Substation Danger Sign Detection and Recognition using Convolutional Neural Networks

Wajid Ali
School of Automation and Electrical Engineering, Lanzhou Jiaotong University, China
wajidaliustb@gmail.com

Guo Wang
School of Automation and Electrical Engineering, Lanzhou Jiaotong University, China
wangguo@lzjtu.edu.cn
(corresponding author)

Kalim Ullah
University of Science and Technology Bannu, Pakistan
engr_kalim125@yahoo.com

Muhammad Salman
School of Automation and Electrical Engineering, Lanzhou Jiaotong University, China
salmankhan5780@gmail.com

Sajad Ali
School of Automation and Electrical Engineering, Lanzhou Jiaotong University, China
sajad8371@gmail.com

Received: 14 November 2022 | Revised: 7 December 2022 | Accepted: 15 December 2022

ABSTRACT

This paper focuses on the training of a deep neural network regarding danger sign detection and recognition in a substation. It involved applying the concepts of neural networks and computer vision to achieve results similar to traffic sign and number plate detection systems. The input data were captured in three distinct formats, i.e. grayscale, RGB, and YCbCr, which have been used as a base for comparison in this paper. The efficiency of the neural network was tested on a unique data set involving danger signs present in industrial and processing facilities. The data set was unique, consisting of four distinct symbols. The trained data were selected so that they would not facilitate overfitting and also would not be under fitted. The accuracy of the model varied with the input type and was tested with two distinct classifiers, CNN and SVM, and the results were compared. The model was designed to be fast and accurate, and it can be implemented on mobile devices.

Keywords-CNN; neural networks; danger sign detection

I. INTRODUCTION

Signs play an important role in everyday life and come in a huge variety of color schemes [1], symbols, and words, originating from different regions, countries, and working environments. The main objective of this study was to recognize four different and distinct types of signs used in the industrial process worldwide. Automatic danger sign detection can help prevent accidents that could endanger employees and civilians. The presence of an application on a smartphone that can recognize and warn about the meaning and danger associated with a sign can save lives. It is an ideal device for new and less experienced workers. Nowadays, almost every person has a mobile phone, and about 80% of these are smartphones, able to integrate with the internet and run complex applications [2-4]. A neural network-based application can easily run on this type of phone. The target audience of this research is huge. It can benefit seasonal

workers, new industrial recruits, engineers, etc. The study focuses on using Neural Networks (NNs) to distinguish several danger signals. The application will be entirely web-based, having a mobile application to integrate with the user and even a web page for a direct online interface. The world has become heavily reliant on IT, and the COVID19 pandemic has shown that the potential of mobile applications and software is at its peak. A danger sign detection technique can be used in any application. The impacting factors are the sample size and training parameters, which have been tuned to give promising results and can be generalized to other symbols with slight modification and testing. The application shows it can accurately identify and classify data and warm people by recommending the prescribed precautions. This information is easy and can be quickly accessed on a phone.

The database used in this study was compiled from scratch, as no dataset of this kind existed. This involved extracting images from online repositories and personal sight visits. Image modification techniques were also used to add diversity and unpredictability to the dataset. The data set can be consistently updated and improved by adding more images. The more the images in the data set, the more accurate the training and testing will be. Although this paper is focused on the danger signs found explicitly in the substation, the same techniques could be used with other data sets to recognize danger signs in different environments. The concept involves computer vision and NN integration. The main objective is to replicate an accurate and similar response to that of traffic sign detection and number plate identification. They both use similar tools and techniques. While extensive literature exists in the field of traffic sign detection, studies related to the identification of signs in other environments are limited at best and non-existent to the best of our knowledge. This study aims to fill this scientific gap.

II. BASIC ABOUT CNNS

A typical Convolutional Neural Network (CNN) contains pooling, activation, normalization function, convolutional and Multinomial Logical Regression (MLR) layers. Usually, the last layer of the network is the MLR. This layer has a subpart known as the softmax function, which determines the final classification of data. The structure of the CNN model is chosen according to the difficulty and the problem that needs to be solved. The CNN model is designed with specific layers and neurons in mind. We can aim for higher accuracy for our results, but this will require more training time, a larger data set, and a more complex network structure [5].

III. RELATED WORK

The ongoing progress in the field of NNs includes improvements in the processing power required, reduction of the complexity of the algorithm, and advancements in computer vision. The closest mainstream topic related to computer vision, CNN, and image identification, is traffic sign or number plate detection. Many papers have been published regarding traffic sign detection with various techniques and methods [1, 6-8]. Most of these studies have existing data sets. These sets are classified into groups and are ready to be fed into the NN algorithm. In the current study, we had to make and classify the data and set and test the identifying markers and check their accuracy for our algorithm from scratch.

IV. CONVOLUTIONAL NEURAL NETWORKS

NNs are often used to train and identify objects. These can be images, trends, anything that can be quantized in the form of data. Though CNNs are a powerful tool, they require high computational power and resources to function properly [9]. As microelectronics gets smaller, is makes nano electronics more prevalent and viable in the electronic landscape. Accurate NN models require several layers, increasing the algorithm's complexity, the computational power required, and the energy needed to execute and process it. Fortunately, the semiconductor industry has made this possible, providing handheld devices with the power of a modest computer. Researchers have tried bringing deep learning models implemented in CNNs to mobile devices. These belong to one of the three following categories:

- Network pruning [10]
- Network quantization [11]
- Training small networks directly.

Finding the ideal number of layers and neurons required for the accurate and efficient working of the network is necessary. Overfitting, excess complexity, and several hidden layers can make the model accurate regarding the corresponding training data but sensitive to change, lacking robustness. The effects of overfitting and its correlation with robustness, validation, and training data were studied in [12].

There have been improvements in different techniques and methodologies to advance the techniques used in NNs, such as [13, 14]. CNNs have strong discrimination powers in terms of classifying objects. They can also accurately localize a portion of an image or data cluster using tools such as regression. In our efforts to deploy a model of CNN compatible with image extraction, we took inspiration from literature such as [14-16]. The main methodology applied in recent literature for detection networks are:

- Direct detection [14, 15]
- Regional detection [13, 17]

A. Direct Detection

The direct detection and prediction algorithms are based on the position or class of the target object. This information is provided with either regression or classification, depending on the problem and type of learning involved [18]. This can be extracted from the convolutional layer of the algorithm, making runtime faster in comparison.

B. Regional Detection

In regional detection, several regions are generated for the data in question, then data classes, and then the prediction operation is performed on object position and class for every region. In this method, regression and classification are conducted twice in different stages. This makes the regional detection method longer and more processing intensive, but more accurate in the computation of direct detection.

For our purpose of danger sign identification and recognition, the regional detection method is preferred, as accurate identification of hazardous symbols can help prevent accidents and provide precautionary safety information for workers. There is no processing and time restraint, thus, the delay is an acceptable trade-off for higher accuracy.

V. BOUNDARY DETECTION TECHNIQUES

Many CNN object identifiers use boundary boxes and class labels to identify and extract the exact image for learning [19, 20]. This can be useful in predicting the exact boundary boxes for the classified images. A danger sign varies depending on the type of hazard. Some are triangular, while others are square or circular. These distinct and different bound conditions for the sign make image extraction significantly more accurate. We can identify an image using its boundary estimation, segmentation of the image itself, or sign and symbol category classification. The pixel segmentation mask would require high processing power and would be slower in identifying the letters and words shown on the symbol. However, this is possible and has many potential applications, such as universal danger sign detection software [20]. This could be compatible with all languages and work environments. Such ideas have been forgone, as failure to identify a single letter in the sign can cause false or inconclusive results. Thus the main techniques used were boundary detection and symbol recognition.

VI. DEEP NEURAL NETWORKS

Deep Neural Networks (DNNs) have become an indispensable tool in image classification used in many applications, such as facial recognition, voice analysis, word or text analysis, etc. Using DNN models helps in achieving high accuracy because of their ability to train and recognize complex input data [21]. The definition of a DNN is defined as a network with a level which is represented by the presence of three or more hidden layers.

VII. CONVOLUTIONAL NEURAL NETWORKS-SUPPORT VECTOR MACHINE

Support Vector Machine (SVM) is a supervised model in machine learning [22] that needs to be given labelled input data and a classification algorithm is used to train itself for each given label. Its main advantage lies in its speed and higher accuracy with a lower input sample count. This can make this algorithm ideal for a task requiring robust classification [23-25]. The SVM technique is used mostly as a classifier in data evaluation. The ability of SVM in data analysis can help find the ideal margin between two distinct classifications. SVM is used to determine a hyperplane for each set of distinct outcomes from a given input. If the hyper plain is situated ideally between the trained data sets, it can provide a degree of robustness to the system. SVM is used in traffic signal recognition and training. The basic CNN is used for convolution and training, while the Support Vector Classifer (SVC) is used for classification. The hybrid combination of SCV and CNN is combined to give accurate results for a small number of training sets [26].

VIII. METHODOLOGY

The training of the system involved five basic steps:

1. Making a data set appropriate for integration.
2. Deciding the most appropriate and accurate type of NN to choose.
3. Training strategy used and data optimization in order to acquire the best results.
4. Model selection involves the layers and neurons in the algorithm, basically determining the complexity of the algorithm and its robustness.
5. The model in question used for the training of our model has 3 convolutional layers followed up by a pooling layer each. The end results of the layers are fed into a flattened layer to convert a 3-dimensional image into 1-dimensional weights. These weights are used in the deep layer to find the weight coefficients to recognize and categorize the image into one of 4 different categories.

The first step was taking in the respective input format. The images captured for processing could be used in one of three ways:

- Grayscale [27]
- RGB [28]
- YCbCr [29]

IX. IMAGE ANALYSIS

A. RGB

The RGB data set is the default spectrum used as input for most image recognition applications. Though it is thoroughly used, our results indicated that a modification of RGB, known as YCbCr, gives better results. The luminance information, known as Y, is scattered in all the dimensions of RGB. This information can be used to get better image parameters for training data. Below is a simple conversion from RGB to YCbCr [31].

$$Y = 0.299R+0.587G+0.114B \quad (1)$$

$$Cb = 0.564(B-Y) \quad (2)$$

$$Cr = 0.713(R-Y) \quad (3)$$

B. YCbCr

The respective training error was the least in YCbCr results for the limited training data set. The YCbCr is inherently superior to grayscale and RBG, as in grayscale many colors and contact information are lost and therefore are not taken into account while training. Any colored image can be converted into a grayscale by multiplying the colored coefficients (usually RGB) with constants. This converts 3-dimensional data into a 1-dimensional data set. Grayscale results were faster to train but had a training error twice that of the YCbCr data set [30]. The model was trained with the same image data but with different feature extraction.

C. *Grayscale*

The following converts an RGB image to grayscale:

$$\text{Grayscale} = 0.299R + 0.587G + 0.114B \quad (4)$$

The grayscale image is less complex as the data can only vary in one parameter. This can help make the data set more robust and less prone to overfitting, though much of the image information is lost when converting an image into grayscale data. The decrease in complexity did not adversely affect the results. Quite the contrary, the grayscale results proved better than the RGB.

X. TRAINING INFORMATION

The training process involved using four distinct image types for categorization. These were:

- Electric hazard
- Flammable risk
- Radiation danger
- Toxic materials

Fig. 1. Image categories.

We chose similar image shapes as already much work has been done on boundary identification markers. Instead of focusing on the shiny shape, we trained the CNN to identify the color pattern and the symbols present [32]. Each image was similar to the others to an extent, e.g. similar sign shape and base color contrast (yellow and black). Thus, the main objective was convolving the image and using the DNN nodes to recognize the district shapes.

XI. IMAGE STANDARDIZATION

We used bounding box regression to standardize the extracted image from the input. It can be taken from an angle and still correctly extract the image [14]. The regression vector can compress or expand the image to the ideal size for data input. We used shape classification for images with different shapes. Most images are of similar shape, including circular, triangular, rectangular, square, hexagonal, and octagonal. For the ease of data usage, we normalized the coordinates of both the x and y-axis. We can use this to find the exact extraction of all shapes except circular images and in our data set, there were no circular image boundaries, so that was not a major concern. Circular images are detected with small sheer errors, though the CNN-trained system was still able to recognize the symbols and give accurate results.

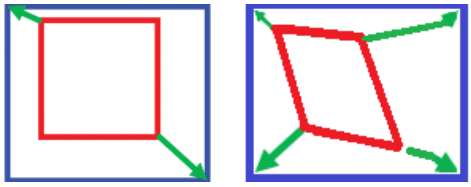

Fig. 2. Image fitting using regression vector.

XII. MODEL OVERVIEW

The model was trained on the gathered images that made a completely new and authentic data set. The network parameters were tuned during training and validation. ReLu, Softmax Deep CNN in a combination with SVM were used. Figure 3 shows some randomly selected images from the data set.

Fig. 3. Samples of the training set.

The data used for training consisted of 425 images, and the testing set included twice that number. The convolutional layers were used to segment the image into 64 blocks, each for a different color, for both RGB and YCbCr sets. The network architecture is shown in Figure 4. We used 3 convolutional layers for quick response. More can be incorporated to increase the accuracy of the results. The training images were transformed into YCbCr-colored data.

$$\text{Feature map size} = N\text{-}F+1 \quad (5)$$

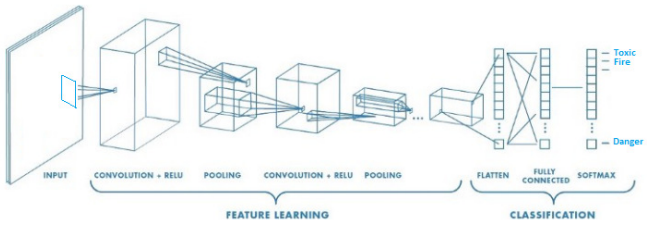

Fig. 4. Network architecture.

XIII. DETAILED WORKING OF THE MODEL

In our training, we passed 3 convolutional layers followed by max pooling. The input image was sized at 64×64 pixels. After passing through the convolution and ReLu layer, our resulting image was a 62×62-pixel image and our kernel was 3×3. We extracted 16 feature maps in these two stages. We have monitored and identified 448 parameters in this step. Then, max-pooling by a 2×2 pool size and 2 unit stride took place, and the processed image reduced in size to 31×31 pixels. This input image was fed to the second convolution layer, resulting in a 29×29-pixel image, granting 32 feature maps and 4046 parameters. The next max pooling layer resulted in a 14×14-pixel image. Pool size, kernel, and stride were kept constant for all convolution and pooling layers. By the last convolution layer, we had 64 feature maps of a 6×6-pixel image. After, a flatten operator was used to convert all the data into a linear array which passed as weights to the neurons. Our model has a total of 388 neurons, 256 in the 1st layer, 128 in the 2nd layer, and 4 in the last layer for the respective outputs. Figures 5-6 show the features of the modeled system.

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 62, 62, 16)	448
max_pooling2d (MaxPooling2D)	(None, 31, 31, 16)	0
conv2d_1 (Conv2D)	(None, 29, 29, 32)	4640
max_pooling2d_1 (MaxPooling 2D)	(None, 14, 14, 32)	0
conv2d_2 (Conv2D)	(None, 12, 12, 64)	18496
max_pooling2d_2 (MaxPooling 2D)	(None, 6, 6, 64)	0
flatten (Flatten)	(None, 2304)	0
dense (Dense)	(None, 256)	590080
dense_1 (Dense)	(None, 128)	32896
dense_2 (Dense)	(None, 4)	516

Fig. 5. CNN model data.

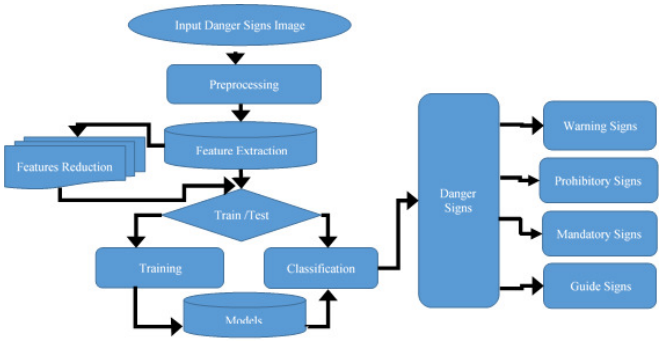

Fig. 6. Flow diagram.

XIV. SVM

The system was tested using both SVM and CNN, and both linear and non-linear SVM types were tested. Linear SVM uses a dot product between two features, and its product is the linear product combination of the input features.

$$f(ax_n, bx_m) = f(ax_n) + f(bx_m) \quad (6)$$

In non-linear SVM, this rule is not valid. Instead, it uses Kernel. This reduces the dimensionality of the input into various classifiers, replacing the dot product of linear SVM. This SVM can be used to learn shapes in higher dimensional data when using the Kernel.

XV. NEURAL NETWORK

An NN with more than one layer is non-linear, and there are 3 main activation functions used in such cases, which are:

$$\text{Logistic function } z = \sigma y \quad (7)$$

$$\text{Hyperbolic tangent } z = \tanh y \quad (8)$$

$$\text{Softmax } \emptyset(y) = \frac{e^{y_i}}{\sum_{i=1}^{k} e^{y_i}} \quad (9)$$

XVI. COMPARISON

For our study, we used the Softmax function. A basic understanding of NNs is necessary for the classification of data and while comparing SVM and NN. In contrast, SVM was a multipurpose non-linear NN designed for classifying the problem. Thus we based our classification approach on the NN. Another reason for choosing SVM was the lack of a large data set. SVM is simpler than the complex NN and has given better results on smaller data sets. Thus our CNN model used the ReLu classifier and our SVM model used the Softmax activation function to clearly identify the effect on filter training and its effect.

XVII. CNN

In our CNN, we have used ReLu (Rectified Linear Unit) as activation function and the Dropout Layer. The NN is assumed to have:

- Linear independence in the input features.
- Low dimensionality in the input space.

By implementing various convolution layers and pooling, we lowered the dimensionality of the image. This helps us resolve the problem of dimensionality by reducing the data size and making learning easier. The purpose of an activation function is to provide non-linearity, but we don't want any negative activation function because that would not assume that the data are independent. We used ReLu as our activation function because it is positive and very easy and simple to model [33]. It is easy to calculate and involve the only comparison between the input and 0.

$$ReLU: f(x) = max(0, x) \quad (10)$$

This is the last part of feature extraction before it is fed into the hidden layer for training and finding the neuron coefficient weights. The weights are necessary for classification as they

assign the neuron their values which recognize features and result in the identification and classification of an image. For a probabilistic result, we used the Softmax activation function [34]. This will give us as output one of the four considered classes each time.

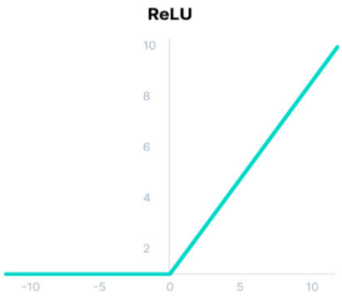

Fig. 7. ReLU function.

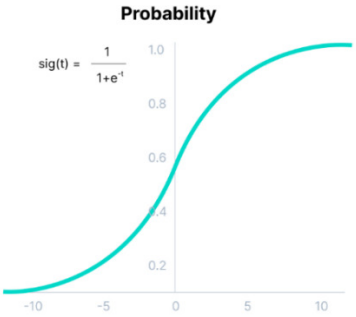

Fig. 8. Softmax function.

$$softmax(z_i) = \frac{\exp(z_i)}{\sum_i \exp(z_i)} \quad (11)$$

For loss calculation, we used Categorical Cross-Entropy loss [35]. This gives us the error caused due to training the data set.

$$CE = -\sum_i^c t_i \log(f(s)_i) \quad (12)$$

$$CE = -\sum_{i=1}^{c'=2} t_i \log(f(s)_i) = -t_i \log\big(f(s_1)\big) - (1 - t_1) log(1 - f(s_1)) \quad (13)$$

XVIII. EXPERIMENTAL RESULTS

Figure 9 shows the training accuracy and loss of the respective data. The model was run for 100 epochs, approaching a maximum accuracy of 91%. The loss is approaching a value close to 8%, signified by the value of 0.3. The grayscale and the YCbCr sets used the CNN classifier, but the RGB data set were modified to use the SVM classifier. The results showed that the RGB classifier did not increase by any significant value after 100 epochs, maybe due to the data limitation. Still, the validation accuracy was very dynamic and fluctuated consistently. In contrast, both grayscale and YCbCr dataset verification accuracy increased and showed a much smoother increase with a few disturbances and small dynamic behaviors, providing concrete proof that the CNN classifier is better and more accurate than SVM for smaller data sets.

The results show that the model's training and validation accuracy are consistently increasing with a slight offset. This is the ideal result required. It is trained on the modal parameters while still having a significant variation to be considered robust. Thus we conclude that the YCbCr model is well-trained and has a high validation accuracy to be considered robust and accurate. The loss graph shows the training and validation losses. It is based on (12) and (13). The cross-entropy loss is used to distinguish two distinct probability distributions. We can see this as proof of the model's ability to recognize and identify the images. The validation loss function traces a similar path in a logarithmic pattern. This is all consistent with accurate and reliable results.

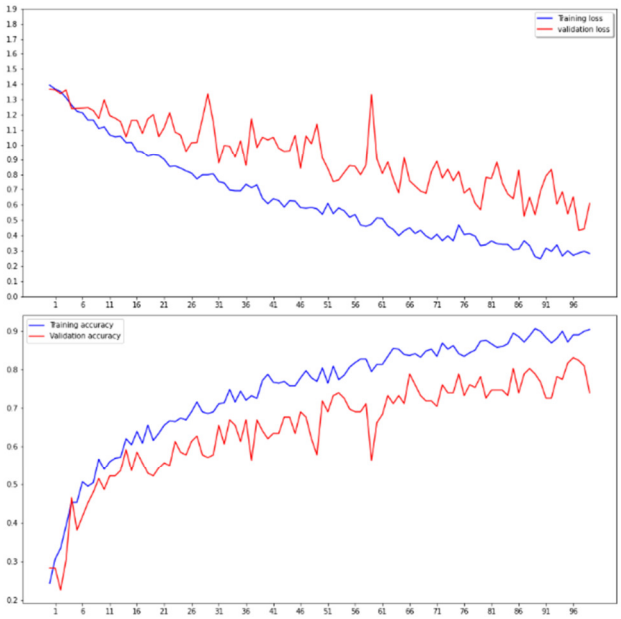

Fig. 9. Training results and loss comparison (YCbCr 100 epochs). Up: loss, down: accuracy.

Similarly, we trained the model with RGB and grayscale images. The results are apparent in Figure 10. It is clear that the RGB data are under-fitted. In the literature, it has been observed that using SVM as a classifier with a small or limited data pool will result in a less accurate and under-fitted plot. The exact pattern is being observed here. Regarding the grayscale results, they are accurate, robust, and ideal. The data are accurately trained and validated, and the loss function is within range. The reason for comparing grayscale and RGB images was to test how the classifier's variation would affect the training and validation results. The RGB dataset used an SVM classifier, while the grayscale dataset used a NN classifier, specifically a CNN classifier. The grayscale data were simpler and less detailed but had better results. As mentioned above, a small dataset could be the reason for low accuracy while using SVM, but the CNN classifier gave ideal results, both with the grayscale data set and YCbCr data. So in our study, the CNN classifier was found to be superior.

Figure 11 shows how increasing training time and duration affected the training accuracy. The YCbCr model validation

data closely followed the training results with a slight set. We increased to training duration to observe the highest accuracy provided by this dataset. We kept all the other parameters, such as the learning rate and the dataset, constant. The training accuracy peak values were achieved at 200 epochs, after which its increase in accuracy was minuscule, while it even started to decrease at 400 epochs. The observed results are consistent with the literature and show that our model's ideal training time is 200 epochs for our specified learning rate. Over-training can result in a loss of accuracy, and this concept is demonstrated here. We also identified a peak accuracy of 99.6%

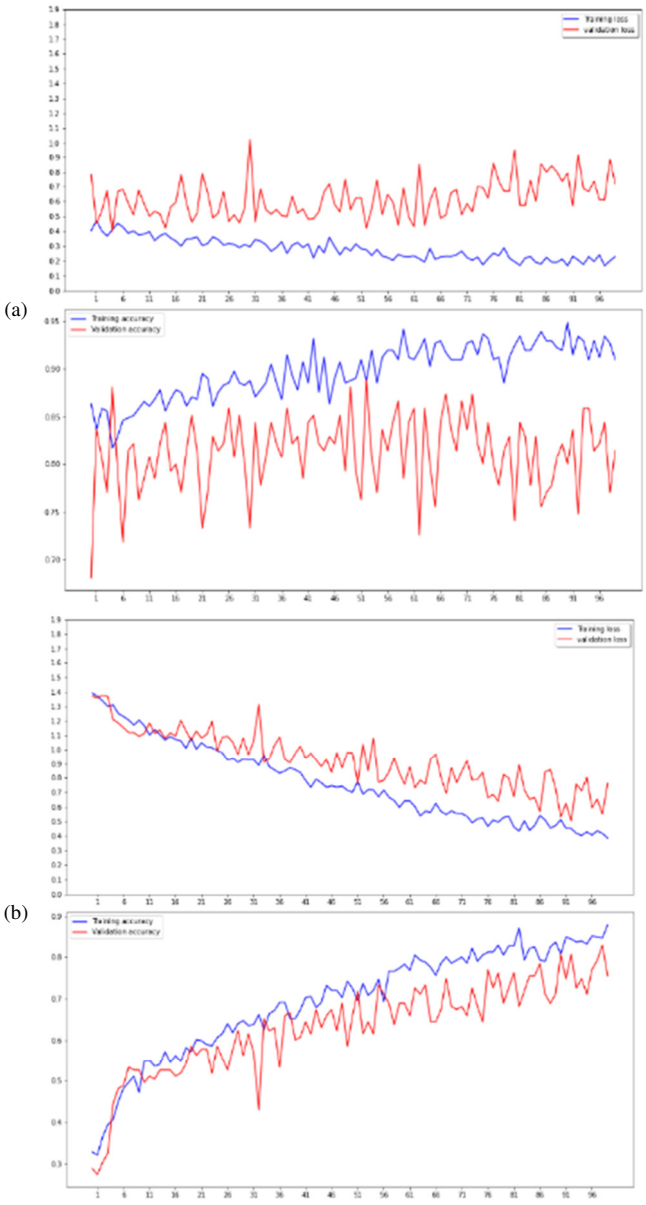

Fig. 10. (a) RGB image training, (b) grayscale image training (100 epochs). Up: loss, down: accuracy.

The effect and duration varied for the number of epochs the network was trained. The result for training and validation accuracy achieved at 500 epochs for the image dataset are shown in Table I. Only the training duration was varied, and readings at 10, 100, and 500 were plotted. The table values were gathered using the same above-mentioned plots extended to 500 epochs, but a smoothing filter was used in order to make the results more consistent and trends easier to identify.

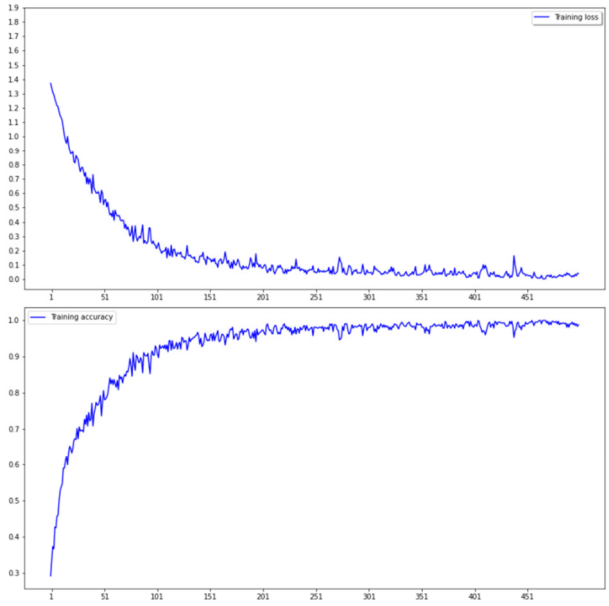

Fig. 11. YCbCr image training for 500 epochs. Up: loss, down: accuracy

TABLE I. RESULTS OF VARIATION OF IMAGE INPUT TYPE AND TRAINING

Time (s)	Epochs	Training accuracy	Validation accuracy	Type of input image
19.13	10	70.7%	66.3%	Grayscale
19.79	10	73%	68%	RGB
20.10	10	86.0%	71.1%	YCbCr
174.11	100	85.3%	80.1%	Grayscale
175 .91	100	89.4%	79.9%	RGB
178.35	100	91.4%	86.7%	YCbCr
952.58	500	93.5%	83.7%	Grayscale
972.32	500	94.4%	80.3%	RGB
982.11	500	97.1%	92.5%	YCbCr

Table I compares the grayscale and YCbCr accuracies. The training model is identical. The only difference is that the image set input was extracted in YCbCr format, which is a 3-dimensional input in matrix form, while grayscale is a 1-dimensional matrix input. This is also evident in the fact that the grayscale program runs in a shorter duration. Also, the YCbCr data show the highest accuracy, followed by the grayscale, and RGB models.

XIX. CONCLUSION

In this paper, we observed the effects on CNN with some slight modifications on a totally new dataset. We used a dataset with 4 different types of images. We varied several factors and compared their effects on the data training and validation accuracy. The image inputs varied and grayscale, RGB, and YCbCr images were utilized. The most accurate results came from the YCbCr model, as it was able to incorporate the effects

of grayscale and normal colors in the mathematical model. The difference was not significant but noticeable nevertheless. Thus the YCbCr image input proved to be a better input type for data extraction. CNN and SVC were used for image classification. The results were recorded, and it was observed that CNN provided better validation accuracy. The most possible reasons for this could be the smaller data set or the lower training duration and the average learning rate. We observed the accuracy variation due to under and over-training from 5 to 500 epochs and their results. The best results are between 100 and 200 epochs regarding time and accuracy. We varied the classification method, the kernel used, and the number of convolution layers. Other factors affecting the accuracy are the optimizer and the loss function calculation, but we did not observed any effect of these factors. We also confirmed the superior behavior of the CNN classifier for this dataset in comparison to the SVM. For future recommendations we should use a model with more layers and a bigger dataset, in order to have a more robust model with better classification ability.

The limitations of this study include the effects of optimizer calculation of the loss function for various values in order to find the optimal position. These steps were not monitored and can be considered as a project for future work and possible improvements.

This study was helpful in identifying that the best method to classify input image is CNN, at least for smaller datasets, and finally, it shows that CNN provides better classification results than SVM.

REFERENCES

[1] H. S. Lee and K. Kim, "Simultaneous Traffic Sign Detection and Boundary Estimation Using Convolutional Neural Network," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 5, pp. 1652–1663, Feb. 2018, https://doi.org/10.1109/TITS.2018.2801560.

[2] "US smartphone ownership 2021," *Statista*. https://www.statista.com/statistics/219865/percentage-of-us-adults-who-own-a-smartphone/.

[3] "Mobile Fact Sheet," *Pew Research Center: Internet, Science & Tech*. https://www.pewresearch.org/internet/fact-sheet/mobile/.

[4] D. G. Gibson, A. Pereira, B. A. Farrenkopf, A. B. Labrique, G. W. Pariyo, and A. A. Hyder, "Mobile Phone Surveys for Collecting Population-Level Estimates in Low- and Middle-Income Countries: A Literature Review," *Journal of Medical Internet Research*, vol. 19, no. 5, May 2017, Art. no. e7428, https://doi.org/10.2196/jmir.7428.

[5] G. Montavon, W. Samek, and K.-R. Muller, "Methods for interpreting and understanding deep neural networks," *Digital Signal Processing*, vol. 73, pp. 1–15, Feb. 2018, https://doi.org/10.1016/j.dsp.2017.10.011.

[6] S. Song, Z. Que, J. Hou, S. Du, and Y. Song, "An efficient convolutional neural network for small traffic sign detection," *Journal of Systems Architecture*, vol. 97, pp. 269–277, Aug. 2019, https://doi.org/10.1016/j.sysarc.2019.01.012.

[7] A. Bouti, M. A. Mahraz, J. Riffi, and H. Tairi, "A robust system for road sign detection and classification using LeNet architecture based on convolutional neural network," *Soft Computing*, vol. 24, no. 9, pp. 6721–6733, May 2020, https://doi.org/10.1007/s00500-019-04307-6.

[8] R. Antar, S. Alghamdi, J. Alotaibi, and M. Alghamdi, "Automatic Number Plate Recognition of Saudi License Car Plates," *Engineering, Technology & Applied Science Research*, vol. 12, no. 2, pp. 8266–8272, Apr. 2022, https://doi.org/10.48084/etasr.4727.

[9] Y. Hatolkar, P. Agrawal, and S. Patil, "A Survey on Road Traffic Sign Recognition System using Convolution Neural Network," *International Journal of Current Engineering and Technology*, vol. 8, no. 1, pp. 104–108, Jan. 2018, https://doi.org/10.14741/ijcet/v.8.1.21.

[10] C. Qi *et al.*, "An efficient pruning scheme of deep neural networks for Internet of Things applications," *EURASIP Journal on Advances in Signal Processing*, vol. 2021, no. 1, Jun. 2021, Art. no. 31, https://doi.org/10.1186/s13634-021-00744-4.

[11] M. Nagel, M. Fournarakis, R. A. Amjad, Y. Bondarenko, M. van Baalen, and T. Blankevoort, "A White Paper on Neural Network Quantization." arXiv, Jun. 15, 2021, https://doi.org/10.48550/arXiv.2106.08295.

[12] S. Lawrence, C. L. Giles, and A. C. Tsoi, "Lessons in Neural Network Training: Overfitting May be Harder than Expected.," in *Fourteenth National Conference on Artificial Intelligence*, Menlo Park, CA, USA, Jan. 1997, pp. 540–545.

[13] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 6, pp. 1137–1149, Jun. 2017, https://doi.org/10.1109/tpami.2016.2577031.

[14] W. Liu *et al.*, "SSD: Single Shot MultiBox Detector," in *14th European Conference on Computer Vision*, Amsterdam, Netherlands, Oct. 2016, pp. 21–37, https://doi.org/10.1007/978-3-319-46448-0_2.

[15] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *IEEE Conference on Computer Vision and Pattern Recognition*, Las Vegas, NV, USA, Jun. 2016, pp. 779–788.

[16] M. Bhalekar and M. Bedekar, "D-CNN: A New model for Generating Image Captions with Text Extraction Using Deep Learning for Visually Challenged Individuals," *Engineering, Technology & Applied Science Research*, vol. 12, no. 2, pp. 8366–8373, Apr. 2022, https://doi.org/10.48084/etasr.4772.

[17] J. Dai, Y. Li, K. He, and J. Sun, "R-FCN: object detection via region-based fully convolutional networks," in *30th International Conference on Neural Information Processing Systems*, Red Hook, NY, USA, Dec. 2016, pp. 379–387.

[18] M. Njah and R. E. Hamdi, "A Constrained Multi-Objective Learning Algorithm for Feed-Forward Neural Network Classifiers," *Engineering, Technology & Applied Science Research*, vol. 7, no. 3, pp. 1685–1693, Jun. 2017, https://doi.org/10.48084/etasr.968.

[19] L. Alzubaidi *et al.*, "Review of deep learning: concepts, CNN architectures, challenges, applications, future directions," *Journal of Big Data*, vol. 8, no. 1, Mar. 2021, Art. no. 53, https://doi.org/10.1186/s40537-021-00444-8.

[20] S. Sahel, M. Alsahafi, M. Alghamdi, and T. Alsubait, "Logo Detection Using Deep Learning with Pretrained CNN Models," *Engineering, Technology & Applied Science Research*, vol. 11, no. 1, pp. 6724–6729, Feb. 2021, https://doi.org/10.48084/etasr.3919.

[21] I. H. Sarker, "Deep Learning: A Comprehensive Overview on Techniques, Taxonomy, Applications and Research Directions," *SN Computer Science*, vol. 2, no. 6, Aug. 2021, Art. no. 420, https://doi.org/10.1007/s42979-021-00815-1.

[22] A. Shmilovici, "Support Vector Machines," in *Data Mining and Knowledge Discovery Handbook*, O. Maimon and L. Rokach, Eds. Boston, MA, USA: Springer US, 2005, pp. 257–276.

[23] H. Basly, W. Ouarda, F. E. Sayadi, B. Ouni, and A. M. Alimi, "CNN-SVM Learning Approach Based Human Activity Recognition," in *9th International Conference on Image and Signal Processing*, Marrakesh, Morocco, Jun. 2020, pp. 271–281, https://doi.org/10.1007/978-3-030-51935-3_29.

[24] S. Ahlawat and A. Choudhary, "Hybrid CNN-SVM Classifier for Handwritten Digit Recognition," *Procedia Computer Science*, vol. 167, pp. 2554–2560, Jan. 2020, https://doi.org/10.1016/j.procs.2020.03.309.

[25] M. Elleuch, R. Maalej, and M. Kherallah, "A New Design Based-SVM of the CNN Classifier Architecture with Dropout for Offline Arabic Handwritten Recognition," *Procedia Computer Science*, vol. 80, pp. 1712–1723, Jan. 2016, https://doi.org/10.1016/j.procs.2016.05.512.

[26] Y. Lai, N. Wang, Y. Yang, and L. Lin, "Traffic Signs Recognition and Classification based on Deep Feature Learning," in *7th International Conference on Pattern Recognition Applications and Methods*, Funchal,

Portugal, Jan. 2018, pp. 622–629, https://doi.org/10.5220/0006718806220629.

[27] C. Kanan and G. W. Cottrell, "Color-to-Grayscale: Does the Method Matter in Image Recognition?," *PLOS ONE*, vol. 7, no. 1, Jan. 2012, Art. no. e29740, https://doi.org/10.1371/journal.pone.0029740.

[28] "R. Zhu, D. Yu, S. Ji, and M. Lu, "Matching RGB and Infrared Remote Sensing Images with Densely-Connected Convolutional Neural Networks," *Remote Sensing*, vol. 11, no. 23, Jan. 2019, Art. no. 2836, https://doi.org/10.3390/rs11232836.

[29] K. B. Shaik, P. Ganesan, V. Kalist, B. S. Sathish, and J. M. M. Jenitha, "Comparative Study of Skin Color Detection and Segmentation in HSV and YCbCr Color Space," *Procedia Computer Science*, vol. 57, pp. 41–48, Jan. 2015, https://doi.org/10.1016/j.procs.2015.07.362.

[30] Z. S. G. Tari, J. Shah, and H. Pien, "Extraction of Shape Skeletons from Grayscale Images," *Computer Vision and Image Understanding*, vol. 66, no. 2, pp. 133–146, May 1997, https://doi.org/10.1006/cviu.1997.0612.

[31] J. Basilio, G. Torres, G. Sanchez-Perez, L. K. T. Medina, H. Meana, and G. S. Perez, "Explicit Image Detection using YCbCr Space Color Model as Skin Detection," in *Applications of Mathematics and Computer Engineering*, 2011, pp. 123–128.

[32] S. Indolia, A. K. Goswami, S. P. Mishra, and P. Asopa, "Conceptual Understanding of Convolutional Neural Network- A Deep Learning Approach," *Procedia Computer Science*, vol. 132, pp. 679–688, Jan. 2018, https://doi.org/10.1016/j.procs.2018.05.069.

[33] K. Eckle and J. Schmidt-Hieber, "A comparison of deep networks with ReLU activation function and linear spline-type methods," *Neural Networks*, vol. 110, pp. 232–242, Feb. 2019, https://doi.org/10.1016/j.neunet.2018.11.005.

[34] S. Bruch, X. Wang, M. Bendersky, and M. Najork, "An Analysis of the Softmax Cross Entropy Loss for Learning-to-Rank with Binary Relevance," in *ACM SIGIR International Conference on Theory of Information Retrieval*, New York, NY, USA, Oct. 2015, pp. 75–78, https://doi.org/10.1145/3341981.3344221.

[35] Z. Zhang and M. R. Sabuncu, "Generalized Cross Entropy Loss for Training Deep Neural Networks with Noisy Labels," May 2018. Available: https://ui.adsabs.harvard.edu/abs/2018arXiv180507836Z.