Drive Assistance Research Project

Carl Christopher Cortez cccortez@syr.edu

Teddie Davis tdavis08@syr.edu

Kotaro Ueoka kueoka@syr.edu

Slides found here

College of Engineering and Computer Science, Syracuse University
900 South Crouse Ave.
Syracuse, NY 13244

Abstract

Drivers around the world are becoming distracted, leading to a rise in accidents and road fatalities. In an effort to keep the roads safe, car companies are actively adding drive assistance to their vehicles [6]. With current drive assistance technology enables our vehicles to be able to perform various tasks. For example, cars can currently merge into highways, parallel park, detect/avoid obstacles and be wary of pedestrians nearby. How have these machines learned the rules of the road so far and what will they be able to do next?

Keywords — Drive Assistance, Road Signs,

I. Introduction

In today's world, the increasing number of distractions among drivers have led to an increase in accidents and road fatalities [8]. To help address this issue car manufacturers have been incorporating driver assistance technology into their vehicles. With current drive assistance technology enables our vehicles to be able to perform various tasks, including merging into highways, parallel parking, detecting and avoiding obstacles, and being aware of nearby pedestrians [2,8]. This advancement raises questions about how these machines have learned the rules of the road thus far and what they will be capable of in the future.

To shed light on the development of intelligent driving systems, we have designed an experiment with the objective of creating and evaluating convolutional neural network (CNN) models for speed limit classification. By carefully selecting our dataset we aim to test and train a model capable of classifying speed limit signs with varying degrees of accuracy. The ability to classify speed limits from images and video holds practical application in a modern world, particularly in regards to autonomous driving, advanced driver-assistance, and advanced public transportation. Being able to accurately recognize speed limit signs will play a crucial role in the safety and promoting the effectiveness of these systems.

The dataset utilized in this experiment is sourced from Kaggle, a platform that facilitates access to datasets for building machine learning (ML) models and collaboratively solving data science challenges. Through careful manual processing we allocate 80 percent of the data for training and 30 percent for testing. To avoid unwanted biases, we maintained class distribution in both datasets.

By conducting this experiment, we aim to advance the understanding and development of deep learning models for speed limit classification, for improved road safety, more efficient driver assistance systems and continued progress of smart transportation systems.

II. Previous work

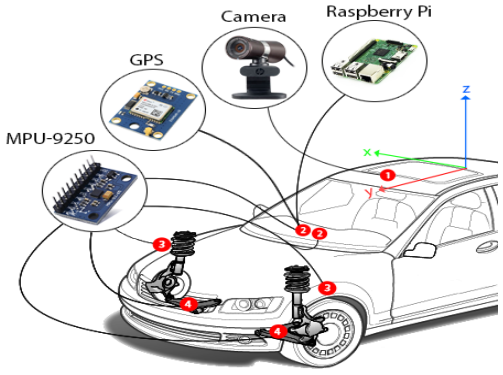
During our research on driver assistance and sign detection methods, we explored a wide range of approaches all aimed at improving the accuracy and efficiency of this important task. Among the various papers we examined, several stood out noteworthy to us.

Yousri Ouerhani; Ayman Alfalou; and Marc Desthieux presented a three-step approach for road

traffic sign recognition and identification using the commercial VIAPIXs module. The first step was to identify all objects in a scene that have any characteristics of traffic signs. The second step ins first-level recognition based on the correlation of where each detected object was compared with a set of reference images from a database. The third step involves a second level of identification to confirm/correct the previous identification. Their approach combines and adapts the Vander Lugt correlator with the nonlinear joint transformation correlator (JTC) to make reliable decisions on road traffic sign identification [4]. This approach can have broad practical applications in transportation as well as for drive assistance.

For cars to understand the road and world around them, their sensors need to know what's expected. Through the use of exteroception and proprioception, car sensors can recognize features of the road and how the car is supposed to move in select situations. Exteroception aims to understand anomalies and obstacles such as potholes, cracks, speed bumps, surface type, and road quality. Proprioception aims to understand vehicular movements such as lane changing, braking, skidding, hydroplaning, turning. The driving profile is also logged with proprioception as being a safe or potentially dangerous driver. These sensors help create the Intelligent Transport Systems such as Advanced Driver Assistance Systems (ADAS) and autonomous vehicles [14].

In our first survey, three different vehicles, drivers and terrains were assessed using the following technology pieces in Figure A



By assessing the drivers' patterns and terrain, the data can help inform machines on what to expect while on the road.

Another remarkable study by S. Maldonado-Bascón and their team introduced an automatic sign detection and recognition system that leveraged the use of support vector machines (SVMs) [7]. Their work showcased the robust and potential of SVMs for sign detection. They demonstrated promising performance accurately identifying and recognizing road signs, showing the potential of SVM based approach.

The next was the approach Fang, Chen, and Fuh for sign detection in more complex traffic environments. Noticing the intricate nature of the scenarios, they devised a novel method involving two neural networks. These networks where designed to extract features from traffic signs based on color and shape characteristics. With the unique perspective they incorporated fuzzy-set discipline and the use of Kalman filter for tracking. This method performed well in various weather and lighting conditions [11].

Continuing our research, we also found the work by Zhang [5]. Like the approach from Fang [11] their approach also focused on shape and color but introduced a "color visual saliency segmentation algorithm" [5]. This algorithm was developed to address the challenge posed by low contract images, which can significantly reduce accurate detection and recognition of signs. With the new propose "novel color visual saliency segmentation scheme they modified the fast Radial Symmetry Transform (RST) and propose an improved RST, name as IRST which helps" [5] more efficiently and accurately detect road signs.

Among the multitude of papers, we reviewed the study by Dhawan K caught our attention. Their work utilized, at the time, the state-of-the-art YOLO v3 and YOLO v4-tiny architectures, that have gained popularity in computer vision applications. With their implementation they achieved an performance accuracy rate of 95.85%, really demonstrating the

effectiveness of these models within the context of sign detection.[2]

Considering our limitations of time and resources, we made the decision to narrow our focus exclusively to evaluate and explore convolutional neural networks for sign detection. Partly inspired by Dhawan [2] and Bayoudh's hybrid 2D-3D CNN for traffic sign recognition [6]. We believed that leveraging CNNs could provide us with a strong foundation for developing an efficient and accurate speed sign detection system that fit within our project's constraints.

III. Experimental Design

The objective of our experiment is to create and evaluate the performance of convolutional neural network (CNN) models for speed limit classification using the data set obtained. From the dataset we can define the training and testing data and create a model that classifies a speed limit sign with a varying degree of accuracy. Being able to classify speed limits from images and video has practical applications in our modern world. Accurately recognizing and analyzing speed limit signs is vital in the safety and usage of autonomous driving, advanced driver-assistance, and advanced public transportation systems.

By performing the experiment, we plan to analyze the effectiveness of models that we generate as well as pre existing models. In addition, this experiment will provide insights into the model's performance metrics such as accuracy and less to allow us to determine the best model configuration for speed limit classification. The findings from this experiment will help with development of robust and effective deep learning models for speed limit classification which would lead to increased road safety, improving driver assistance systems, and overall lead to more efficient transportation systems.

IV. Data set

The dataset used for this dataset is from Kaggle, which allows users to find datasets for building AI models, publishing datasets, and working with other engineers to solve complex data science challenges. The dataset provided consists of various speed limits (40, 60, 80, 100) of real-world speed limit signs using cameras and has a sufficient number of examples of each speed limit class to avoid large amounts of bias and maintain balance between classes.

We manually went through the provided dataset and correctly split the data, into training and testing directories. A common practice is to allocate 80 percent of the data to training and 20 percent to testing. When splitting the data we trained to maintain class distribution in both training and testing datasets to avoid unnecessary bias or imbalances. The speed limits (40, 60, 80, 100) each have a subdirectory within the test and train directories. Images are placed within these subdirectories for the code to process. Figure 1 shows an example of the input image for training and testing.

Figure 1

V. Data Preparation

For data preparation of the given dataset, we need to ensure that there is consistency and compatibility for use with the CNN model that we generate. The images provided are all in the .png file extension and are 300 by 400 pixels. However, there may be some images that may not adhere to the size so we use the resize function to make sure all images are preprocessed to the same pixel dimensions. As part of the experiment, we will also look into various preprocessing techniques such as normalization and augmentation to improve the diversity and robustness of the model.

VI. Developing a CNN Model

Our CNN model was developed using a popular architecture that is commonly used for image classification tasks. CNN models are effective in extracting key features from images and have been successful in various computer vision applications. The structure of the CNN model in the code is built upon multiple layers.

The Input Layer takes in RGB images that are 300x400 pixels. The convolutional layers of the model consist of 23 filters with a size of 3x3. These layers apply convolutional operations to detect key features from the image. Additional convolutional layers increase the number of filters and extract more complex features with each pass. The rectified linear unit 'relu' function is applied after each convolutional layer to enhance the model's ability to learn and detect complex patterns.

Max Pooling Layers are also applied after each convolutional layer to downsample the feature maps, keeping the most important information while reducing the size. After the last convolutional layer a flatten layer is added to transform the 2D feature maps into a 1D feature vector to prepare the data for following fully connected layers. The first fully connected 'Dense' layer has 128 neurons that use the ReLU activation function to extract the complex patterns. The second 'Dense' layer has the same number of neurons as classes and uses the softmax activation function to produce class probabilities. The final step is to compile the model using an Adam optimizer. This allows for an adaptive learning rate optimization algorithm. The loss function 'sparse_categorical_crossentropy' is well suited for our multi-class classification problem. Our accuracy is the percentage of correctly classified samples.

VII. Results

During our performance testing of our model, we mainly focused on the effect of increasing the number of epochs. We mapped the results as shown in Figure 2. Based on these results, it seems that the model's performance improves gradually with each epoch up to a certain point. The accuracy, precision, recall, and F1 score increase consistently from epoch 5 to epoch 20, suggesting that the model is learning and improving over time.

However, it's important to note that starting from epoch 25, there is little to no improvement in the performance metrics. This could indicate that the model is reaching its optimal performance, and further training might not yield significant improvements. The number of epochs decides the number of times to change weights of the network. As epoch increases the boundary goes from underfitting to overfitting.

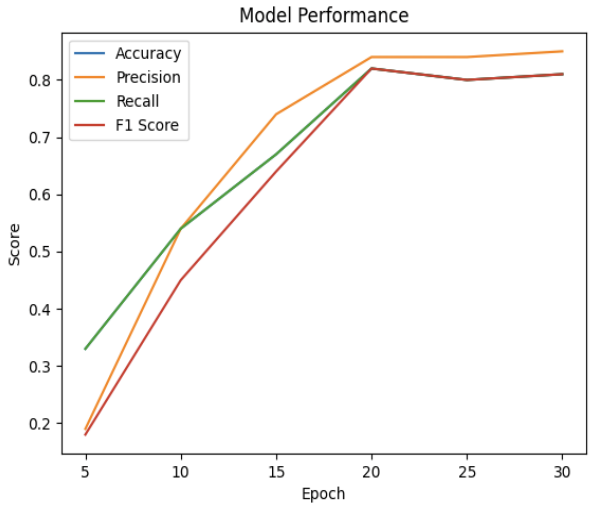


Figure 2

In the future we hope to measure validation and training error to define a number of epochs that is best fitted for identifying speed limits. As shown in Figure 3, the ideal number of epochs to have is determined by the point at which testing error no longer decreases. If testing error increases it's possible that the model is overfitted.

Figure 3

Furthermore, we would like to explore and evaluate some of the latest advancements in object detection models, like YOLO v7. Incorporating YOLO v7 into our research can help us better understand its advantages, limitations, and compatibility with our existing dataset and image processing techniques. Our goal is to remain at the forefront of technology by investigating and utilizing the most recent advancements. This will allow us to enhance the accuracy and efficiency of our speed sign classification system, ultimately making significant contributions to improving road safety.

VIII. Conclusion

In conclusion, our experiment focused on developing and evaluating convolutional neural network (CNN) models for speed limit classification. By utilizing a carefully selected dataset and implementing image processing techniques, we aimed to train a model that accurately classifies speed limit signs. The model successfully learned patterns in the speed classes and demonstrated accurately identified images. Increasing the number of epochs resulted in gradual improvements in accuracy, precision, recall, and F1 score metrics. Evaluating the model's performance on a separate test set is crucial to ensure its generalization and real-world effectiveness in classifying speed limit signs.

The findings from our experiment contribute to the advancement of deep learning models for speed limit classification, which is significantly benefiting autonomous driving, driver-assistance systems, and public transportation. Accurate recognition of speed limit signs enhances road safety and improves the efficiency of intelligent systems.

Acknowledgment

Thank you to Dr. Wagner for his continued support, feedback, and wisdom on this project. Thank you to the classmates of our CIS 735 course for constructive feedback on this project.

References

[1] Al-Ezaly, E., M. El-Bakry, H., Abo-Elfetoh, A. et al. An innovative traffic light recognition method using vehicular ad-hoc networks. Sci Rep 13, 4009 (2023). https://doi.org/10.1038/s41598-023-31107-8

[2] Dhawan K, R SP, R. K. N. Identification of traffic signs for advanced driving assistance systems in smart cities using deep learning. Multimed Tools Appl. 2023 Mar 4:1–16. doi: 10.1007/s11042-023-14823-1. Epub ahead of print. PMCID: PMC9985491.

[3] Rundo, F., "Deep Learning Systems for Advanced Driving Assistance", arXiv e-prints, 2023. doi:10.48550/arXiv.2304.06041.

[4] Ouerhani, Y (02/01/2017). "Advanced driver assistance system: Road sign identification using VIAPIX system and a correlation technique". Optics and lasers in engineering (0143-8166), 89 , p. 184.

[5] Zhang, T., Zou, J. & Jia, W. Fast and robust road sign detection in driver assistance systems. Appl Intell 48, 4113–4127 (2018). https://doi-org.libezproxy2.syr.edu/10.1007/s10489-018-1199-x

[6] Bayoudh, K., Hamdaoui, F. & Mtibaa, A. Transfer learning based hybrid 2D-3D CNN for traffic sign recognition and semantic road detection applied in advanced driver assistance systems. Appl Intell 51, 124–142 (2021). https://doi-org.libezproxy2.syr.edu/10.1007/s10489-020-01801-5

[7] S. Maldonado-Bascon, S. Lafuente-Arroyo, P. Gil-Jimenez, H. Gomez-Moreno and F. Lopez-Ferreras, "Road-Sign Detection and Recognition Based on Support Vector Machines," in IEEE Transactions on Intelligent Transportation Systems, vol. 8, no. 2, pp. 264-278, June 2007, doi: 10.1109/TITS.2007.895311.

[8] De-Las-Heras G, Sánchez-Soriano J, Puertas E. Advanced driver assistance systems (ADAS) based on machine learning techniques for the detection and transcription of variable message signs on roads. *Sensors*. 2021;21(17):5866. https://libezproxy.syr.edu/login?url=https://www.proquest.com/scholarly-journals/advanced-driver-assistance-systems-adas-based-on/docview/2571519276/se-2. doi: https://doi.org/10.3390/s21175866.

[9] Xu, Xianghua (05/01/2019). "Smart data driven traffic sign detection method based on adaptive color threshold and shape symmetry". Future generation computer systems (0167-739X), 94 , p. 381.

[10] Pazhoumand-dar, H., Yaghoobi, M. A new approach in road sign recognition based on fast fractal coding. *Neural Comput & Applic* **22**, 615–625 (2013). https://doi-org.libezproxy2.syr.edu/10.1007/s00521-011-0718-z

[11] Chiung-Yao Fang, Sei-Wang Chen and Chiou-Shann Fuh, "Road-sign detection and tracking," in *IEEE Transactions on Vehicular Technology*, vol. 52, no. 5, pp. 1329-1341, Sept. 2003, doi: 10.1109/TVT.2003.810999.

[12] XiongFei Liu and Fan Xiong 2020 IOP Conf. Ser.: Mater. Sci. Eng. 787 012034

[13] V Makarov et al 2018 IOP Conf. Ser.: Mater. Sci. Eng. 315 012016

[14] J. Menegazzo and A. von Wangenheim, "Multi-Contextual and Multi-Aspect Analysis for Road Surface Type Classification Through Inertial Sensors and Deep Learning," 2020 X Brazilian Symposium on Computing Systems Engineering (SBESC), Florianopolis, 2020, pp. 1-8, doi: 10.1109/SBESC51047.2020.9277846.

Appendix A

```
import os
import cv2
import numpy as np
import sklearn.metrics as metrics
import tensorflow as tf

def train(epoch_count):
    train_dir = 'train/'
    test_dir = 'test/'
    classes = ['40', '60', '80', '100']

    train_data = []
    test_data = []

    for cls in classes:
        path = os.path.join(train_dir, cls)
        for img in os.listdir(path):
```

```
        with open('train.txt', 'a') as f:
            f.write(path + '/' + img + '\n')
        img_array = cv2.imread(os.path.join(path, img))
        img_array = cv2.resize(img_array, (300, 400))
        train_data.append([img_array, classes.index(cls)])

for cls in classes:
    path = os.path.join(test_dir, cls)
    for img in os.listdir(path):
        with open('test.txt', 'a') as f:
            f.write(path + '/' + img + '\n')
        img_array = cv2.imread(os.path.join(path, img))
        img_array = cv2.resize(img_array, (300, 400))
        test_data.append([img_array, classes.index(cls)])

# Shuffle the data
np.random.shuffle(train_data)
np.random.shuffle(test_data)

# Preprocess the data
x_train = []
y_train = []
x_test = []
y_test = []

for feature, label in train_data:
    x_train.append(feature / 255.0)
    y_train.append(label)

for feature, label in test_data:
    x_test.append(feature / 255.0)
    y_test.append(label)

# Convert to NumPy arrays
x_train = np.array(x_train)
y_train = np.array(y_train)
x_test = np.array(x_test)
y_test = np.array(y_test)

"""
```

```
    Option 1
    """
    # Define the model
    model = tf.keras.models.Sequential([
        tf.keras.layers.Conv2D(32, (3, 3), activation='relu', input_shape=x_train.shape[1:]),
        tf.keras.layers.MaxPooling2D(pool_size=(2, 2)),
        tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
        tf.keras.layers.MaxPooling2D(pool_size=(2, 2)),
        tf.keras.layers.Conv2D(128, (3, 3), activation='relu'),
        tf.keras.layers.MaxPooling2D(pool_size=(2, 2)),
        tf.keras.layers.Conv2D(256, (3, 3), activation='relu'),
        tf.keras.layers.MaxPooling2D(pool_size=(2, 2)),
        tf.keras.layers.Flatten(),
        tf.keras.layers.Dense(128, activation='relu'),
        tf.keras.layers.Dense(len(classes), activation='softmax')
    ])
    # Compile the model
    model.compile(optimizer='adam',
                  loss='sparse_categorical_crossentropy',
                  metrics=['accuracy'])
    # Train the model
    history = model.fit(x_train, y_train, epochs=epoch_count, validation_data=(x_test,
y_test))

    model.save('speed.h5')
    evaluate_model(x_test, y_test, model, epoch_count)

def classify_speed(image_path):
    # Load the pre-trained model
    model = tf.keras.models.load_model('speed.h5')

    # Load and preprocess the input image
    img_array = cv2.imread(image_path)
    img_array = cv2.resize(img_array, (300, 400))
    img_array = img_array / 255.0
    img_array = np.expand_dims(img_array, axis=0)

    # Perform the prediction
    predictions = model.predict(img_array)
    class_index = np.argmax(predictions[0])
```

```
    classes = ['40', '60', '80', '100']
    speed_limit = classes[class_index]

    return speed_limit



def evaluate_model(x_test, y_test, model, epoch):
    # Perform predictions on the test set
    y_pred = model.predict(x_test)
    y_pred_classes = np.argmax(y_pred, axis=1)

    # Calculate accuracy
    accuracy = metrics.accuracy_score(y_test, y_pred_classes)

    # Calculate precision, recall, and F1 score
    precision, recall, f1_score, _ = metrics.precision_recall_fscore_support(y_test,
y_pred_classes, average='weighted')

    # Print the evaluation metrics
    print("Epoch: {}".format(epoch))
    print("Accuracy: {:.2f}".format(accuracy))
    print("Precision: {:.2f}".format(precision))
    print("Recall: {:.2f}".format(recall))
    print("F1 Score: {:.2f}".format(f1_score))

if __name__ == '__main__':
    train(5)
    train(10)
    train(15)
    train(20)
    train(25)
    train(30)

    image_path = 'test_image.png'
    print("Predicted Speed: ", classify_speed(image_path))
```

Appendix B

Datasets

Traffic Light Detection Dataset:
https://www.kaggle.com/datasets/wjybuqi/traffic-light-detection-dataset
Road Sign Detection:
https://www.kaggle.com/datasets/andrewmvd/road-sign-detection