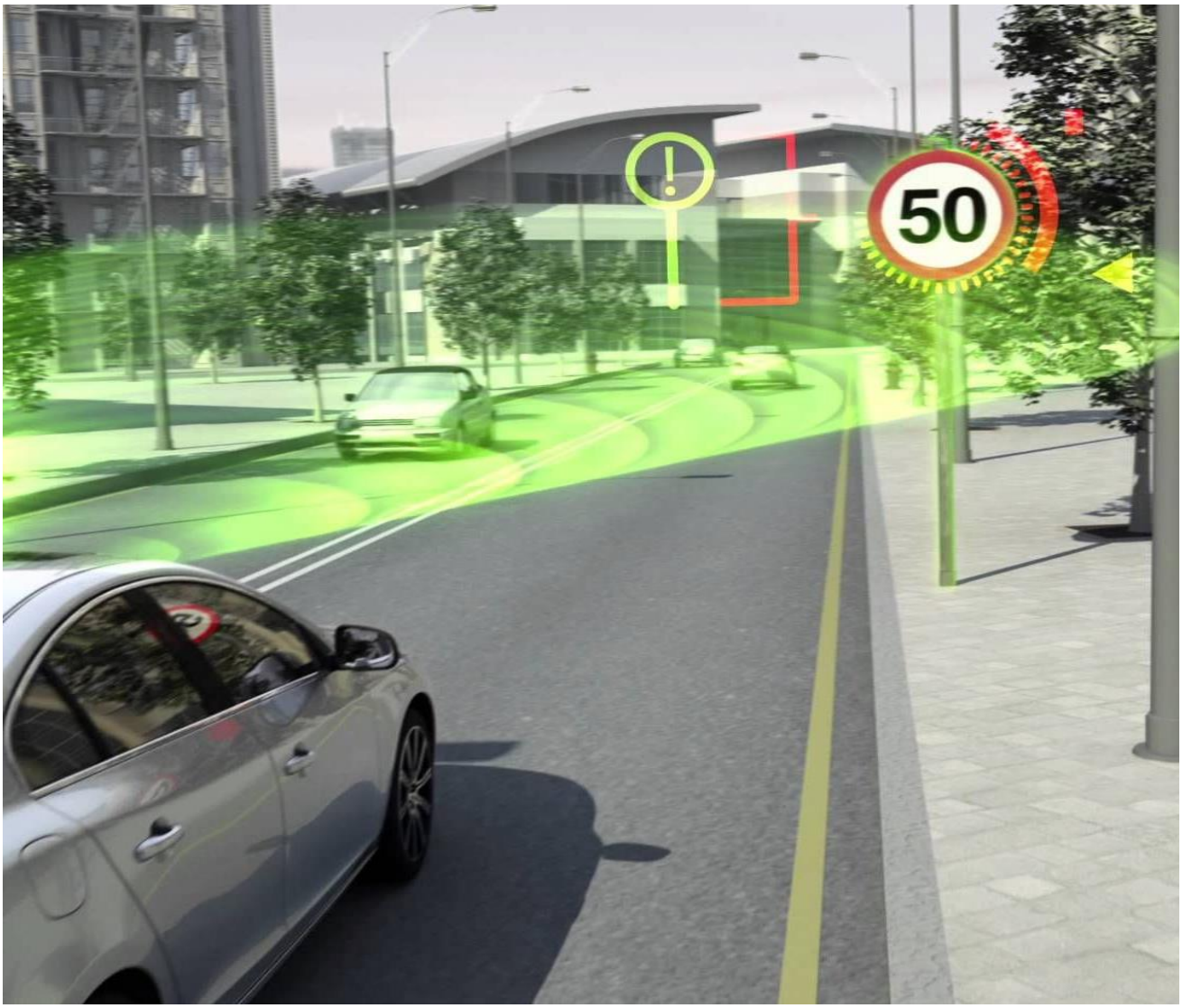

IMAGE CLASSFICATION ON TRAFFIC SIGN DETECTION

ISM 6251 FINAL PROJECT

Dr. Balaji Padmanabhan

GROUP 6:

Nynali Gopu

Venu Gopal Guddati

Lavanya S. Kumar

Adhithyan Rangarajan

Keerthana Yelchuri

Contents

Introduction

In recent years, machine learning models have been increasingly used to address large-scale applications in several industries including transportation, security, healthcare, marketing, and banking. The type of machine learning models adopted vary by the use case as well as the industry in which they are used. Artificial intelligence through machine learning algorithms is being viewed as the most efficient way to address the complex problems of an ever more technologically driven world.

With the advent of the driverless cars and the widespread everyday use of navigation systems all over the world, the importance and accuracy of detecting road traffic signs has achieved paramount importance in the field of perception and imaging. Automation has offered promised returns of improvements in safety, productivity and reduced costs. Many industry leaders are specifically working on the application of autonomous technology in transportation to produce "driverless" or fully autonomous vehicles. A key technology that has the potential to drive the future development of these vehicles is deep learning. Self-driving cars represent a high-stakes test of the powers of machine learning, as well as a test case for social learning in technology governance

Problem statement

Over the past 70 years, the percentage of urban population in the world has increased from around 30% in 1970 to 56% in 2020, and this number is projected to further increase to 68% by 2050 [4]. This growing urbanization across the globe presents unique challenges to the transportation industry leading to the development of autonomous driving cars. Machine learning solutions pave the way for the use of advanced technologies in self-driving cars and can not only enable safe and efficient navigation of driverless cars but also aid the growth of the automobile industry to contribute to the global economy. One of the most important problems in the use of driverless cars is the recognition of traffic signs and accurate vehicle navigation based on the detected road sign. To address this problem, we decided to apply deep learning algorithms to an existing German traffic sign dataset. The dataset, our analysis, and findings using different neural network models through the latest packages in Python programming language are described in the following sections.

Research Questions that frame the exercise

1.With the advent of the driverless cars which push the machine learning process to the highest limit, we wanted to dive deeper and work on this most important use case within its ecosystem to achieve best efficiency in predicting traffic signs accurately for flawless navigation. There have been cases where traffic sign detection has gone wrong and this can result in fatal outcomes, including death. One example of a recent failure is of a Tesla Auto-Pilot mistaking the moon for an orange traffic light and slowing down in the middle of the highway.

(https://www.independent.co.uk/tv/news/tesla-autopilot-system-mistakes-moon-for-traffic-light-slowing-car-down-v6fe32431)

2. Navigation systems are in widespread use today on a daily basis but most of them do not have the potential to detect the road signs and output instructions via speech. A model like this one can be integrated to work with a navigation system video feed which can greatly help in reducing accidents by ensuring minimal number of blind spots or driver mistakes.

Method

Dataset description

The dataset used in this project was publicly available online and downloaded for free from Kaggle. The data pertains to that used for the German Traffic Sign Recognition Benchmark (GTSRB) challenge held at the International Joint Conference on Neural Networks (IJCNN, 2011). The dataset comprised 39209 images in the training set and 12630 images in the test set, all in .png format. Each image consisted of a single traffic sign and belonged to one of 43 distinct classes (e.g., 20km/h speed limit, pedestrian crossing, slippery road). Appendix A shows the list of classes. Apart from the captured training and test images, three .csv files provided additional details on the images. The training.csv and test.csv files provided image size, class identification, and image folder path details. The third meta.csv file had information on the shape and colour of each of the 43 distinct classes. Shape corresponded to one of 5 categories: triangle, circle, diamond, hexagon, or inverse triangle and colour corresponded to one of 4 colours: red, blue, yellow, and white, all coded with distinct numerical labels. These details were used to decide on our levels of analysis.

We have 43 classes of road signs in the dataset and as shown in Figure 1 histogram represents the number of images for each class.

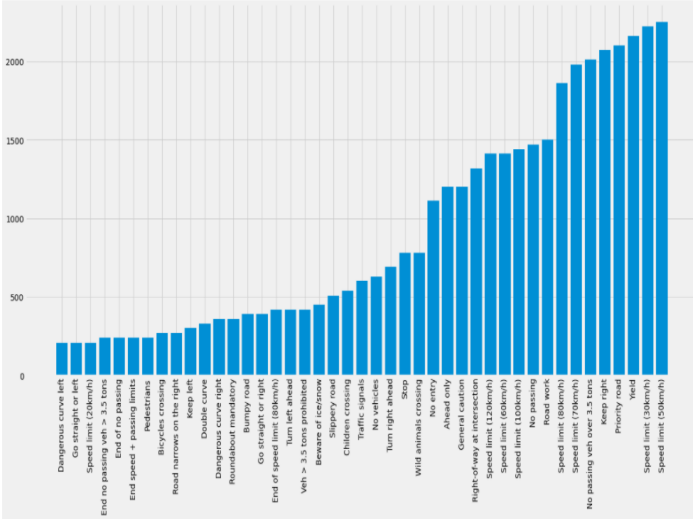

Figure 1.Number of images for each class

Version Goals

Based on gradually increasing complexity, we came up with the below multiple version goals for the scope of our work in this project and post the end of the semester.

- ➢ Version 1 - Work on leveraging a ML model to predict the road sign correctly using available top Python libraries on a small dataset size.
- ➢ Version 2 - Perform V1 on a messy dataset which contains images of all real-life scenarios i.e., images containing road signs which are blurred, pitch dark, rotated, translated, unclear etc. for adversarial ML.
- ➢ Version 3 - Perform V2 accurately on the complete dataset of 50000+ images with different dataset chunks and shuffle index for all 43 classes of the road signs and compare model metrics.
- ➢ Version 4 - Use V3 model and train it on live video to provide speech output and sensible instructions based on the road sign detected in the image frame to guide with navigation and driverless cars.

Within the scope of this project, we have successfully accomplished up to version 3.
Version 4 is an experimentation scope for us to leverage the knowledge gained in the DSP course and work deeper technically on this interesting use problem.

The python code for our project is not single sourced from any GitHub repository nor combined from multiple online references. The main source of the ML model code is from Canvas from one of Professor's notebooks shown in class which we have tweaked and experimented with based on the knowledge gained from papers and documentation.

Data cleaning and pre-processing

There were no missing data in the training.csv and test.csv files for any of the images in the dataset. For the training data, the folder itself contained 43 subfolders of images pertaining to each target class. Therefore, all the data points were used for analysis.

Trials were carried out with and without data pre-processing. Since the automobile cameras can capture traffic sign images at several angles and differing levels of clarity while in transit resulting in either rotated or blurred images, it was decided to look at the benefits of pre-processing images in the existing dataset. Resizing all the images to a standard 30 x 30 pixels was the only pre-processing change done. A sample view of the images in the dataset is as shown in the Figure 2.

Figure 2.Sample view of images in the dataset

As we can observe in the images shown in the output of the cell above, most of the different types of images are already available i.e., blurry images, dark images, rotated and translated ones along with some clear images. The reason the dataset covers all such possible cases is because these image frames were taken by using in-motion videos from the real-life driving video sample by using features such as burst mode to break the video into multiple frames/second to build this dataset. Leaving out resizing which was performed as mentioned in the earlier section, Data Augmentation and translation/rotation are not performed since the dataset already contains images for these scenarios (as we can see in the above cell output) because these images are taken from a video feed at varying speeds, distances and angles.

We are not experimenting with Translation and Rotation for the same reason mentioned above. Also, gaussian blurring has been skipped since the dataset already contains extremely blurry and dark images. Reflection/Flipping, Grayscale conversions have also been skipped since we need the coloured images to check the RGB values for predicting the type and label of the stop sign.

Histogram Equalization to tweak contrast levels and RGB Colour Inversion has also been skipped since the model needs to be able to identify in adversarial conditions and must not look for perfect stop sign image capture through video all the time as these kinds of giving the model only clean and processed images would make it perform worse on a real video feed.

Data Analysis

Data analysis was carried out through the Python programming language and different libraries such as TensorFlow, Keras, PIL, and OpenCV were used in different stages of analysis. NumPy, Pandas, Matplotlib, and Sci-kit learn were also used for general statistics, accuracy metrics, and data visualization.

The number of images pertaining to each class in the training set varied from 210-2250. Since all the images were of different pixel formats, the images in the training and test sets were resized to 30*30-pixel format. In addition, the OpenCV library reads images in BGR format, images were converted to the conventional RGB format before feeding into the algorithms for training and testing. The full training set was split into training (70%, 27446 images) and validation set (30%, 11763 images) used to train the deep learning models and model evaluation was carried out on the test set (12630 images).

As described earlier, all the levels of analysis were carried out through different types of deep learning models. The types of models used are specified in the below section.

Models

Stochastic Gradient Descent

Since on massive datasets SGD converges faster, we created a model using the SGD classifier. We trained the model using 70% of the data and tested it using the remaining 30% of the data. And, noted that the accuracy of the model is 70.3%. We want to try deep learning models (single perceptron, multi perceptron) and check if accuracy can be improved.

Single Perceptron and Multi Perceptron

We tried a single perceptron model since it's the simplest type of neural networks and can classify linearly separable cases. We created a single perceptron model with 70% of data as train data and 30% of data as train data and noted that the model accuracy is 90.4%.

We want to try a multi-perceptron model as it can have more hidden layers helping the model to better understand the data. We created the model and noted that accuracy is 97.2%.

Convolutional Neural Networks (CNN)

Further, we tried developing models using CNN architecture since it is widely used for image classification. CNNs use raw pixel data of an image, trains the model and then extracts the features which help in better image classification. In a CNN layer, a neuron in a layer is connected only to a subset of neurons in the previous layer. In CNN, each layer is a grid because data comes as a matrix. So, you take your input as a grid. Each of these convolutional layers has neurons distributed in a grid but every neuron has its own receptive field. This architecture lets higher level CNN layers let the model learn raw feature maps in gradual steps. Convolutional layers identify features.

We included Pooling as it reduces the amount of computation to be performed and sends only the required inputs to the next convolution layers. We used MaxPooling since it picks largest pixel values from the receptive field and can help in better image recognition. In addition, we added batch normalization as it accelerates the training process of a neural network and included activation function in the hidden layer as it controls how well the network model controls the training dataset. Also, dropout layer is added because some values will be randomly dropped during feed forward and this will help the model to reduce overfitting,

The process of training data using a neural network architecture was slow. So, we were using an optimizer as it helps in reducing the losses and provides more accurate results possible. We tried different optimizers when building a neural network apart from Gradient Descent - Momentum, Nesterov Accelerated Gradient, AdaGrad, RMSProp and Adam. We finally used Adam as it worked faster comparatively on our large dataset. We tried all the various permutations and combinations for activation functions, optimizers and hyperparameter tuning based on the knowledge gained from reading the Keras and TensorFlow documentation in our CNN model like increasing/ decreasing number of filters and number of dense layers. We were tuning hyperparameters as updating the learning rate is a process done to reach the minimum point. But this dataset is augmented and did not lead to improvements in efficiency.

While we did not search or use existing solutions specific to any traffic sign dataset, we used deep learning architectures that successfully classified other image datasets available online to guide our analysis [8]. Each of these models were trained with different architectures and optimizers before choosing the best models for each level of analysis. The architectures for these best models are described above and reflect the best accuracy as well as training efficiency. The code for each of these is provided in the zip folder.

Model Evaluation and Results

To evaluate each model, we studied the prediction accuracy, false positive rate, true positive rate, and confusion matrices. We also looked at the training versus test accuracy curves and the training versus test loss curves for increasing the number of training cycles or epochs. The below table shows the accuracy of the models we experimented with in the ascending order. We observed that the CNN model using Keras and TensorFlow gave us the best accuracy over the rest at 98%.

Table 1.Models and Accuracy Scores

MODELS	ACCURACY
SGD	70.4 %
Single Perceptron	90.4 %
Multi Perceptron	97.1 %
CNN	98%

The prediction output of the final CNN model is as shown in the below figure.

Figure 3.Actual vs Predicted label in the output

Business Recommendations and future scope

Our findings show that deep learning models, particularly CNN models can be used to train traffic sign datasets with high accuracy of prediction (greater than 99%). For predicting traffic sign class, CNN models were best with 99.7% accuracy on the test set. This indicates that our solution varies with each level of analysis. Also, because of the huge size of the dataset with blurred, slightly rotated, zoomed in and zoomed out images (perhaps already augmented dataset), data pre-processing and data augmentation did not lead to improvements in accuracy. Therefore, we recommend CNN architecture with batch normalization for traffic sign recognition. Data pre-processing may be more useful on smaller datasets and automatic data augmentation is a good feature to try out while training the model.

Based on the research questions mentioned previously, there are two main business questions that can be answered in the areas of driverless cars and navigation systems. Recommendations include the below-

1. To experiment with advanced machine learning models and image detection libraries to be integrated into the auto-pilot system in order to improve accuracy of the system to predict road signs.

2.Everyday navigation systems such as google maps do not provide guidance on detected road signs and preventive action, it only outputs speech for the directions or re-routing. A model like this can be integrated to the existing navigation system camera input to detect these road signs and provide sensible instructions to drivers especially when warning or dangerous signs are detected.

Future work in this area is to train the models on videos obtained while driving and later deploy the model to work on a live video to break down the video into fragments of images and accuracy can be checked on the trained algorithm to detect the road signs correctly. The next level from here would be to use the model to output speech using natural language to provide sensible navigation instructions to the driver or the driverless car system.

Conclusion

Our project presents one application of deep learning models to self-driving cars. Specifically, we looked at recognition of traffic signs that enable safe performance of these cars bringing value to the transportation industry. In order to do so, we used an existing dataset and applied three types of deep learning models to predict the outcomes. We discovered that CNN model was best at multi-class classification at the highest level, particularly with 43 target classes. The architecture we used allowed us to predict the outcome at an accuracy of 97% on a separate test set. The findings can further be used to predict outcomes on driving videos to emphasize further value as well as use the recommendations in the above section to improve these existing systems for overall safety and improvement in technology.

References

1. Persson, Siri. "Application of the German Traffic Sign Recognition Benchmark on the VGG16 network using transfer learning and bottleneck features in Keras." (2018). https://www.diva-portal.org/smash/get/diva2:1188243/FULLTEXT02.pdf
2. R. Qian, Y. Yue, F. Coenen and B. Zhang, "Traffic sign recognition with convolutional neural network based on max pooling positions," 2016 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD), 2016, pp. 578-582, doi: 10.1109/FSKD.2016.7603237.

3. Andrew Campbell, Alan Both, Qian (Chayn) Sun,Detecting and mapping traffic signs from Google Street View images using deep learning and GIS,Computers, Environment and Urban Systems,https://doi.org/10.1016/j.compenvurbsys.2019.101350 https://www.sciencedirect.com/science/article/pii/S0198971519300870
4. https://towardsdatascience.com/identifying-traffic-signs-with-deep-learning-5151eece09cb
5. https://www.kaggle.com/meowmeowmeowmeowmeow/gtsrb-german-traffic-sign
6. https://www.un.org/development/desa/en/news/population/2018-revision-of-world-urbanization-prospects.html
7. https://iq.opengenus.org/basics-of-machine-learning-image-classification-techniques/
8. https://www.linkedin.com/pulse/classify-traffic-signs-deep-learning-james-medel/

Appendix A

List of target classes for the German Traffic Sign Recognition Benchmark (GTSRB)

Label Overview

classes =

0:'Speed limit (20km/h)',

1:'Speed limit (30km/h)',

2:'Speed limit (50km/h)',

3:'Speed limit (60km/h)',

4:'Speed limit (70km/h)',

5:'Speed limit (80km/h)',

6:'End of speed limit (80km/h)',

7:'Speed limit (100km/h)',

8:'Speed limit (120km/h)',

9:'No passing',

10:'No passing veh over 3.5 tons',

11:'Right-of-way at intersection',

12:'Priority road',

13:'Yield',

14:'Stop',

15:'No vehicles',

16:'Veh > 3.5 tons prohibited',

17:'No entry',

18:'General caution',

19:'Dangerous curve left',

20:'Dangerous curve right',

21:'Double curve',

22:'Bumpy road',

23:'Slippery road',

24:'Road narrows on the right',

25:'Road work',

26:'Traffic signals',

27:'Pedestrians',

28:'Children crossing',

29:'Bicycles crossing',

30:'Beware of ice/snow',

31:'Wild animals crossing',

32:'End speed + passing limits',

33:'Turn right ahead',

34:'Turn left ahead',

35:'Ahead only',

36:'Go straight or right',

37:'Go straight or left',

38:'Keep right',

39:'Keep left',

40:'Roundabout mandatory',

41:'End of no passing',

42:'End no passing veh > 3.5 tons'