Traffic Sign Recognition

Deep Learning – 046211

Final Project

Kfir Levi – 208017038

Li-or Bar David – 314665779

Semester – Spring 2021

Abstract

In this project we tried to classify the German Traffic Sign Recognition Benchmark[1] (GTSRB) dataset, which consists of 43 classes and over 39,000 samples of 30×30 traffic sign images. for example:

We tested multiple architectures, including a network that was proposed in an online article, a network we saw earlier in the course and two pre-trained networks where we will train a new output layer to have 43 outputs, using transfer learning with 'fine tuning' method.

Project goal

Our aim is to take a network implemented in TensorFlow and implement it in PyTorch, while also trying to improve the results by using other models, and hopefully compete with the top performers on this dataset. We will use tools obtained in this course such as using pre-trained models and transfer learning, building an architecture of our own and more.

The Dataset

As mentioned above, we will work with GTSRB[1] dataset. The dataset was taken, processed and published by a german group of developers and was first used in a competition at IJCNN 2011.

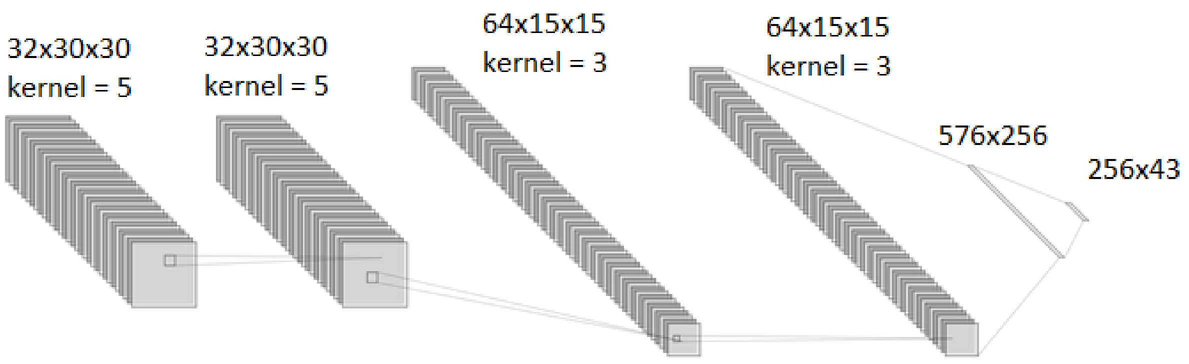
The dataset is well organised and in addition to having a separation to folders for each class, it also has csv files that contain labels for each sample, and therefore allows the freedom to use it however one wishes to.

With more than 40 classes, over 50,000 images overall with each traffic sign only appearing once in the whole dataset, and also having a reliable ground-truth data, this dataset is very useful when trying to learn real-life signs classification.

Another feature of the dataset is having the ground truth necessary for object detection - the height and width of a bounding box and it’s top left corner’s location, while also having the bottom left corner if needed.

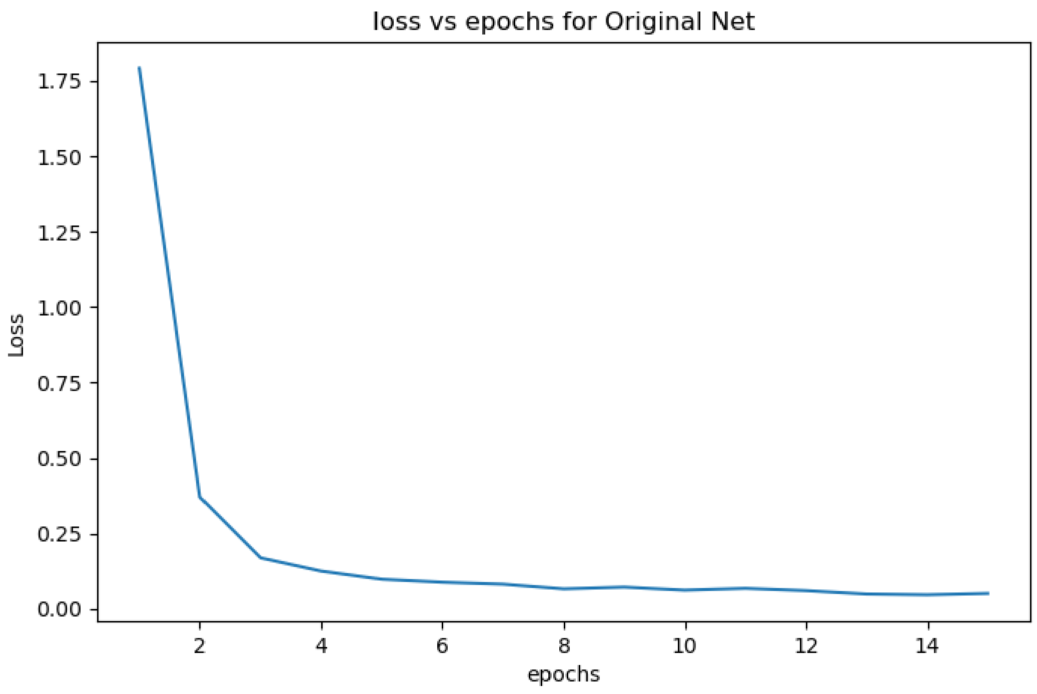
The Original Architecture

The first model we will test is a model that was proposed online, and the model is:



The model has 2 convolution blocks, each includes 2 convolutional layers, one dropout layer and using ReLU as the activation function, and max pooling at the end. The parameters for each layer are as shown in the diagram above. After the convolution part, there are 2 fully-connected layers. The input to the first FC layer is 576, as we calculated in the code.

The results are:

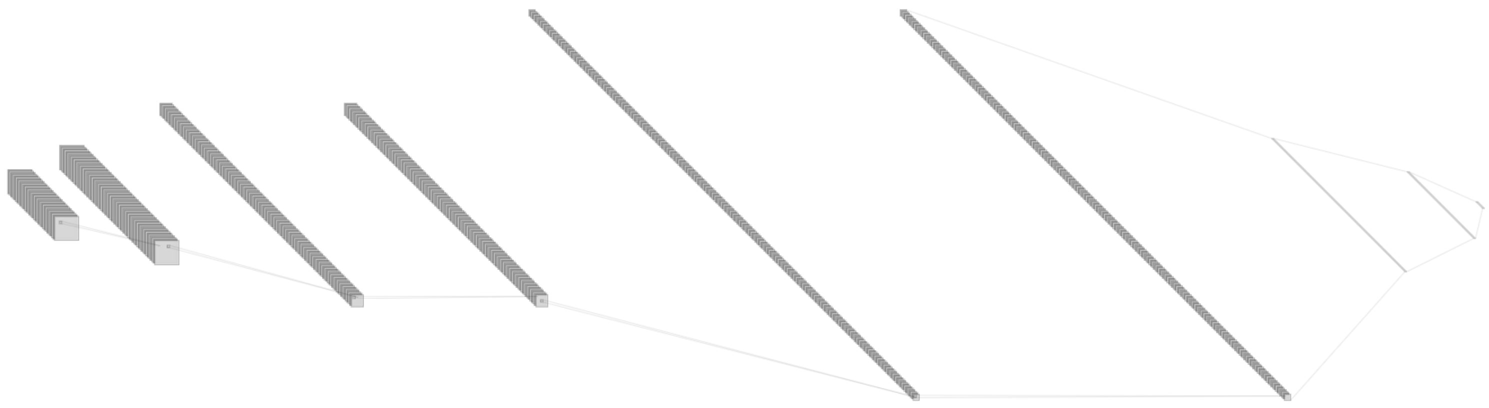


When we used the cross entropy cost function with one-hot vectors.

And the final accuracy score on the test set is: 96.06%

The CifarCNN Architecture

The next model we tested was the model we saw in tutorials 6, 7 about convolutional neural networks (and also modified for the homework). The model has 3 convolutional blocks, uses batch norm and ReLU as the activation function. In case it helps, the diagram is (we are aware it is far from clear):



We can see that we have 6 convolutional layers in total, with number of kernels increasing and size of input decreasing as we progress in the architecture. Outside of this architecture being deeper and wider than the previous one, it uses batch norm layers as well, and therefore we expect it to perform better than the original. The accuracy score on the test set is 96.785%, which is a slight improvement over the previous network.

Pre-trained Models with Transfer Learning

Now, after we tried two models of our own and trained them ourselves, we will turn to more powerful models and will try to modify them to suit our needs. We chose two models to test – the ResNet-18 and DenseNet models. In both of them, we will change the last layer, the output layer, to be with 43 output neurons instead of the 1000 outputs they have for ImageNet. The results are 96.825% for ResNet-18 and 98.060% for DenseNet, both are better than the CifarCNN and the original net, but are a lot more expensive to train and even saving the weights alone is much more memory consuming.

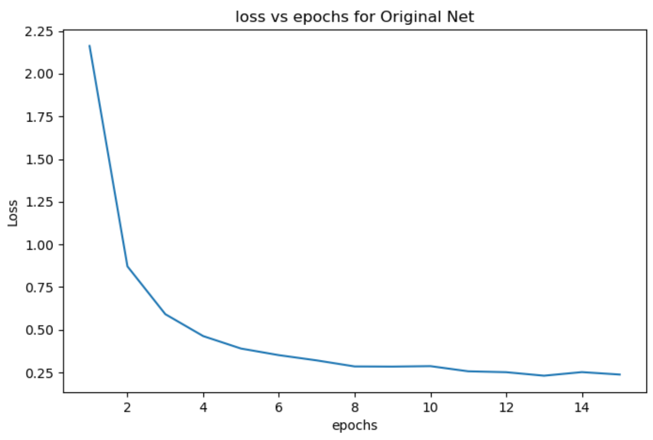
Data Augmentation

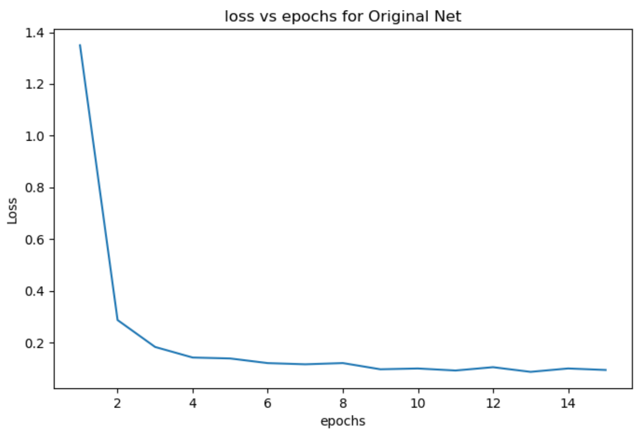
After training each model, we trained all of them again, this time adding data augmentation to the training process. We didn't want to change the original data a lot, so we used some 'gentle' augmentations. First, we added horizontal and vertical translation of 10, which generalizes the data in the aspect of not being centered around the traffic sign, and also added color jitter for brightness with parameter of 0.5 . This will simulate the traffic sign images being taken at different times of the day. All the augmentations we chose represent changes that we believe are likely to happen when encountering a traffic sign in real life (maybe when using autonomous driving) and that is why we think they are important.

Results

	Original	CifarCNN	ResNet-18	DenseNet
Original Data	96.097%	96.785%	96.825%	98.060%
Augmentation	96.793%	97.356%	98.480%	98.393%
Noise	59.691%	49.042%	83.753%	79.881%

Like we saw earlier, we are able to achieve better results the more complicated the nets are. we can also see that the augmentations we chose actually improve the performance as well. we also added random gaussian noise to the test data and inputed it into the nets, to test their robustness to noise. Expectedly, we saw major decrease in the accuracy of the nets, but one can see that both ResNet-18 and DenseNet achieve respectable results, even though we didn't modify the data to increase noise robustness, and we assume that if we add random gaussian noise as part of the data augmentation during training, we will get an even greater noise robustness from the models.
The affects of training with and without data augmentation, respectively: (e.g. the original classifier)

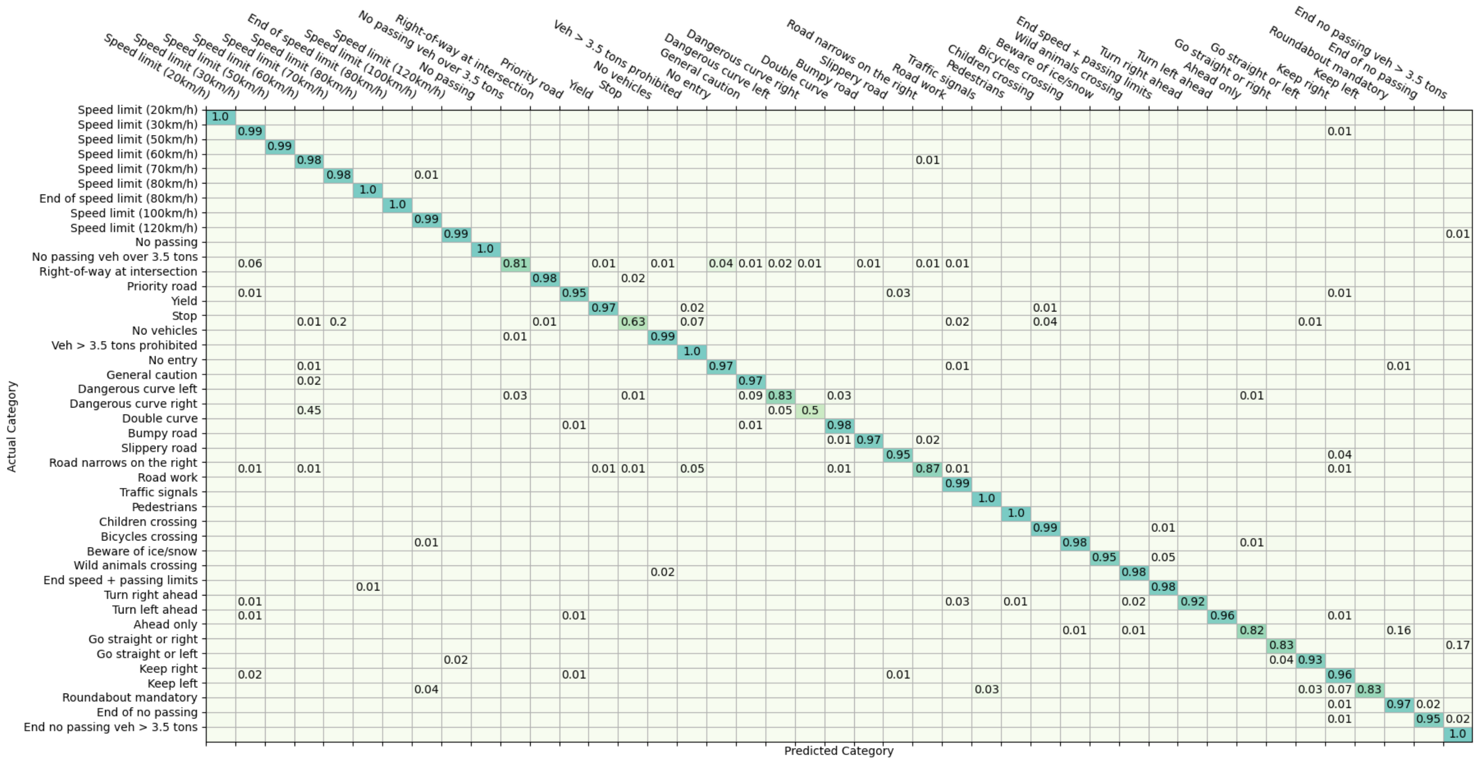




we can see that the task with the augmentation is more difficult and that is why the net converges slowlier in comparison to the task without augmentation, but eventually achieves better accuracy on the test, and this is the criterion that we look to improve.

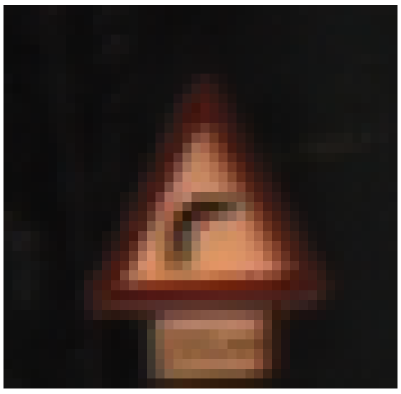
Discussing The Results

Now we will try to analyse the results and when the algorithm has problems. We will use the confusion matrix of the original classifier because he is the most interesting one. The confusion matrix:

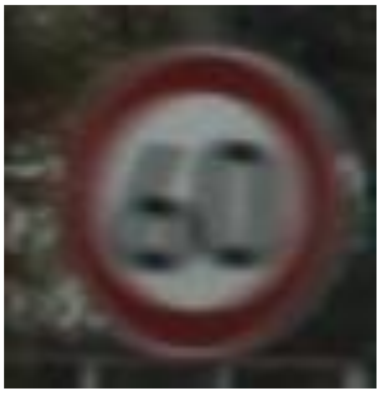


Notice that some of the classes are not classified very goo d, for example the model classifies the 'dangerous curve right' (class 20) as 'speed limit (60 km/h)' (class 3) on 45% of the samples, classifies as 'dangerous curve left' (class 19) on 5% of the samples and classifies correctly on 50% of the samples.

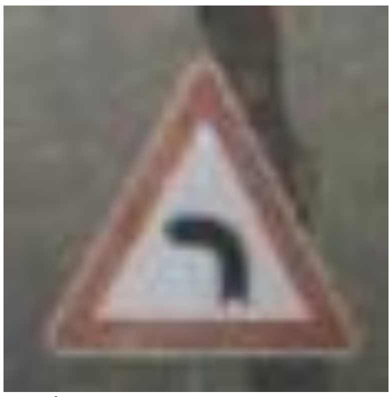
Ground Truth: 20



Ground Truth: 3



Ground Truth: 19



We can guess that the confusion happens since 'speed limit (60 km/h)' also has a curvature nature to it, but it is also very important to add that we have 1410 samples of class 3 and only 360 samples of class 20 (and 330 for class 21) and because dataset imbalances create a bias towards classes that appear more, we can assume it affects our classifying ability as well.

References

[1] - J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel. The German Traffic Sign Recognition Benchmark: A multi-class classification competition. In Proceedings of the IEEE International Joint Conference on Neural Networks, pages 1453–1460. 2011.