Journal of Research High School

Vol. 2023 (2) xxx – xxx
ISSN 2688-3651 [online]

Traffic Sign Detection and Recognition with Deep Learning, CNNs, YOLOv3, and Keras

Aniket Gupta[1 *]

[1]Silver Creek High School, San Jose, CA, USA
*Corresponding Author: anigupta102@gmail.com

Advisor: Prof. Susan Fox, fox@macalester.edu

Received February 25, 2023; Revised July 22, 2023; Accepted, August 28, 2023

Abstract

The growing industry of transportation being integrated with Artificial Intelligence has contributed to applications regarding driving assistance and autopilot software. With this application being relatively new, more research still needs to be done to improve these algorithms' recognition/classification of traffic signs. In order to improve modern driving softwares, I proposed designs for a basis for an object detection model and a functioning neural network classification model to be used as an end-to-end solution. The object detection model serves as a locator of a traffic sign in an image to be inputted into the classification model to determine the identity of the sign. The detection model was designed using Keras and YoloV3. During the development phase for the classification model, I used a previously made model as a basis and used image processing techniques of deep learning, OpenCV, and the multi-layers of convolutional neural networks to modify it. Subsequently, I used the German Traffic Sign Recognition Benchmark (GTSRB) for training which contains over 50,000 images of German traffic signs for training and validation. The classification model resulted in 99.71% training accuracy and 98.7% - 99.21% validation accuracy. Furthering, this study discussed each of the convolutional layers used for the classification model and how it was changed from the original model.

Keywords: Image Processing, Deep Learning, OpenCV, Convolutional Neural Networks, Classification, Object Detection, Keras, YoloV3

1. Introduction

The transportation industry has advanced immensely within the last few decades, especially in automobiles. Recently, car manufacturers have begun integrating AI for driving assistance and even autopilot. One aspect of this situation is figuring out how to make autopilot and driving assistance safer for drivers and passengers.

For autopilot, one skill the AI/car needs is to identify street signs to evaluate its surroundings and abide by traffic laws for the passenger's safety. Driving assistance must also properly detect, identify, and classify signs to account for the millions of car accidents due to driving errors. Much attention has been brought to Traffic Sign Recognition as there is much research and work on this problem. Traffic sign recognition involves two main parts first, detecting the sign from a video feed or still image, and second identifying what sign it is. Detecting a sign uses image processing and deep learning object detection architectures. Classifying the type of sign uses a deep learning CNN classification model.

In recent years, deep learning has been the “go-to” over machine learning for tackling this problem as it requires less human intervention during training. A sub section of deep learning, Convolutional Neural Networks (CNN), based on computer vision (or OpenCV) with other models, has been used to efficiently analyze street view images and pick out unique features that can be used to identify signs. However, this has been proven difficult as many unaccounted

Vol. 2023 (2) xxx – xxx
ISSN 2688-3651 [online]

factors interfere with the identification process, such as shadows, light reflection, the sign being partially covered, the quality of the sign being poor from the camera, etc.

Thus to tackle this problem and to improve driving softwares, the objective of this research project is to produce a system that can efficiently detect and classify traffic signs, with both easy and rough environmental conditions, to be applied in real time applications. With sufficient identification and classification of traffic signs, the world of driving could become significantly safer as computers and driving algorithms can gain more information regarding the vehicle's current surroundings.

Furthermore, this two-part system involves object detection and classification programs which I created by improving a CNN model that has been proven effective on one dataset and use it for another dataset, the GTSRB dataset, which stands for German Traffic Sign Recognition Benchmark dataset. This famously known dataset contains 43 different classification classes and more than 50,000 images for training and testing, making it a big enough dataset for me to train a coherent classification model.

To produce this project, TensorFlow and Keras were used for the models. "TensorFlow is a library containing multiple machine learning tasks and APIs such as Keras" (Terra 2022). Keras is a deep learning API developed by Google and runs on TensorFlow and is highly used because it is built with Python.

1.1 Deep Learning

Deep learning is related to artificial intelligence. It utilizes big chunks of data with complex algorithms to create artificial neural networks to train computers and machines to make data-based decisions (happiest minds, 2022). Feature extraction is a crucial aspect of deep learning. Feature extraction is where features, patterns, and properties are extracted from inputted data given to the neural network model. This is also how image recognition is done. Kondamari, Itha (2021) defines image recognition as where the model extracts details and patterns from small sections of the image using the pixels, which depend on other surrounding pixels. These models have been proven to be great substitutes for complex codes/programs for extracting features.

1.2 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks are deep learning algorithms trained to take in an input image, assign importance by adding learnable weights and biases to various objects in the image, and be able to differentiate one from the other (Saha, 2018). The goal of CNNs is to shape or reduce the image into a form that is easier to process without losing the necessary features to extract for a prediction in image recognition or classification. These networks contain many layers/filters, each summing up to allow the network to detect spatial relationships to allow the classification of an object. Spatial relationships are where one thing depends on another, such as pixels in an image correlating/depending on its surrounding pixels and vice versa.

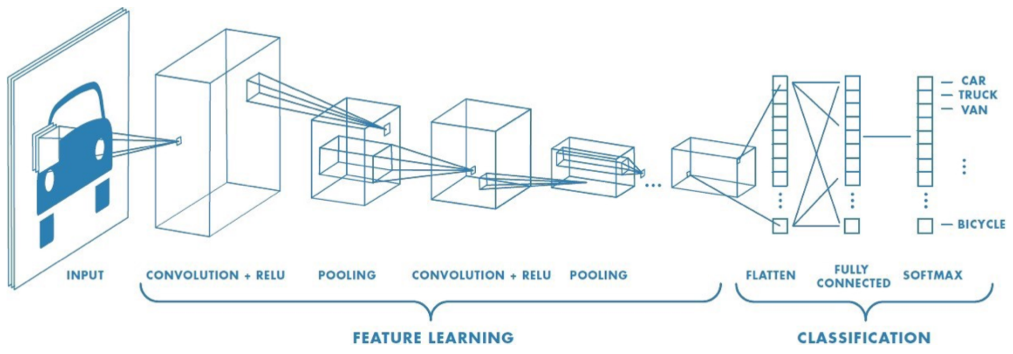


Figure 1. A diagram of a CNN with all usual its layers and classification portion (Saha, 2018).

CNN models have an input layer that takes in an image and an output layer at the end, where the output is a reduced vector of probability scores. As seen in figure 1, in between are the many different layers in a CNN network: convolutional, max pooling, flatten, dense, fully connected layer, and softmax. The first couple of layers is for the model to learn features, while the second portion is to classify the features detected.

The first layer is the convolutional layer which captures low-level features (colors, gradient orientation, etc.) and uses the convolutional mathematical operation to extract high-level features (such as edges from the input image) and uses these to train its layers and adapt to allow the network to perceive the image as humans would (Saha 2018). The convolution operation is expressed as:

$$s(t) = (x * w)(t)$$

where x is the input data or the pixels from the input image, and w is the kernel applied to these pixels (Kondamari & Itha, 2021). A Kernel is a matrix with smaller dimensions than the image and filter. This operation slides this filter over the input until every location is covered by the kernel at least once and is multiplied by its respective values on the input and then summed up (Dertat, 2017). This sum is then placed in the feature map of the same size as the input (Figure 2), which becomes the output. This output is then sent as an input to the next layer and so on.

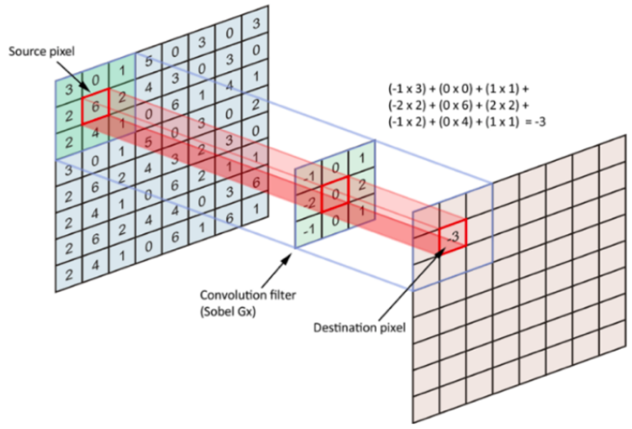

Source pixel
(-1 x 3) + (0 x 0) + (1 x 1) +
(-2 x 2) + (0 x 6) + (2 x 2) +
(-1 x 2) + (0 x 4) + (1 x 1) = -3
Convolution filter
(Sobel Gx)
Destination pixel

Figure 2. The process of how the operation is used with the kernel to produce a feature map (Cornelisse 2018).

The Pooling layer reduces the dimensions/size of the feature map summarizing the features presented in the feature map, so the amount of computation is decreased, making the data much more efficient to process (Saha 2018). There are two types of pooling, Max Pooling, and Average Pooling. Max Pooling returns the maximum values from the image region the kernel is covering as it slides over, while Average Pooling returns the average values.

Max Pooling Noise Suppressant where it reduces the noise (unnecessary features usually in the background that are picked up) as the dimensions are reduced. After the pooling, convolutional and pooling layers may be repeated as many times as necessary for the model.

The next layers are the Fully-Connected layers which make up the last few layers of the network for classification. The output from the final layer before the Fully-Connected (Conv, MaxPool, AvgPool) is a three-dimensional matrix that is flattened into a one-dimension vector to be connected to the Fully-Connected layers (Arc, 2018). After the model runs through a series of epochs (the number of passes through the dataset the learning algorithm has completed), it can differentiate between dominating and low-level features with the Softmax Classification technique. The model outputs a vector with zeros and ones for each classification class from the dataset where a “1” is the identified class.

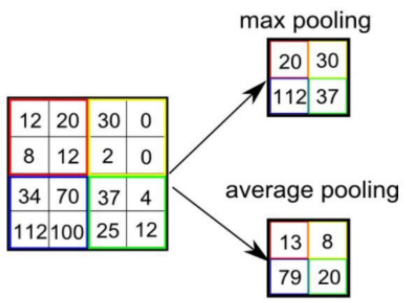


Figure 3. The process of how max pooling and average pooling works is shown. Max pooling takes the largest value in each kernel, while average pooling takes the average.

1.3 Object Detection with YOLOv3 and Keras

YOLO stands for “You Only Learn Once.” YOLO is a real-time object detection algorithm or CNN that identifies specific objects in videos or images by providing class probabilities of the detected images (Meel, 2019 & Anirudh, 2020 & Karimi, 2021). This means it will provide the likelihood of each object detected being a certain object. The YOLO algorithm uses a DarkNet framework. The number of layers depends on the version but typically consists of 3

Vol. 2023 (2) xxx – xxx
ISSN 2688-3651 [online]

x 3 or 1 x 1 filters, convolutional and residual (Genç, 2019 & Karovalia, 2021). A neural network has data flowing through each layer consecutively, where the output of one layer is the input for the next. A residual layer is when a residual connection provides an alternate path to reach the later parts of the neural network by skipping layers in between (Wong, 2020).

First, an image is split into grid cells with equal dimensions smaller than the image's dimensions to analyze the image piece by piece. Next, YOLO uses the concept of anchor boxes to detect multiple objects. In the model, the parameter for the number of anchor boxes and their respective shapes would be inputted, and each object lying in a close neighborhood will be detected and assigned to a box. More anchor boxes, more objects that can be detected and processed. The object detected in the image would be bounded with a bounding box as an outline consisting of its width, height, class (general categories like person, car, traffic light, building, etc.), and the center of the bounding box.

During this process, Non-max Suppression (NMS) ensures an object is only detected once rather than multiple times when the object lies across multiple grid pixels in the image. NMS works by discarding cells whose probability of the object being present calculated in the final softmax output layer is less than or equal to 60%, then will take the cell with the largest probability among the remaining possible candidates and select it as its prediction. Finally, NMS will discard any leftover cells with an intersection-over Union (IOU) value greater than 60% (Natashia, 2020). Karimi (2021) defines IOUs as a phenomenon in object detection which describes how these boxes and grid cells overlap. The YOLO model would then use this IOU to output a box surrounding the object and its label. This is the predicted box. Karimi also stated that YOLO would ensure the bounding box and the prediction box is the same as the IOU will equal 1.

1.4 Related Works

Latif et al. (2020) developed a recognition software for traffic signs (in this case, Arabic traffic signs) to aid drivers in reducing vehicle accidents on the streets and help with self-driving cars or some recognition system. Their project used Deep CNN to develop their system, which first detects the sign and then classifies or recognizes it. They collected 2718 images to form their Saudi Arabian Traffic and Road Signs database. Before Latif et al. (2020) published their paper, they had tested their initial CNN network, and their improved one showed 100% accuracy for all batch sizes tested on their developed dataset. With their CNN model already showing promising results, it would be useful to implement it in my research.

Saha, Islam et al. (2018) developed a unique traffic recognition software using not one but three CNN models. Two models were sub-models meant to classify signs, but the main model was trained to decide which of the two models to use to classify the signs using data augmentation. This is a unique approach resulting in accuracy results of close to 99%. However, it would be more efficient to use one or maybe two models in total than three or more. They also used the GTSDB dataset to train their models in this research project.

Pramod Sai Kondamari and Anudeep Itha (2021) discussed Traffic Sign Recognition (TSR) for drivers and self-driving cars. The author's techniques used Deep Learning CNN and OpenCV. The dataset they used was the GTSRB dataset. They discussed methods used during experimentation of how their model worked and layers used(four convolutional layers, two max-pooling layers, and two dense layers) (Kondamari & Itha, 2021). Their model gave results of 95% model accuracy with 30 epochs. Their paper was a great reference to me as they heavily explained each concept (deep learning, CNNs, CNN layers, etc.) involved in their project.

An object detection model was produced by Valentyn Sichkar & Sergey A. Kolyubin (2020). They discussed in their paper where the GTSDB dataset was used to train their first detection model, which would detect and separate the signs into four categories. The GTSDB dataset was split into training and validation sections with 85% and 15% proportions, respectively, with 630 and 111 images. Once detected, the located fragment of the sign would be cut and inputted into the second classification model to be classified and outputted with its coordinates, bounding boxes, and labels.

Vol. 2023 (2) xxx – xxx
ISSN 2688-3651 [online]

2 Materials and Methods

My project successfully implemented the classification network, and I designed an object detection network for the same dataset with YOLOv3 to predict the location of the signs (localization) from video frames and images.

The overall structure would work like this. First, the image would go through preprocessing, resized to 30 by 30, and not converted to grayscale. Subsequently, the image would go through the object detection model to detect the sign contained in the image. The next part would be to lower the region of interest (ROI) to the area of the detected sign and then input that region into the classification model to identify which sign it is. Finally, the model would return information on the sign to the autopilot algorithm.

2.1 Classification Model

I used the GTSRB dataset to train the classification model, which contains over 50,000 RGB images. For preprocessing, I resized each image in the dataset to 30 x 30 and stored them in a NumPy array, and the labels for each sign in another array. Afterwards, I split the images with 80% for training and 20% for validation.

Furthering, I built the model as a sequential model. There are 3 convolutional layers, 3 max-pooling, 1 flattening, and 2 dense layers, as seen in Figure 4. All convolutional layers used the relu activation, and the kernel initializer was set to uniform, and the padding set to same. The first portion of layers serve to provide feature learning for the model. The first layer of the portion is a 2D convolutional layer with 32 filters and 3,3 kernel size, and an input shape of (30, 30, 3). The second one is a 2D max-pooling layer with a pool size (2, 2). Following is another convolutional layer with 64 filters and a 2D max pooling with (2, 2) pool size. Lastly, there is one convolutional layer and max-pooling with 128 filters and (2, 2) pool size. After feature learning, the next set of layers serve as classification. We first transform the pooled feature map into a one-dimensional vector with flattening. Then we have two dense layers with the first activation set to relu and the second dense is an output layer with its activation set to softmax. The Adam optimizer was used for training to handle sparse gradients on “noisy” problems. A final summary of the classification model can be seen in Table 1.

Table 1. Summary of the complete final Classification model

Layer	Output Shape
conv2d (Conv2D) (input shape (30, 30, 3)) RGB image (32 layers)	(None, 30, 30, 2)
max_pooling2d (MaxPooling2D) (2, 2)	(None, 15, 15, 32)
conv2d_1 (Conv2D) (64 layers)	(None, 15, 15, 64)
max_pooling2d_1 (MaxPooling2D) (2, 2)	(None, 7, 7, 64)
conv2d_2(Conv2D) (128 layers)	(None, 7, 7, 128)
max_pooling2d_2 (MaxPooling2D) (2, 2)	(None, 3, 3, 128)
flatten (Flatten)	(None, 1152)
dense (Dense)	(None, 128)
dense_1 (Dense) (Softmax)	(None, 43)

2.2 Object Detection Model

For sign detection, although more research needs to be completed, a pre-trained YOLO network was modeled. The model used the DarkNet code base and GTSDB dataset (German Traffic Sign Detection Benchmark). 106 2D convolutional block layers were loaded, and the model was defined and loaded with the provided model weights. Once the model was loaded with its weights, the model would be saved to use for a prediction. We would require an input

Vol. 2023 (2) xxx – xxx
ISSN 2688-3651 [online]

image, in this case, a street-view image of a traffic sign (or a video frame from a video feed) and have the model predict the location of the sign.

3 Results

As stated before, the classification network produced in this research project was based on the neural network provided by Latif et al. in their research paper (Table 2).

Their model gave 97% accuracy from training their model with the Arabic dataset. Although when I was training the model with the GTSRB dataset, the accuracy results were minimal. After attempting to train over 50 epochs, I observed no improvements in training and validation accuracy and training and validation loss remained constant at 3.49% and 3.48% respectively.

For the final CNN model, I removed the two dropout layers from the original model and also followed a convolutional, max pooling layer pattern to efficiently process the input data. There are three conv2d layers, each has max-pooling right after. Following these changes, the model showed significant improvement as it stopped underfitting and could learn better without the dropout layers shown in Figure 5. The model showed about 99.71% training accuracy and 98.7% - 99.21% validation accuracy. However, the validation loss was minimally increasing over time with each epoch but stayed low enough for sufficient accuracy results.

Table 2. Summary of the original classification model (Latif et al. 2020).

Layer	Output Shape
conv2d (Conv2D) (input shape (30, 30)	(None, 28, 28, 64)
max_pooling2d_1 (MaxPooling2)	(None, 14, 14, 64)
dropout_1 (Dropout)	20%
max_pooling2d_1 (MaxPooling2D)	(None, 12, 12, 32)
max_pooling2d_2 (MaxPooling2D)	(None, 6, 6, 32)
flatten_1 (Flatten)	(None, 1152)
dense_1 (Dense)	(None, 128)
dropout_2 (Dropout)	(None, 128)
dense_2 (Dense)	(None, 32)

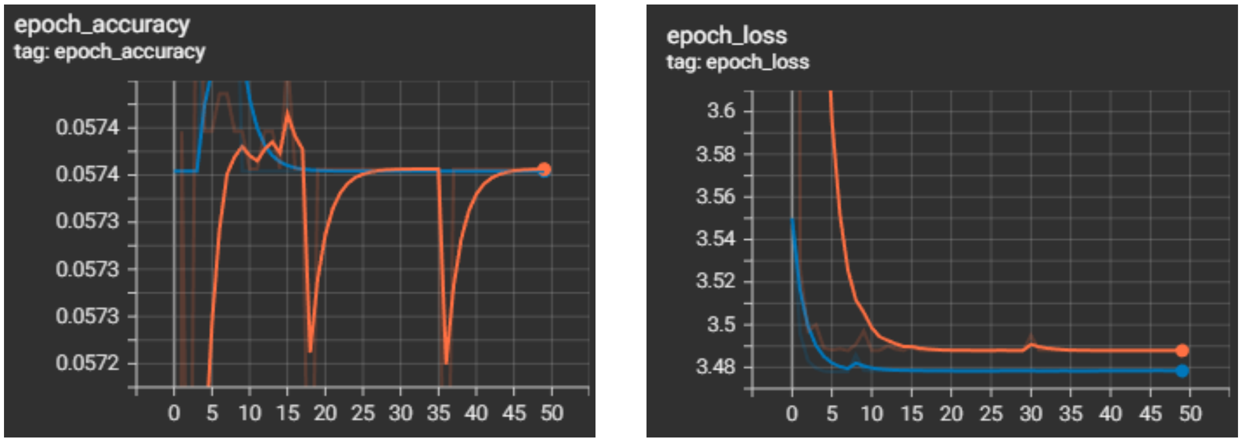
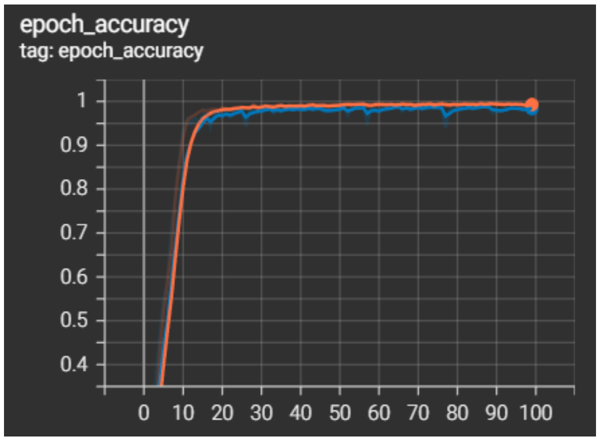


Figure 4. The overall accuracy and loss over 50 epochs of the original CNN model. Orange is the training data and blue is the validation data.

For object detection, I tested the model with other generic pictures that should work (a zebra, multiple zebras, planes, etc.), and the model was properly able to detect them. This confirmed the model was working. However, in the MSCOCO dataset, I discovered that it never included images of signs for the model to detect. As such, when inputted an image of a German street sign, the model could not recognize it. Further research will need to be done to see if this model can efficiently run and detect signs when trained on the GTSDB dataset.



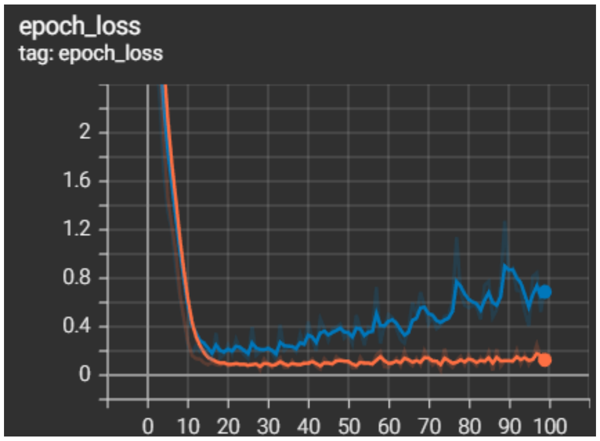


Figure 5. the accuracy and loss data for both training and accuracy across 100 epochs for the final CNN classification model. Orange is the training data, and blue is the validation data.

4 Discussion

4.1 Improving Original Model

When using the original classification model developed by Latif et al., the model showed no improvements in accuracy or loss. The reason for this was perhaps because this model was developed alongside a significantly smaller dataset. Another possibility was the two dropout layers (Table 2) took out too much data. Instead of causing "generalizing" to prevent the model from memorizing, too much data was dropped, and the model could not learn and train effectively.

To improve results, I removed the two dropout layers as too much data was being taken out to work with the current dataset. Another thing I had to account for was the lack of convolutional layers in the original model, as there is only one 2D convolutional layer. More CNN layers were needed so more elementwise multiplication could be conducted on the 2D input data to have better feature maps for the output/production. I also increased the output shape of the first layer as well as the max pooling layers to leave more information. After implementing these changes, the results significantly improved (Figure 5). Although, with each additional epoch, the training loss stayed relatively constant, the validation loss started to increase. The reason for this was perhaps because I didn't convert the images to grayscale (making the images gray) during preprocessing. This would've reduced some of the background noise in each image, preventing the validation loss from increasing and increasing accuracy results.

4.2 Future Work

This research project aims to integrate these detection and classification models into built-in vehicle systems to improve driving assistance or autopilot software. Too many deaths occur due to driving errors; if the car or vehicle could recognize signs, help manage the speed, and be aware of how dangerous the surroundings are, many lives could be saved. For future work, my first step would be to get the sign detection model working and add improvements based on performance. To start out with, the object detection model can be attempted to be trained by the GTSDB dataset and add improvements depending on results. Next would be to see where minimal improvements to the CNN model could be added. Although the model produces promising results, even a 1-2% increase in accuracy to close to 100% will be more beneficial for driving assistance and autopilot software, perhaps by converting the images to grayscale to reduce noise. After, I would then prepare a program to use both models to detect the sign, lower the ROI to the region of the sign(s), and input it into the classification model to classify. There, I would have it return information on the sign such as its shape, label of sign, the image of the sign, etc.

5 Conclusion

In this paper, I presented a working classification CNN model as well as a proposed plan for an object detection CNN model (and presented methods that worked in the past) to detect and classify signs.

The CNN classification model proposed by Latif et al. didn't work for the GTSRB due to the dropout layers. Furthermore, the classification model was improved by adding more convolutional layers with max-pooling layers and removing the dropout layers. The model was modified to work with RGB images and trained on the GTSRB dataset to achieve increased accuracy in performance.

Object detection proved to be a little tricky in the given time as detecting signs can be difficult as we can't solely rely on finding triangular, rectangular, or circular shapes to find these signs. Signs in images or videos can partly be covered by an object, or the lighting may interfere wildly if the light bounces back from the sign. Using Keras, YOLOv3, DarkNet, and a dataset (such as GTSDB) can be used to train a sign detection model.

From this research, it is well enough possible for computers to detect and classify signs in real-time with minimal errors successfully. AI for self-driving or driving assistance can majorly be improved with it being able to properly assess the potential dangers of the surroundings and follow traffic laws simply by correctly identifying signs.

Acknowledgment

I would like to thank my mentor, Professor Fox for the guidance of using OpenCV, deep learning, CNN's, and object detection. I would also like to thank her for her guidance throughout the research process when brainstorming, researching, and model building and for editing this paper.

References

Latif, G. et al. (2019). Autonomous traffic sign (ATSR) detection and recognition using Deep CNN. Procedia Computer Science, 163, 266–274.

Alpaydin, E. (2021). Machine learning. Amazon. Retrieved July 3, 2022.

Anirudh. (2020, July 9). German traffic signs detection using yolov3. Medium. Retrieved July 3, 2022.

Arc. (2018, December 25). Convolutional Neural Network. Medium. Retrieved July 7, 2022.

Bouguezzi, S. (2020, November 20). GTSDB - German Traffic Sign Detection Benchmark. Kaggle. Retrieved July 7, 2022.

Brownlee, J. (2017, July 3). Gentle introduction to the adam optimization algorithm for deep learning. Machine Learning Mastery. Retrieved July 7, 2022.

Brownlee, J. (2019, October 7). How to perform object detection with yolov3 in Keras. Machine Learning Mastery. Retrieved July 3, 2022.

Chollet, F. (2017). Save and Load Models : Tensorflow Core. TensorFlow. Retrieved July 3, 2022. Updated: 6/16/2022

Convolutional Neural Networks (CNN) with Deep Learning. HappiestMinds. (2020, January 28). Retrieved July 3, 2022.

Dertat, A. (2017, November 13). Applied deep learning - part 4: Convolutional Neural Networks. Medium. Retrieved July 3, 2022.

Doval, G. et al. (2019). Traffic sign detection and 3D localization via deep convolutional neural networks and Stereo Vision. 2019 IEEE Intelligent Transportation Systems Conference (ITSC

Forson, E. (2022, March 13). Recognising traffic signs with 98% accuracy using Deep Learning. Medium. Retrieved July 3, 2022.

freeCodeCamp.org. (2018, April 24). An intuitive guide to Convolutional Neural Networks. freeCodeCamp.org. Retrieved July 3, 2022.

Genç, Ö. (2019, January 4). Hands on machine learning example. Medium. Retrieved July 3, 2022.

Jain, S. (2021, July 29). CNN: Introduction to pooling layer. GeeksforGeeks. Retrieved July 7, 2022.

Karimi, G. (2021, April 15). Introduction to yolo algorithm for object detection. Section. Retrieved July 3, 2022.

Karovalia, A. (2021, April 25). Traffic signs detection using tensorflow and YOLOV3/ yolov4. Medium. Retrieved July 3, 2022.

Khanvt. (2022, June 6). Traffic signs classification. Kaggle. Retrieved July 3, 2022.

Kondamari, P. S., & Itha, A. (2021). A Deep Learning Application for Traffic Sign Classification. Faculty of Computing at Blekinge Institute of Technology, i-40. Supervisor: Shahrooz Abghari, PhD

McCullum, N. (n.d.). The flattening and full connection steps of Convolutional Neural Networks. Nick McCullum Headshot. Retrieved July 3, 2022.

Meel, V. (2022, April 19). Yolov3: Real-time object detection algorithm (what's new?). viso.ai. Retrieved July 3, 2022.

Mykola. (2018, November 25). GTSRB - German Traffic Sign Recognition Benchmark. Kaggle. Retrieved July 3, 2022.

Natashia, N. (2020, June 15). Yolo V3 object detection with keras. Medium. Retrieved July 3, 2022.

Rivas, P. (2020). Deep learning for beginners: A beginner's guide to getting up and running with deep learning from scratch using Python. Packt Publishing Ltd.

Rosebrock, A. (2020, July 13). R-CNN object detection with Keras, tensorflow, and Deep Learning. PyImageSearch. Retrieved July 3, 2022.

Saha, S. (2018, December 17). A comprehensive guide to Convolutional Neural Networks-the eli5 way. Medium. Retrieved July 3, 2022.

Saha, S. et al. (2018). An efficient traffic sign recognition approach using a novel deep neural network selection architecture. Advances in Intelligent Systems and Computing, 849–862.

Sichkar, V. (2021). Traffic signs detection by Yolo V3, opencv, keras. Kaggle. Retrieved July 3, 2022.

Simplilearn. (2021, September 18). What is Keras and why it so popular in 2021: Simplilearn. Simplilearn.com. Retrieved July 3, 2022.

Stewart, M. (2020, July 29). Simple introduction to Convolutional Neural Networks. Medium. Retrieved July 3, 2022.

Świeżewski, J. (2020, May 22). Yolo algorithm and Yolo Object Detection. Machine Learning. Retrieved July 7, 2022

Team, K. (n.d.). Keras Documentation: MaxPooling2D layer. Keras. Retrieved July 3, 2022.

Terra, J. (2022, May 27). Keras vs tensorflow vs pytorch [updated]: Deep learning frameworks: Simplilearn. Simplilearn.com. Retrieved July 3, 2022.

Wong, W. (2021, December 18). What is residual connection? Medium. Retrieved July 7, 2022.