vanhoucke, V., & Alemi, A. A. Inception-v4, inception-resnet and the impact of residual connections on learning. In AAAI (pp. 4278–4284), 2017.

[14] Ciresan, D., Meier, U., Masci, J., & Schmidhuber, J. Multi-column deep neural network for traffic sign classification. Neural Networks, 32, 333–338, 2012.

[15] De La Escalera, A., Moreno, L. E., Salichs, M. A., & Armingol, J. M. Road traffic sign detection and classification. IEEE transactions on industrial electronics, 44, 848–859, 1997.

[16] Gudigar, A., Chokkadi, S., Raghavendra, U., & Acharya, U. R. Local texture patterns for traffic sign recognition using higher order spectra. Pattern Recognition Letters, 94, 202 – 210, 2017.

[17] Huval, B., Wang, T., Tandon, S., Kiske, J., Song, W., Pazhayampallil, J., Andriluka, M., Rajpurkar, P., Migimatsu, T., ChengYue, R. et al. (2015). An empirical evaluation of deep learning on highway driving. arXiv preprint arXiv:1504.01716, 2015.

[18] Jaderberg, M., Simonyan, K., Zisserman, A. et al. Spatial transformer networks. Advances in Neural Information Processing Systems, (pp. 2017–2025), 2015.