|  |  |
| --- | --- |
|  | Traffic sign classification  April Meyer  DSC680-T301 Applied Data Science (2215-1) |

# **Executive Summary**

This project demonstrates how to use a convolutional neural network to classify traffic signs. Cars have come a long way and require traffic sign detection. Convolutional neural networks or ConnvNets are a great way to classify images and thus can be used to classify traffic signs. The data used was preprocessed German traffic signs that were saved into the nine pickle files (6). The dataset has 43 types of signs, classes. Training subset contains 86989 images, validation subset contains 4410 images and testing subset contains 12630 images. The model was validated using accuracy. The results showed that the model performed well with a testing accuracy of more than 77%. All models used the same parameters, but different filter dimensions of 3 X 3, 5 X 5, 9 X 9, and 13 X 13. The classification accuracy equal to 89.6%, 82.9%, 80.1% and 77.8 %

# **Preliminary Analysis**

## **Background of the problem**

With the major advancements in self driving cars and smart cars has given way to a new classification problem for identifying traffic signs. The self-driving cars require the ability to know the current speed limit, stop signs, etc. Also, some car manufacturers can use a classification system to identify the traffic sign and notify the driver.

Convolutional neural networks or ConnvNets are a great way to classify images and thus can be used to classify traffic signs (3).

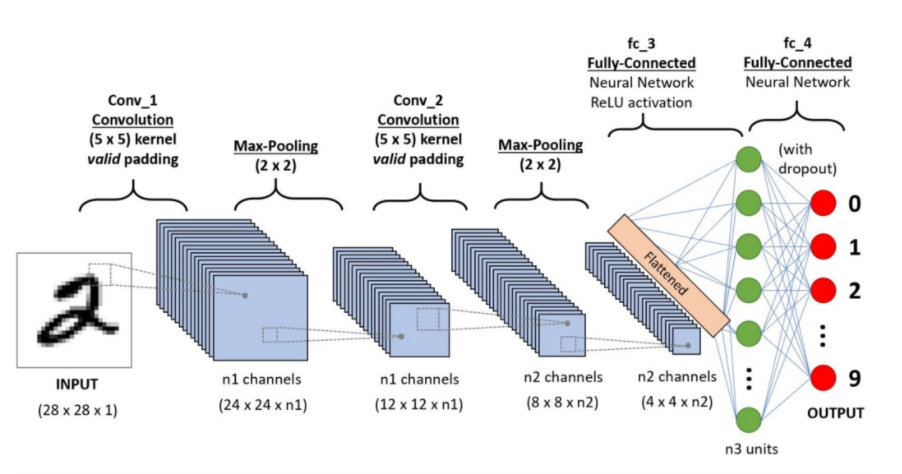
## **Problem Statement**

This project seeks to show how to build a convolutional neural network (CNN) identify traffic signs.

## **Methods**

Convolutional neural networks (CNN) which are sometimes known as convnets (1). It is a deep learning model that is predominantly used for computer vision (4). A ConvNet algorithm will take in an input image, give importance to facets in the image, and separate one from another (4). It will take in an image and process it in every convolutional layer where each layer has a set of trainable filters. The advantage of using ConvNets are that they require little pre-processing and are still able to understand the complexity of an image. The major goal is to reduce image to make it easier to process without losing key features (4).

Below is a sequence to classify handwritten digits:



## **Data Sources**

The link to the data is https://www.kaggle.com/valentynsichkar/traffic-signs-preprocessed. This is ready to use preprocessed data for Traffic Signs saved into the nine pickle files (6). It contains German Traffic Signs. Original datasets are in the following files, train.pickle, valid.pickle, test.pickle (6).

## **Data Exploration**

The dataset has 43 types of signs, classes. Training subset contains 86989 images, validation subset contains 4410 images and testing subset contains 12630 images. The images are 3-channeled RGB with a size resolution of 32 X 32 pixels.

y\_test: (12630,)

y\_validation: (4410, 43)

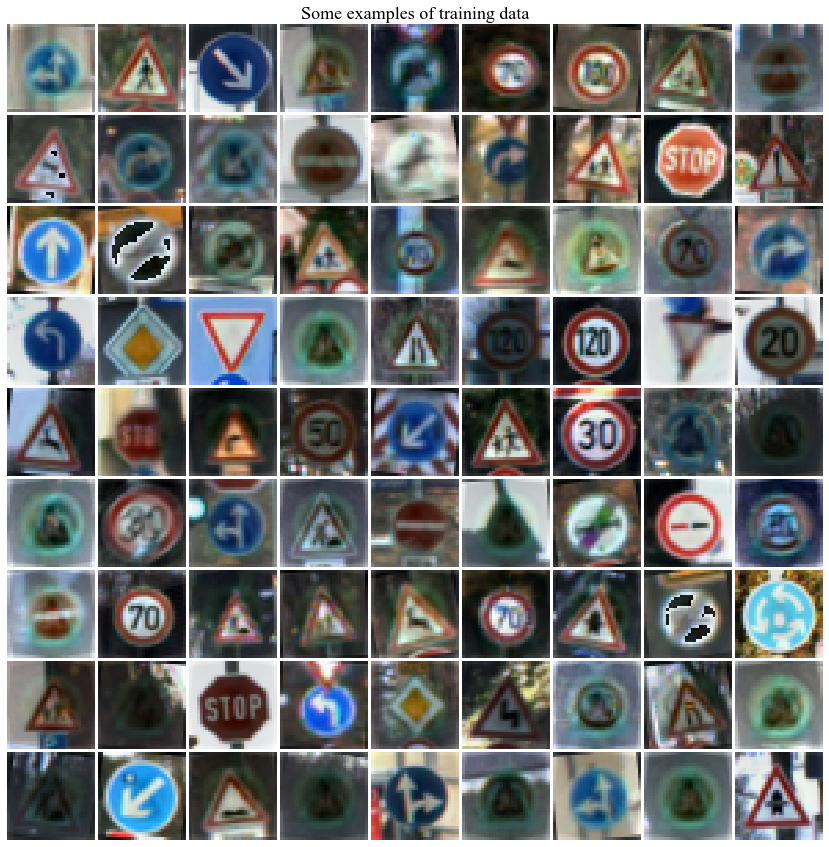
x\_validation: (4410, 32, 32, 3)

x\_train: (86989, 32, 32, 3)

y\_train: (86989, 43)

labels: 43

x\_test: (12630, 32, 32, 3)

****

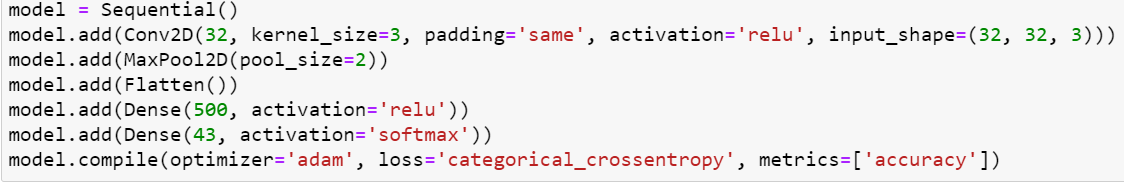
## **Data Preparation**

There was no data preparation since this dataset was pre-processed. One note for ConvNets is that they require the dimensions of all images to be the same. The dimensions for these images are 32 X 32 X 3.

## **Modeling**

After the preprocessed dataset is loaded from the pickle file, the RGB image is fed into the ConvNet layer that contains 32 filters. Every filter of the ConvNet layer will have 3 channels. There is a Rectified Linear Unit (relu) activation function that excludes negatives with zeros (3). After that layer, a pooling layer is applied that reduces the layer (3) then a hidden fully connected layer that is 500 neurons. The output layer has 43 neurons which aligns with the number of classes.

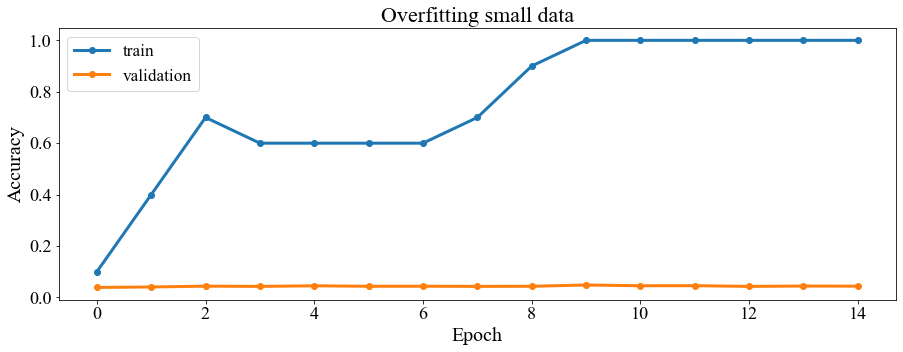
The compilation step has the optimizer, loss function, and the metric used to observe (3). The optimizer will decide how the network will be revised using the loss function (3). Adam is used as the optimizer. Instead of being a gradient descent it is an adaptive optimizer. The nice aspect is that they do not require tweaking of the learning rate. The loss function defines the feedback signal used for learning (3). Categorical Crossentropy was used since it is for multi-class classification tasks. This is where only one image can belong to one out of several possible categories, and the model determines which one (3).



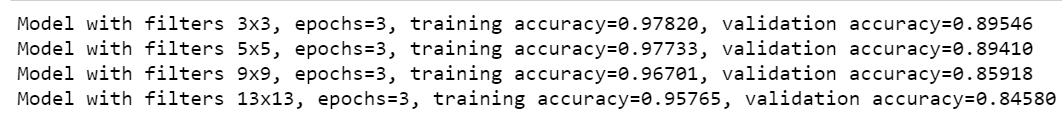
## **Results**

The model was validated using accuracy. My first model was built with a filter size of 3 X 3 and was overfitted. It was overfitted by only training on 10 samples and validating on 4410.

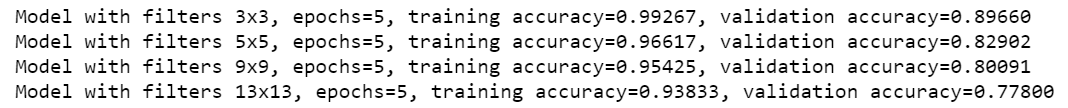
Epochs=15, training accuracy=1.00000, validation accuracy=0.04830

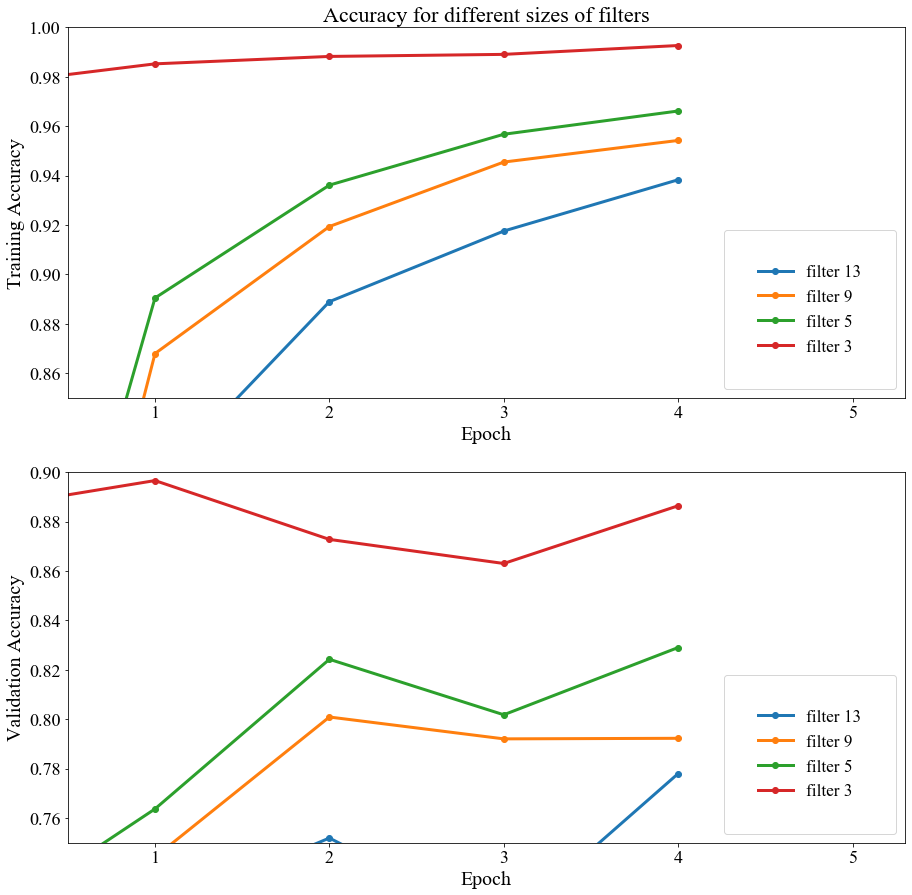


My next model used the whole training set (86989) with different filters but took a very long time to run. Below are those accuracy results.

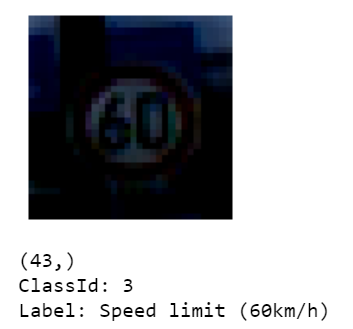


After this I scaled back to only using 12000 of the training set.





Lastly, I tested it by predicting one image from the test data.



## **Discussion/Conclusion**

Overall, this project proved that convolutional neural network can be used to classify traffic signs. The data used was preprocessed German traffic signs. The dataset has 43 types of signs, classes. The model was validated using accuracy. The first model was overfitted by using a small training set. The final model results showed that the model performed well with a testing accuracy of more than 77%. All models used the same parameters, but different filter dimensions of 3 X 3, 5 X 5, 9 X 9, and 13 X 13. The classification accuracy equal to 89.6%, 82.9%, 80.1% and 77.8 %

## **Acknowledgements**

I would like to acknowledge the Kaggle Repository for providing this dataset I used for the project.

# **References:**

1. Bansari, S. (2020, May 18). *Using a Convolutional Neural Network to Identify Traffic Signs*. Medium. https://medium.com/@simranbansari/using-a-convolutional-neural-network-to-identify-traffic-signs-c7da4d3a7cd7.
2. Brownlee, J. (2020, August 18). *Transfer Learning in Keras with Computer Vision Models*. Machine Learning Mastery. https://machinelearningmastery.com/how-to-use-transfer-learning-when-developing-convolutional-neural-network-models/.
3. CHOLLET, F. R. A. N. C. O. I. S. (2021). *Deep Learning With Python*. O'REILLY MEDIA.
4. Doshi, S. (2019, December 8). *Traffic Sign Detection using Convolutional Neural Network*. Medium. https://towardsdatascience.com/traffic-sign-detection-using-convolutional-neural-network-660fb32fe90e.
5. Saha, S. (2018, December 17). *A Comprehensive Guide to Convolutional Neural Networks - the ELI5 way*. Medium. https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.
6. Sichkar, V. (2019, August 31). *Traffic Signs Preprocessed*. Kaggle. https://www.kaggle.com/valentynsichkar/traffic-signs-preprocessed.