TRAFFIC SIGN DETECTION USING CONVOLUTIONAL NEURAL NETWORK

Ravichandra Thota, Sivateja Reddy Kasi, Madhu Hannimi Reddy Tatiparthi, Revathi Chowdary Mallineni

Department of Computer Science

University of Missouri-Kansas City, Missouri, 64111

ABSTRACT

The creation and comparison of a deep learning-based architecture for identifying 43 types of traffic signals will be part of this research. The classification accuracy and prediction speed of common CNN-based designs were assessed. A simple and effective CNN (Convolutional Neural Network) architecture was designed based on the findings. It combines high prediction speed with great accuracy. Traffic sign recognition is a system used in self-driving cars and driving assistance systems that automatically recognizes various traffic signs and acts or guides them according to the signs. This system supports people when traffic limits are exceeded with a driving assistance system. This system guides the vehicle to the self-driving car according to the traffic signs. In this project, we developed a traffic sign recognition system using Deep Neural Networks Convolution Neural Network and VGG16. We used a German dataset containing about 50,000 images and 43 classes to develop this model. In the proposed model, our approach is to (i) sequentially detect the type of task present in the image, (ii) selectively preprocess the image to enhance the feature of the marker, and (iii) special. Is to make traffic sign suggestions from preprocessed images using trained images. Classify the detected challenge type networks, and finally (iv) the labeled proposal extracted from the image. These tasks are performed by implementing CNNs of various architectures and depths. The challenge detector and classifier are implemented using a VGG16 type architecture, but the localizer is similar to the UNet architecture.

1. INTRODUCTION

The recognition and classification of traffic signs is an intriguing problem in computer vision, and it's especially essential in the context of self-driving cars. Robust real-time traffic sign recognition algorithms are required for self-driving cars to become prevalent on roadways in the future. Many methods have been proposed in the literature for this purpose. However, current state-of-the-art technology is only trained on images taken without noise under natural light conditions. Also, the challenges of real-time image capture on the road can be diverse. Even reported models that can efficiently handle lighting changes have not been tested against other types of challenges such as blurring and discoloration. In these cases, it is unlikely that you will get satisfactory results. Therefore, it is very difficult to design an integrated model that can handle all these problems. To address these issues, we will introduce a new traffic sign recognition method based on a completely deep convolutional neural network (CNN). There are two motives for our model. First, algorithmic or statistical approaches to denoising, segmentation, and classification have proven to be highly computationally intensive due to species diversity and the level of challenges within the dataset. As a result, this approach is unsuitable for real-time applications. Therefore, one reason to consider neural network-based detection and classification schemes is computational efficiency during implementation. Second, in recent years, CNN has achieved incredible accuracy and efficiency in image classification. Dataset analysis shows that the number of challenge and sign types to consider is limited, and that the type of noise is different and unique for each image. Therefore, neural network-based noise classification can give good results.

In the next section of this paper explains about the testing the model with an overview of the model, the detailed architecture of the network used in each phase, network implementation and training procedures, and the test data provided in the dataset.

2. RELATED WORK:

Machine learning and deep learning have been the subject of several studies and research, particularly in the realm of self-driving automobiles. CNN is a well-known image classification architecture that is commonly utilized in image classification. This section highlights some of the

publications and projects that have most inspired us and helped us comprehend the issue.

We started by using a support vector machine (SVM) to find a solution to my problem. In this study, automatic character recognition was integrated into the road, and the model was able to recognize all kinds of characters, including circles, triangles, rectangles, and octagons. Due to the high success rate of the final prediction, we have shown that SVM may be a suitable model for this purpose.

There is a vast body of literature on traffic sign recognition and several reviews are available. As some recent studies have pointed out, either approach is generally better, as there is no standard publicly available benchmark dataset, primarily containing an extensive set of different traffic sign categories. Determining whether to produce overall results is very difficult. Most authors evaluate their approach to one of many public datasets, where the set of traffic sign categories is relatively limited.

Finally, I referred to this research study to better understand the CNN network. According to this study, most convolutional network architectures have the same design and structure, but the filter dimensions of each convolutional layer play an important role in image classification. Therefore, filters of various dimensions have been tested on those architectures. The dimension they investigated was 3 X 3, 5 X 5, 9 X 9, 13 X 13, 15 X 15, 19 X 19, 23 X 23, 25 X 25. This allowed us to experiment with different sizes and compare accuracy results based on different filter sizes. Through this white paper, I realized that when designing a model, you also need to keep in mind factors such as filter size dimensions and hyperparameters.

3. DATASET

The dataset used in this project is the German benchmark for traffic sign recognition. This data contains a total of 51,830 images, each of which is (32 x 32 x 3) in size. Where 3 represents the RGB channel of the color image. The dataset was further divided into 34799 training, 4410 validation, and 12630 test images.
[www.kaggle.com/datasets/meowmeowmeowmeowmeow/gtsrb-german-traffic-sign]

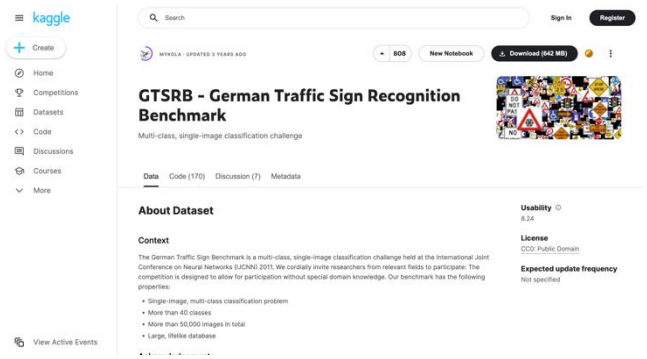

Fig. 1: About dataset

4. METHODOLOGY

This section details data preprocessing techniques and implemented CNN-based architectures, along with training details and metrics.

4.1. Overview

The approach adopted to create this computer vision model is to import and load datasets, perform exploratory data analysis (EDA) on datasets, visualize dataset variables and their properties, and fine-tune the model. Includes preprocessing the data to be done, designing the model. Use CNN to train and test your designed model using training and testing examples as shown in below figure.

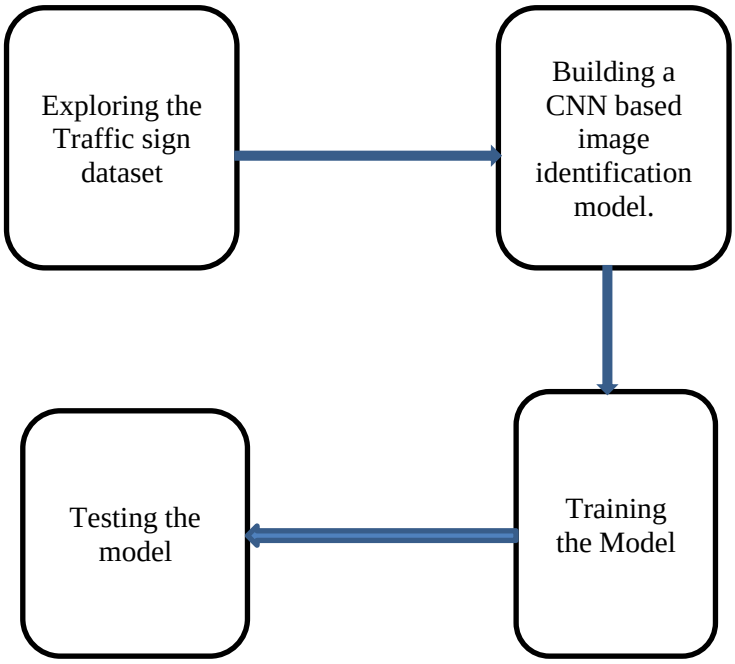

Fig. 2: Implementation

4.2. Languages & Libraries:

The Python programming language is particularly useful for creating machine learning models, artificial intelligence applications, and data analysis. Python programming is used for this work since it is an interpreted language with a fast code evaluation, simple syntax, and dozens of modules for analyzing real-world circumstances. A Python module for data processing that is frequently utilized by NumPy, and Pandas was also used. Jupiter Notebook is a web-based open-source tool that supports Python and other programming languages. Individual snippets can be debugged and ran more easily, and there are various advantages, including data training, exploratory data analysis, data visualization, data preprocessing, and data augmentation. TensorFlow is a machine learning platform that includes a rich library, flexible tools, and open-source core libraries for developing machine learning applications. It was used to train the NN model from sequence to sequence in this study. Kera's APIs (Application Programming Interfaces) are quite user-friendly, making

troubleshooting a breeze. Kera's is one of the most extensively used deep learning frameworks. Within TensorFlow, it's a massive, centralized system for offering machine learning workflows, hyperparameter training, and a variety of other concerns. Image and text categorization, data loading support, different utilities to transform raw data to the right dataset, reliable function normalization, and image rescaling are just a few of Kera's features. As well as the use of transforms. Flask is a compact, lightweight Python web framework with handy tools and capabilities that make building online applications in Python simple. With a single Python file, you can quickly build web apps, offering developers more freedom and making the framework more accessible to new developers.

4.3. Data Preprocessing

The dataset has the images of different sizes by using the below code, all the images resized to 30x30 from the tarin folder. After converting the images to desirable shape, images converted into data list of NumPy array using np.array( ) function. We can see the image data shape i.e., 39209 and labels shape i.e., 39209. The data and labels are stored as. npy format for to skip the conversion if we are using the dataset again.

Fig. 3: a) Resizing the images to 30X30. b) Converting image data and labels to NumPy array

4.4. Parameters Used

CNN learns filter values during the model training process. Before the training process, you need to specify some parameters here, such as the number of filters, the network architecture to use, the size of the filters, etc. Increasing the number of filters will make it more efficient to detect invisible patterns in the image. It also significantly improves the quality of extraction levels and pattern evaluations. Some of the important model parameters are:

Filters: The filter image convolution operation is used to identify image dependencies. As a result, slide the filter over the input image to generate a feature map. Again, performing convolution operations with different filters produces different feature maps. Filters and images are saved as a numeric matrix for operation.

Epochs: The number of times the full dataset is transported back and forth over the neural network is indicated by this entity. The dataset is too huge to enter into the system only once, thus many submissions are required to get the best results.

Batch Size: It is a number that represents the number of training samples in a batch, as opposed to the number of batches. The batch number is essentially a partition of the dataset that can be processed easily and accurately because the complete dataset cannot be provided to the neural network model at once.

Learning rate: The steepest descent method is an iterative optimization process used in machine learning. This iterative technique is repeated several times to acquire the best results. The learning rate is the parameter in this case. This also aids in the proper fit of the underfit chart.

Loss regularization: Not training faults, but adjustments or transformations made to reduce generalization errors. The learning algorithm has been modified in this way. This is a critical aspect in avoiding overfitting and ensuring precision. The use of dropout layers, the introduction of weight penalties, the expansion of datasets, and the built-in early stop to successfully modify hyperparameters like as epochs and batch counts are among the approaches used to execute data adjustments.

Dropout rate: The dropout layer selects some nodes in the neural network at random and moves them along the incoming and outgoing links. Both hidden and input layers can use this layer. Neural network training failures can be thought of as an ensemble collection of 2^n sparse networks, which is also known as the ensemble technique. This is useful for learning more robust functions from varied random subsets of neurons.

4.5. Proposed CNN Network

Finally, we built a convolutional neural network architecture that consisted of two stacks of three convolutional layers, an average pooling layer, and two fully connected layers. To minimize gradient eruptions and disappearances propagating through several layers, we employed stack normalization after each convolution layer. To avoid overmatches, dropout layers were applied after each stack.

```
model = Sequential()
model.add(Conv2D(filters=32, kernel_size=(5,5), activation='relu', input_shape=X_train.shape[1:]))
model.add(Conv2D(filters=32, kernel_size=(5,5), activation='relu'))
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(rate=0.25))
model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu'))
model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu'))
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(rate=0.25))
model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dropout(rate=0.5))
# We have 43 classes that's why we have defined 43 in the dense
model.add(Dense(43, activation='softmax'))
```

Fig. 4: The Sequential model

The 32 convolution filters in the first two convolution layers have a kernel size of 5. Grid Search for convolution layers was performed on several kernel dimensions: 3, 5, 7, 9, 11, 19, and 31. With kernel size = 3, the highest accuracy was achieved. After each convolution layer, the ReLU activation function was utilized as a non-linearity that set negative input to zero. To enhance the receptive field, the next three convolution layers each feature 64 convolution filters. Furthermore, there are 128 convolution filters in the final convolution layer. After that, an average pooling layer is applied to gather the essence of all received feature maps. Its output is routed through a 512-neuron

fully linked hidden layer. To calculate the probability corresponding to 43 classes in the GTSDB dataset, the final output layer includes 43 neurons with SoftMax as the activation function. For the most part, Random Search was used to build this architecture.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 26, 26, 32)        2432

 conv2d_1 (Conv2D)           (None, 22, 22, 32)        25632

 max_pooling2d (MaxPooling2D  (None, 11, 11, 32)       0
 )

 dropout (Dropout)           (None, 11, 11, 32)        0

 conv2d_2 (Conv2D)           (None, 9, 9, 64)          18496

 conv2d_3 (Conv2D)           (None, 7, 7, 64)          36928

 max_pooling2d_1 (MaxPooling  (None, 3, 3, 64)         0
 2D)

 dropout_1 (Dropout)         (None, 3, 3, 64)          0

 flatten (Flatten)           (None, 576)               0

 dense (Dense)               (None, 256)               147712

 dropout_2 (Dropout)         (None, 256)               0

 dense_1 (Dense)             (None, 43)                11051

=================================================================
Total params: 242,251
Trainable params: 242,251
Non-trainable params: 0
```

Fig. 5: Model summary

4.6. Training & Testing details

For model training and testing, the dataset is separated into two parts: 80 percent and 20%. The third comprehensive step is to train and validate the constructed ML model, which is done using the model.fit () method in the validation set.

4.7. Results

For various data samples utilized in training, testing, and validation to accurately recognize and categorize traffic sign photos, the following noteworthy outcomes were discovered.

The accuracy of the training set is approximately 95%.
• The validation set gives correct results with a 99.06 percent accuracy rate.

```
Epoch 17/20
981/981 [==============================] - 43s 44ms/step - loss: 0.2432 - accuracy: 0.9498 - val_loss: 0.0520 - val_accuracy:
0.9899
Epoch 18/20
981/981 [==============================] - 42s 43ms/step - loss: 0.2084 - accuracy: 0.9562 - val_loss: 0.0559 - val_accuracy:
0.9888
Epoch 19/20
981/981 [==============================] - 43s 44ms/step - loss: 0.2506 - accuracy: 0.9497 - val_loss: 0.0581 - val_accuracy:
0.9881
Epoch 20/20
981/981 [==============================] - 45s 46ms/step - loss: 0.2466 - accuracy: 0.9492 - val_loss: 0.0479 - val_accuracy:
0.9906
```

Fig. 6: Accuracy of the model

• The testing set produces results with 90.3% accuracy
Precision was found to be 97.8% and recall being 98.06% as per information.

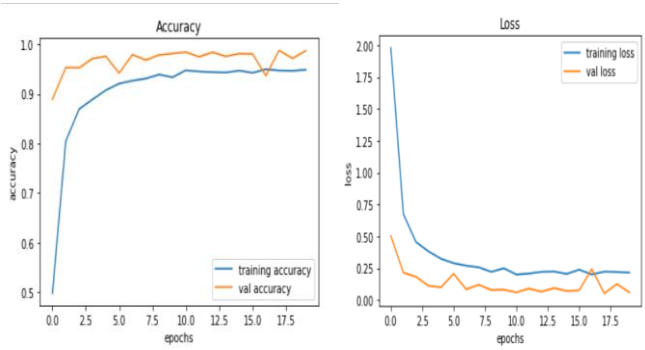

Fig. 7: Accuracy and Loss of the model

5. Conclusion

This research successfully builds a model of a traffic sign recognition system for a car as a traffic safety precaution, utilizing a convolutional neural network. Using Python machine learning programming and Jupiter Notebook's strong deep learning package, the full task is completed. The developed neural network model's parameters have been fine-tuned to achieve good accuracy. The training set was 95 percent accurate, whereas the test set was 90.3 percent accurate. It's worth noting that utilizing many CNNs might boost the model's learning rate, and while training photos are usually scaled, real-time image capture isn't as quick. Reliability is essential. Integration with systems for acquiring photos in the dark and at long distances, as well as spatial optimization, are other issues. Computer vision errors when capturing images with fog or smog should also be considered. Integrating this identifier into the card system can be a major challenge. This study can be applied to make vehicles and other related transportation systems intelligent and autonomous to the environment in which they are exposed. Real-world implementations require integration with other technical disciplines, including both hardware and software, enabling targeted actions that can be tracked with less user intervention. The capture device must also be able to use backup methods in the event of a failure. Center Net can be used to adapt the model like any other CNN model. Image feature extraction can be improved by understanding character recognition and related functions such as responsive system driving behavior in heavy and light traffic**.**

6. References

1. S. Maldonado-Bascon, S. Lafuente-Arroyo, P. Gil- Jimenez, H. Gomez-Moreno, and F. Lopez-Ferreras. Road-sign detection and recognition based on sup- port vector machines. Trans. Intell. Transport. Sys., 8(2):264–278, June 2007.
2. Valentyn Sichkar and Sergey Kolyubin. Effect of various dimension convolutional layer filters on traffic sign classification accuracy. Scientific and Technical Journal of Information Technologies, Mechanics and Optics, 19:546– 552, 06 2019.
3. X. Mao, S. Hijazi, R. Casas, P. Kaul, R. Kumar, & C. Rowen, Hierarchical CNN for traffic sign recognition. In 2016 IEEE Intelligent Vehicles Symposium (IV) (pp. 130-135). IEEE, 2016, June.

4. X. Changzhen, W. Cong, M. Weixin, and S. Yanmei, “A traffic sign detection algorithm based on deep convolutional neural network”, 2016 IEEE International Conference on Signal and Image Processing (ICSIP), 2016.
5. S. Raschka, & V. Mirjalili, “Python Machine Learning: Machine Learning and Deep Learning with Python. Scikit-Learn, and TensorFlow”.