**Traffic Sign Recognition**

**Introduction**

Computer vision has become in recent years one of the hottest subjects in computer science. This fact comes with no surprise – it is an interesting intersection between image processing, deep learning, and systems theory. The interest taken in this field is also a result of automation and the rise of self-driving cars, which cannot be safely deployed without reliable systems capable of interpreting the world around them. A primary function of such systems is to recognize traffic signs to determine the safest way to navigate through that portion of the street. The goal of this paper is to find ways to improve accuracy of a model either though tinkering with the convolution layers, image editing or the number of inputs and epochs the system runs on.

**Related Work**

The problem of classifying images is typically one tackled with deep learning algorithms based on convolutional neural networks, a subset of artificial neural network which mimic the way our brains work but on a very small scale. CNN’s are the go-to solutions when it comes to visual supervised learning. There are many ways to approach such a complex problem. An approach is to edit all the input pictures, applying a contour filter, but rendering color features useless or purposely adding blur, stretching and rotation on the images to give the classifier more images to work on. One big disadvantage of the latter is that the training set gets many times bigger, requiring a proportionally longer period for training. It can get prove inefficient to train an adequate model for such big inputs only to get a marginal accuracy gain.

**How It All Works**

The test and validation trains are a subset of the traffic sign images made available by the GTSRB. It contains 39209 images for training and 12569 images for validation, which are different from the training train. As another layer of testing performance, a number of images were selected and modified randomly in order to verify the model’s ability to accurately predict the class of signs it has never seen before. A number of variations of convolution layers were tested to see which had the highest accuracy while also looking at their training times to see which achieved the best performance relative to its training time. The layers used are convolution layers, max pool layers, dropout layers, and dense. We will go into detail for each one of these types.

The network architecture consists of several convolution layers, each followed by a pooling and dropout layers. After several such combinations, the features extracted are flattened and fed into 2 layers of fully interconnected neurons. The first one can contains a variable number of neurons but the second one has to have a calculation unit for each class.

The convolution layers are given a number of filters to be applied to the picture. This number starts small and gets bigger as the information is propagated forward though the CNN. Their function is simply to generate quantifiable information about the properties of the picture, or, more visually, to detect patterns in the image. This is also the reason why the number of filters is getting bigger – the deeper layers combine the previous inputs in order to detect higher-order patterns – circles, squares, or anything that can prove useful in image classification.

The max pooling layer is necessary in order to prevent overfitting. After a number of filters have been applied and their results calculated, this layer picks the one with the highest value and discards them. The reasoning is that only the highest values are relevant to the classification of the image and eliminating the less important features not only saves memory and computation time, but also permits the model to extrapolate its training on new input.

The dropout neurons come in smaller number than their direct predecessors. Their role is to save a percentage of the previous layer’s outputs picked at random and throw away the remaining. This is done, again, to prevent overfitting the input and correct for the cases when the pooling layer is not enough is not enough to prevent this. The downside of using this is the longer training periods and the introduction of randomness in the training of them model. Several training periods can yield very different models.

The last layer, and the most important one for decision making, is the dense layer. Each neuron is interconnected with all the neighboring layers’ neurons. This is where the actual training takes place. The patterns detected earlier by the convolution neurons are fed into all the dense neurons and they prescribe some weights to these inputs. After training, the model knows exactly what shapes are relevant for each class and picks the only most likely to be the correct one.

**Results**

A total of 8 different configurations were trained in order to test their efficacy. The parameters that were tuned are the size of the images after being edited, the number of neurons discarded by the dropout layer, the number of convolution layers and the number of filters they apply and, finally, the total number of epochs the model is trained for.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Image Size | Epochs | Minutes per Epoch | Dense Layer Sizes | Convolution Layers | % Neurons Dropped | Test Acc | Validation Acc | Comments |
| 128x128 | 10 | 4 | 16, 32,  64, 64,  64 | 128, 43 | 0 | 96.6% | 96.6% | Initial configuration. Not very good on inputs never seen before |
| 64x64 | 10 | 1 | 8,  16,  32,  32 | 64, 43 | 0 | 96% | 98% | Comparable performance with the first one but considerably faster. |
| - | - | - | - | - | 20% | 88% | 94.5% | Lower accuracy on the trains but good accuracy on new inputs. |
| - | - | - | - | - | 10% | 93% | 98.5% | Very high accuracy on validation set. Has problems on brightly lit signs. |
| - | - | - | - | 128, 43 | 10% | 92% | 95% | Worse in all aspects compared to its 64, 43 counterpart |
| 32x32 | 10 | 0.66 | - | - | 20% | 83% | 94% | Considerably faster but the high dropout asks for higher number of epochs |
| - | 20 | - | - | - | - | 85% | 95% | Poor performance on new images |
| - | 20 | - | - | - | 10% | 85% | 95% | Average accuracy on new inputs but very good shape detection |

**Conclusion**

There are many ways to fine tune a neural network and, as found out, small changes in the configuration lead to entirely new results after the model has been trained. An important factor to be taken into account is how long the training takes. An interesting curiosity in the case of sign classification is that bulky network architectures with many features and trained for large stretches of time can perform worse than lightweight architectures trained for more modest periods. This is because of overfitting and in machine learning it is an important aspect to be kept in one’s mind.