IIETA International Information and Engineering Technology Association
Advancing the World of Information and Engineering

Revue d'Intelligence Artificielle
Vol. 35, No. 4, August, 2021, pp. 341-347
Journal homepage: http://iieta.org/journals/ria

Classification of Rigid and Non-Rigid Objects Using CNN

Aparna Gullapelly*, Barnali Gupta Banik

Research Scholar, Department of CSE, Koneru Lakshmaiah Education Foundation, Hyderabad, Telangana 500075, India

Corresponding Author Email: aparnag932012@gmail.com

https://doi.org/10.18280/ria.350409

Received: 26 July 2021
Accepted: 12 August 2021

Keywords:
KNN, Haarcascade, CNN, classification, rigid, non-rigid

ABSTRACT

Classifying moving objects in video surveillance can be difficult, and it is challenging to classify hard and soft objects with high Accuracy. Here rigid and non-rigid objects are limited to vehicles and people. CNN is used for the binary classification of rigid and non-rigid objects. A deep-learning system using convolutional neural networks was trained using python and categorized according to their appearance. The classification is supplemented by the use of a data set, which contains two classes of images that are both rigid and not rigid that differ by illuminations.

1. INTRODUCTION

Detection and classification of rigid and non-rigid objects with high Accuracy in video surveillance may be a challenging task after taking things like background, brightness, color, or environmental condition into consideration as the behavior of each rigid and non-rigid body are quite the opposite, it becomes tricky to train and let our CNN model know the difference between those objects. However, hopefully, CNN always acts as a game-changer in these classification tasks. One object is classified according to 4 features - color, shape and size, movement, and texture [1]. The detected object is categorized as human or non-human. It will be based on insights matching tactics. Insights are just layouts of descriptors with some characteristics. Previous dimension models for objects under different classes are kept separate to identify the segmented object. The shape is retrieved from the segmented object and compared with predefined model datasets under the various object classes.

In this article, we talked about the detection and classification of rigid and non-rigid object using CNN, and we formed a model and tested 200 images among 2000 images from a dataset. The Output of system is rectangular delimiter boxes, and it shows the category (class) information of an object if it is rigid or not rigid. We first had this complex problem of solving the classification scenario between rigid and non-rigid bodies. But the solutions we found for these problems are not accurate and are not efficient as well, and some of those are haarcascade and KNN classifiers and which stands as the reason for the conduct of this study and experiment using CNN which eventually leads to this article.

The present document is structured as follows. Section 1 of the work has a detailed introduction of our proposed system and explains the key concepts involved in our proposed system like CNN, rigid, non-rigid. Moreover, we were then followed by Section 2, which discusses some work that has already been done by some other people which has similarities to our proposed work with some identified disadvantages. Moreover, in Section 3, the discussion is all about our proposed solution for rigid and non-rigid objects classification. This part has a detailed walk through of our proposed algorithms with each execution step. Section 4 will have a detailed discussion regarding our CNN algorithms obtained results and performance and results at the train part, test part, and its overall Accuracy. Section-5 compares our proposed system accuracy with the related algorithms like KNN and Haarcascade, which eventually highlights our proposed system work. Section 6, the last section, speaks about the advantages, results, and improvements of the proposed work and points out the gaps of the previous related works we have tried to eliminate. The results that CNN showed are considerably efficient and much accurate than other algorithms with an accuracy of 92% at an epoch rate of 100 and we can confidently say that it can increase with increased dataset and enlarged epoch size.

2. RELATED WORK

Existing classification methods haven't progressed into their advanced form due to many issues such as low resolution, low quality, occlusions, and bad weather environment and illumination conditions. Before stepping into the proposed approach, let's compare the proposed approach with two more algorithms, some specialized proposed systems for object classification, and what imitations of these made us opt for this proposed system. We mention only a few of the most relevant articles in this area because our method focuses on classifying objects from an image [2]. The classification process is split into stages K. In each stage k; only one function is evaluated. The value of this feature contributes to confidentiality and alignment [3].

Engel proposed an off line approach to the detection and segmentation of non-rigid objects in video based on the analysis of 3-D medium offset [4]. In this work [5], a new method of detecting no rigid form is implemented by directly learning the relative distance in the shape space. No assumptions regarding the distribution of form or appearance are required in this. CNN has many benefits over traditional methods that can be summed up as follows. Detected and

classified objects that range from multiple instances. In terms of speed, CNN is approximately twice as fast. The efficiency of the system is measured in terms of the accurate detection that it is performing. More profound architecture offers an enhanced expressive ability than traditional shallow methods. The base architecture of the CNN is illustrated in the Figure 1. It contains the different layers used in the design of an NFC.CNN layers include Convolution, ReLU Layer, Max-pooling, Flattening, and fully connected layer. It is known that CNN is the most representative model of deep learning; Feature maps refer to the layers within CNN. Each layer is a 3D matrix, with pixel intensity being the color channel (e.g., RGB). The feature maps of the input layer are 3D Matrix. Each neuron is connected with two or more adjacent neurons from the previous layer, so the Feature Map of any hidden layer is a multi-channel image.

KNN classifier: KNN has been used in measurable assessment and example acknowledgment from now inside the 1970s as a non-parametric strategy.

Disadvantages:

- doesn't function admirably with enormous dataset.
- doesn't function admirably with high measurements.
- Need include scaling.
- Delicate to uproarious information, missing qualities, and Anomalies.

Table 1. KNN performance

Performance with different metrics	Actual data	Actual data After Imputation	After imputation and scaling
Accuracy	71.71	71.32	71.84
Error	28.29	28.68	28.16

Due to all the drawbacks and lazy performance of KNN-algorithms, it eventually stands out from our chosen list of classification algorithms and eventually due to its drawbacks. Its performance has fallen by limiting itself to a minimal accuracy, as shown in Table 1.

Haarcascade Classifier: It is an AI-based methodology where course work is prepared from many positive and negative pictures. It is then used to recognize objects in other images. Detecting a face from an image utilizing Haarcascade classifiers requires the resulting steps to be followed [6]. Haarcascade is one of those algorithms which can observe even the most minor point on an image and makes it helpful for the classification process. It detects and classifies real-time objects like vehicles and pedestrians using Haarcascade with background subtraction technique, but it failed to count the detected objects is tedious due to flickering [7].

Drawbacks

- The problem with this implementation was that it had been unable to detect side faces.
- Fails in detecting smaller objects.
- It will not work on low-quality images.
- It might not offer you the simplest Accuracy.

The Table 2 below shows the Accuracy of the haarcascade and its error rate on a mean when chosen for object classification. The Accuracy seems good when compared with the Accuracy and performance of KNN-algorithms but is still not the most effective one. As a whole, let us bring the downsides of the existing algorithms together.

- In KNN, the Output depends on the nearest neighbor, who eventually is not a good choice.
- It is more sensitive to distance metrics.
- Computation time and complexity are more with KNN.
- CNN's Accuracy depends on data quality on a majority basis.
- In KNN, if the K value is chosen in correctly, then that leads to either underfitting or overfitting of data.
- Haarcascades are more likely to be following a false-positive detection trend. Haarcascades are less accurate and fail in detecting objects with low image quality.

Table 2. Accuracy of the HaarCascade classifier

Performace with different metrics	Detection rate, %	False alarm rate, %	SW Execution Time, s
40 -Stage	87.6	13.5	18.1
22-Stage	75.2	15.9	24.7

3. PROPOSED SYSTEM

As per the downsides of the existing systems that we have discussed in the above section, our proposed system must resolve those drawbacks and add an advantage to our proposed implementation methodology. The statements below are advantages that our proposed system deals with.

- Automatic feature detection without any human interference
- High Accuracy even for the low quality of images
- The pooling process is an extra advantage that helps in dimensionality reduction.
- The dropout process helps in avoiding overfitting Augmentation methodology helps in an increased quantity of image sets, which help train the model with every possibility of instance.

CNN's are well suitable for large unstructured data because of their computation power. This proposed method has all the above methods being implemented in the CNN architecture of our proposed model solution. CNN has numerous benefits against customary strategies that can be summed up as follows. More profound engineering gives an expanded expressive ability than customary shallow methods. The essential design of the CNN appears in Figure 1. It contains the different layers that are utilized in the plan of a CNN. Convolution, ReLULayer, MaxPooling, Flattening, fully connected layers are the layers of CNN. CNN is the most delegate model of profound learning. Each layer of CNN is known as a component map. The element guide of the info layer is a 3D matrix. The grid pixel forces are distinctive shading channels (for example, RGB). The Feature guide of any hidden layer is a multi-channel picture, its 'pixel' can be seen as a particular feature. Features are separated from the prepared edges, and these removed highlights are applied for classification [8]. Each neuron is associated with a bit of part of adjoining neurons from the past layer; CNN is a broadly utilized neural organization design for PC vision-related assignments. CNN separates the highlights of pictures naturally; for example, significant highlights are recognized by the organization itself.

Convolutional Layer: The initial layer takes input image and performs different convolutional calculations between the input and the filter.

Pooling Layer: It is always important to make your model light weighted and make sure it does its work in a faster, efficient way, and there comes the pooling in hand for a user. It reduces the dimensionality of the feature map and thus helps in

reducing the parameters that are required in computing calculations for the network and Max pooling is such an example that does the process by selecting large elements in the selected section of the filter area as shown in Eq. (1).

$$\text{Output} = ((\text{input-kernel\_size}+2*\text{padding}) / (\text{stride})) +1 \quad (1)$$

Fully Connected Layer: In most Neural networks, this FCN is used as an output layer because it gives all the inputs to each activation unit of the next layer (most probably an output layer).

Dropout Layer: Sometimes, it is equally important to cut down something that is not useful in the process, and Drop out Layer does the same in a Neural Network by deactivating some neurons at a particular level of architecture by setting up the value of those in active neurons to 0. Moreover, one of the major reasons is to avoid over fitting the neural network for that purpose we will use Eq. (2).

$$y_i = \sum_j (wij, xj) \quad (2)$$

Activation Functions: An activation function is something that defines the Output for the particular node. It defines the values as Output as 0 or 1 and sometimes -1 to 1 based on the functionality that is shown in Eq. (3).

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \quad (3)$$

The Sequential type model is the most commonly used model type in python programming. A CNN model is built simply in Keras. It allows us to create a layer-by-layer model. The "add ()" function is used to add feature layers to the model.

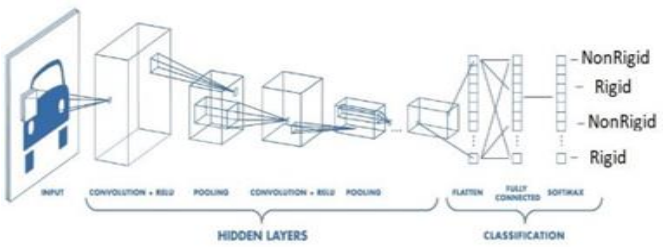


Figure 1. Basic architecture of CNN

3.1 CNN

As shown in the above Figure 1, A Convolutional Neural Network, like wise alluded to as CNN or ConvNet, is one among the classes of Neural Network designs that includes a specific standard of working and engineering to oblige advanced pictures.

The human mind measures immense amounts of information directly from the moment we see an image is in a similar way a CNN does its data stockpiling and handling; every neuron during a CNN measures information just in its responsive field. The layers are organized in such the most straightforward manner, so they identify less difficult examples first (lines, bends, and so on) and more mind-boggling designs (faces, objects, and so forth) further along. By utilizing a CNN, one can empower the site to PCs.

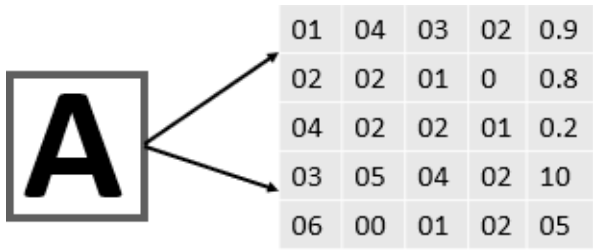


Figure 2. An example image is representing pixel value conversion

The above Figure 2 shows how a CNN design attempts to remember the data and interaction data of each pixel of the picture that is the RGB estimation of that picture at that particular pixel level. The convolution layer is the center structure square of the CNN. It conveys most of the organization's computational load. This layer plays out a speck item between two lattices, where one network is that the arrangement of learnable boundaries, in any case, is alluded to as a bit, and the other grid is that the limited bit of the responsive field. The portion is spatially more modest than an image however is more top to bottom. This suggests that if the picture comprises three (RGB) channels, the part tallness and width will be spatially little, yet the profundity stretches out up to all or any three channels. If we have a contribution of size W x W x D and D_{out} number of pieces with a spatial size of F with step S and measure of cushioning P, at that point, the resulting recipe might dictate the size of yield volume as shown in Eq. (4).

$$W_{out} = (W-F+2P)/+1 \quad (4)$$

Table 3 shows the architecture of implemented solution.

Table 3. Architecture of the implemented solution

Layer(type)	Output Shape	Param#
Conv2d (Conv2D)	(None, 148, 148, 32)	896
activation (Activation)	(None, 148, 148, 32)	0
max_pooling2d(MaxPooling2D)	(None, 74, 74, 32)	0
Conv2d _1(Conv2D)	(None, 72, 72, 32)	9248
activation_1(Activation)	(None, 72, 72, 32)	0
max_pooling2d-1(MaxPooling2D)	(None, 36, 36, 32)	0
Conv2d _2 (Conv2D)	(None, 34, 34, 64)	18496
activation_2(Activation)	(None, 34, 34, 64)	0
max_pooling2d- 2 (MaxPooling2D)	(None, 17, 17, 64)	0
Flatten (Flatten)	(None, 18496)	0
Dense (Dense)	(None, 64)	1183808
activation_3 (Activation)	(None, 64)	0
dropout (dropout)	(None, 64)	0
activation_4 (Activation)	(None, 64)	0
Total params: 47, 137		
Trainable params:47, 137		
Non-Trainable params:0		

Parameter Calculation in CNN:

$$(n * m * l + 1) * k \quad (5)$$

The above Eq. (5) is used to calculate the total number of parameters in CNN, where n* and m is the dimensions of the filter is the size of the input feature map, and k is the Output feature map. The above diagram indicates the complete architecture of CNN that has been employed in the proposed system, with different kinds of layers being included in between.

3.2 Process flow of implementation

The necessary libraries are imported first, and afterward, preparing information is given as input utilizing Google drive. Google Colab is an online reenactment device for python, and the Tensorflow algorithm was used. The algorithm gathers the data and gains from it in an exceptionally directed manner. The algorithm is alluded to as an administered classification algorithm. Then the data streams from different CNN layers and different activities are performed on the information. The learning rate and callbacks are defined. The number of epochs and cluster size are moreover characterized. Epochs are executed through which calculation is learned through the preparation data. Training exactness and preparing misfortune are continually monitored. If the preparation precision is beneath the limit, the callback work is conjured, and epochs are stopped. After accomplishing better precision, the Confusion lattice is plotted utilizing preparing and testing data. Various execution boundaries are to be characterized and noticed utilizing the array framework.

The below Figure 3 indicates the method we followed for implementation.

Step-1: Import all the desired libraries.
Step-2: Store the input data in the drive and mount it within the notebook file for training purposes.
Step-3: Data-preprocessing.
Step-4: Build the CNN model.
Step-5: Set a learning rate and epoch size.
Step-6: Run the model (compile).

To urge Accuracy, the dataset which is of 1000 images of which 500 belongs to rigid and the other 500 belongs to non-rigid and for better model input, it's better we should preprocess the entire dataset according to our model input capacity. The accompanying activities are to be administered inside the preprocessing phase. Resizing, Grayscale conversion, Noise evacuation and morphological operation (Dilation and Erosion) and improving precision are essential to have a huge dataset. We have been going for a picture expansion measure that improves the amount of our dataset for excellent preparation progress. The learning rate is one among the preeminent significant hyper boundaries to tune for preparing profound neural networks. The preparation should begin from a similarly tremendous learning rate because, in the first place, irregular loads are off from ideal, so the learning rate can diminish during the preparation to permit all the more fine-grained weight updates. If the learning rate is low, at that point, the preparation is more reliable. If the learning rate is high, then the preparation probably will not meet or might be separate. If the Accuracy is good enough, then plot the confusion matrix and save the model, else then change learning parameters and run the model again till you achieve the most effective or desired Accuracy. A neural network will be trained with all its training data once in an epoch. It uses everything at once, whether it is a forward pass or a reverse pass. Epochs are created of 1 or more batches.

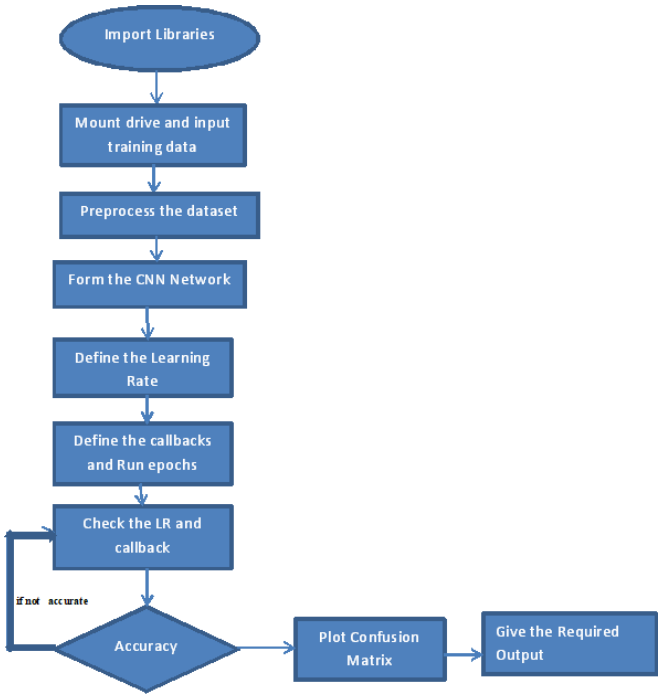


Figure 3. Flow chart of the object detection

A Callback could be a bunch of capacities to be applied at given phases of the preparation methodology. Use callback to ask a perspective on inside states and insights of the model during training. The most incessant order assessment metric that should be is 'Exactness (Accuracy). If the training model achieved better Accuracy, it would select a confusion matrix; otherwise, it rechecks and defines the callback and learning rate. The disarray network helps estimate Recall (likewise called Sensitivity), Precision, Specificity, Accuracy, and, above all, the AUC-ROC Curve.

3.3 Data analysis

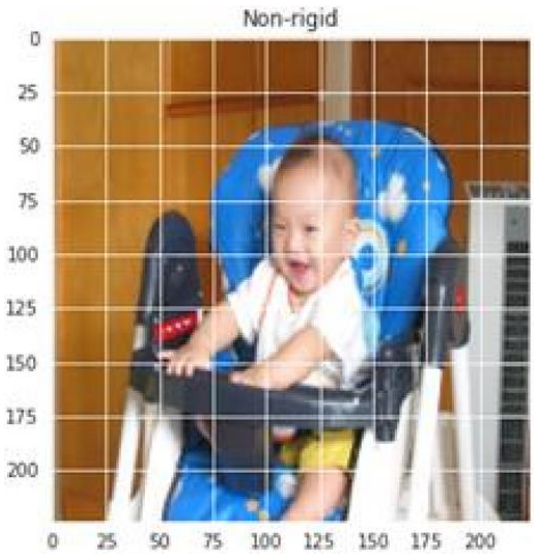


Figure 4. Graphical_representation_of data sample

Let's analyze and understand the actual data distribution of our dataset as per the data mentioned in Table 4.

Dataset Specification:

- Number of classes: 2[Rigid, Non-Rigid]
- All out number of input images: 2000
- Preparing picture: 1800
- Testing picture: 200
- Rigid images: 1000(900+100)
- Non-Rigid: 1000(900+100)

Class Distribution of Dataset:

Table 4. Distribution of dataset

Class distribution	Class 0	Class 1
Type of Object	Non-Rigid	Rigid
No.of Object type	1000(900+100)	1000(900+100)

The above Figure 4 of a baby is an example of how images are dimensionally included and to show up their quality.

4. RESULTS AND PERFORMANCE

Execution of the classification frameworks is generally assessed utilizing the information in the disarray network. A disarray grid for four results is given below. How the four arrangement measurements are determined (TP, FP, FN, TN) and our anticipated worth contrasted with the actual worth in a disarray grid are introduced in the accompanying disarray framework table. The following equations are used to calculate 4 key classification metrics as shown in Table 5.

And when it comes to Accuracy, Accuracy= (TP+TN)/N is the parameter of measurement that we take into consideration to measure the actual Accuracy of a deep learning model. Now, let us see what we have achieved in our proposed system of implementation.

Table 5. Classification metrics

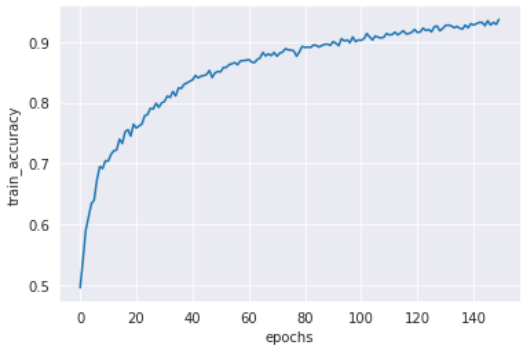
Accuracy Acc=(TP+TN/(TP+TN+FP+FN)	Recall Rec=TP/(TP+FN)
Precision Pre=TP/(TP+FP)	F1-score F1=2/(1/Rec)+(1/Pre)

Input and Output using CNN architecture

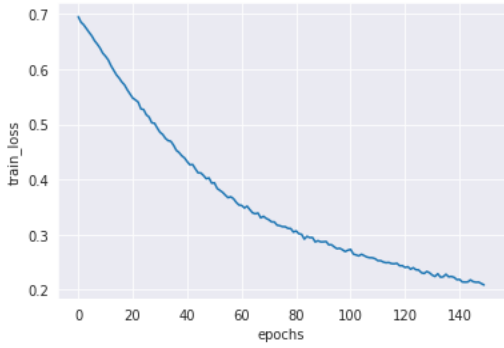
The below Figure 5 and Figure 6 indicate the input passed to the CNN architecture and output that we have got from it.

Figure 5. Input Passed to our CNN model

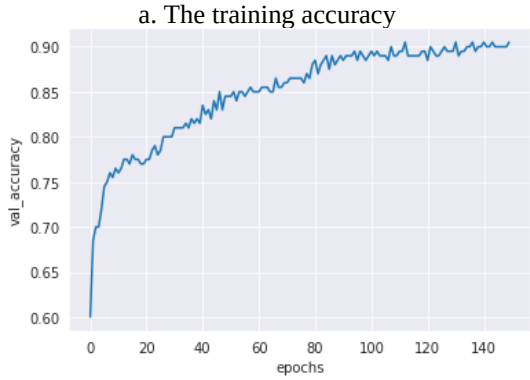
Figure 6. Output given by our CNN model



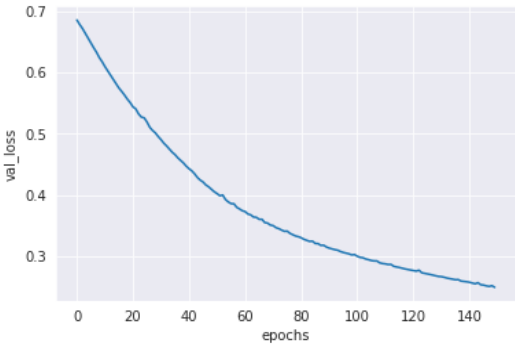
a. The training accuracy



b. The training loss



c. The validation accuracy



d. The validation loss

Figure 7. Accuracy and loss of training and validation

```
57/57 [==============================] - 115s 2s/step - loss: 0.2840 - accuracy: 0.8958 - val_loss: 0.2992 - val_accuracy: 0.9100
Epoch 92/100
57/57 [==============================] - 114s 2s/step - loss: 0.2673 - accuracy: 0.9097 - val_loss: 0.2978 - val_accuracy: 0.9100
Epoch 93/100
57/57 [==============================] - 114s 2s/step - loss: 0.2821 - accuracy: 0.8919 - val_loss: 0.2968 - val_accuracy: 0.9000
Epoch 94/100
57/57 [==============================] - 114s 2s/step - loss: 0.2688 - accuracy: 0.9054 - val_loss: 0.2951 - val_accuracy: 0.9050
Epoch 95/100
57/57 [==============================] - 114s 2s/step - loss: 0.2716 - accuracy: 0.9099 - val_loss: 0.2935 - val_accuracy: 0.9050
Epoch 96/100
57/57 [==============================] - 115s 2s/step - loss: 0.2641 - accuracy: 0.9147 - val_loss: 0.2935 - val_accuracy: 0.9000
Epoch 97/100
57/57 [==============================] - 114s 2s/step - loss: 0.2771 - accuracy: 0.8978 - val_loss: 0.2912 - val_accuracy: 0.9050
Epoch 98/100
57/57 [==============================] - 115s 2s/step - loss: 0.2456 - accuracy: 0.9215 - val_loss: 0.2907 - val_accuracy: 0.8950
Epoch 99/100
57/57 [==============================] - 115s 2s/step - loss: 0.2487 - accuracy: 0.9171 - val_loss: 0.2906 - val_accuracy: 0.9000
Epoch 100/100
57/57 [==============================] - 115s 2s/step - loss: 0.2512 - accuracy: 0.9203 - val_loss: 0.2873 - val_accuracy: 0.9100
```

Figure 8. Accuracy of the both sets training and validation

```
Epoch 138/150
57/57 [==============================] - 109s 2s/step - loss: 0.2297 - accuracy: 0.9222 - val_loss: 0.2589 - val_accuracy: 0.8950
Epoch 139/150
57/57 [==============================] - 109s 2s/step - loss: 0.2279 - accuracy: 0.9160 - val_loss: 0.2584 - val_accuracy: 0.9000
Epoch 140/150
57/57 [==============================] - 109s 2s/step - loss: 0.2225 - accuracy: 0.9324 - val_loss: 0.2577 - val_accuracy: 0.9000
Epoch 141/150
57/57 [==============================] - 109s 2s/step - loss: 0.2118 - accuracy: 0.9308 - val_loss: 0.2570 - val_accuracy: 0.9050
Epoch 142/150
57/57 [==============================] - 109s 2s/step - loss: 0.2090 - accuracy: 0.9355 - val_loss: 0.2557 - val_accuracy: 0.9000
Epoch 143/150
57/57 [==============================] - 109s 2s/step - loss: 0.2110 - accuracy: 0.9320 - val_loss: 0.2544 - val_accuracy: 0.9000
Epoch 144/150
57/57 [==============================] - 109s 2s/step - loss: 0.2126 - accuracy: 0.9349 - val_loss: 0.2562 - val_accuracy: 0.9050
Epoch 145/150
57/57 [==============================] - 109s 2s/step - loss: 0.2040 - accuracy: 0.9350 - val_loss: 0.2529 - val_accuracy: 0.9000
Epoch 146/150
57/57 [==============================] - 109s 2s/step - loss: 0.2220 - accuracy: 0.9334 - val_loss: 0.2524 - val_accuracy: 0.9000
Epoch 147/150
57/57 [==============================] - 109s 2s/step - loss: 0.2147 - accuracy: 0.9250 - val_loss: 0.2510 - val_accuracy: 0.9000
Epoch 148/150
57/57 [==============================] - 109s 2s/step - loss: 0.2091 - accuracy: 0.9387 - val_loss: 0.2501 - val_accuracy: 0.9000
Epoch 149/150
57/57 [==============================] - 109s 2s/step - loss: 0.2153 - accuracy: 0.9246 - val_loss: 0.2513 - val_accuracy: 0.9000
Epoch 150/150
57/57 [==============================] - 109s 2s/step - loss: 0.2073 - accuracy: 0.9297 - val_loss: 0.2486 - val_accuracy: 0.9050
```

Figure 9. Accuracy improvement with change in epoch value

The training accuracy is shown in Figure 7a. Training loss is shown in Figure 7b, where the accuracy has been gradually increasing, and loss has been falling down the minimal state, which is very much considerable the validation accuracy which is shown in Figure 7c and validation loss is shown 7d. Here, the array contains the sequence of True positive (89), True Negative (92), False Positive (8), False-Negative (9).

The above Figure 8 shows the Accuracy for both the training set and validation set that we have got after 100 epochs of each with 57 as batch size. The performance metrics of our CNN model is shown in Table 6.

Table 6. Performance metrics of CNN model

	Precision	Recall	f1-score	Support
Non-rigid (Class 0)	0.92	0.89	0.9	100
Rigid (Class 1)	0.89	0.92	0.91	100
Accuracy			0.91	200
Macro avg	0.91	0.91	0.9	200
Weighted avg	0.91	0.91	0.9	200

Table 7. Performance matrix of proposed CNN model

	Precision	Recall	f1-score	Support
Non-rigid (Class 0)	0.91	0.91	0.91	100
Rigid (Class 1)	0.91	0.91	0.91	100
Accuracy			0.91	200
Macro avg	0.91	0.91	0.91	200
Weighted avg	0.91	0.91	0.91	200

Compared with the above results section, we have just trained our model with 150 epochs that is shown in Figure 9, but we have given the epoch parameter as 150, which eventually has brought us this improved accuracy as shown in Table 7.

Also, through the comparison with the results above, we can observe the prediction accuracy in the above confusion matrix and see what made us achieve that improved performance.

```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_val,predictions)
```

```
array([[91,  9],
       [ 9, 91]])
```

The above piece of code indicating the confusion matrix results we have obtained for our validation set containing 100 images for both Non-rigid and Rigid individually.

Compared with the above results section, we have just trained our model with 150 epochs, but we have given the epoch parameter as 150, which eventually has brought us this improved accuracy as shown in Figure 9.

Table 8 shows the Confusion metrics after 150 epochs.

Table 8. Confusion Matrix on validation set after 150 epochs

Predicted	**Non-Rigid**	**Rigid**	**Total**
Non-Rigid	91	9	100
Rigid	9	91	100

5. COMPARATIVE STUDY

As we have discussed the other two algorithms. Let us compare and see why our algorithm is much better and considerable as shown in Table 9.

Table 9. Performance evaluation of three classifiers

Classifier	Accuracy
CNN	0.91
Haarcascade	0. 87
KNN	0.671

Our CNN performance stands out from the other 2 algorithms, and we can even improve our model's Accuracy to 97 or 98 by including an extra-large dataset or increasing learning rate and epoch size or both at a time.

6. CONCLUSION

This identification is valuable for assessing the direction of the moving vehicles and different Objects. Further work will be centered on improving the recognition and classification performance. Once the classification is accomplished, further work can be carried out the individual distinguishing proof of unbending (Rigid) and non-rigid articles like whether an inflexible item is a car, bus, or bike. Moreover, the bending (non-Rigid) item individual identification whether an individual is male or female. We want to make a dataset and retrain the organization with this dataset. And at the same time there are short span of drawbacks that this particular algorithm can exhibit and they are like as the dataset quantity is quite less, we can make efforts to gather some extra dataset quantity which might increase the accuracy and insight gathering capacity of the model and at the same time exhibition can be additionally improved by utilizing a much more extensive model like GoogleNet or transfer learning can also be applied for further efficiency and accuracy.

REFERENCES

[1] Ashwini, B., Yuvaraju, B. (2017). Application of machine learning approach in detection and classification of cars of an image. International Journal of Signal and Imaging Systems Engineering, 10(1/2): 8. https://doi.org/10.1504/ijsise.2017.10005425

[2] Bharath, R.R., Dhivya, G. (2014). Moving object detection, classification and its parametric evaluation. International Conference on Information Communication and Embedded Systems (ICICES2014), pp. 1-6. https://doi.org/10.1109/icices.2014.7033891

[3] Biswas, P., Prabhakar, G., Rajesh, J., Pandit, K., Halder, A. (2017). Improving eye gaze controlled car dashboard using simulated annealing. Proceedings of the 31st International BCS Human Computer Interaction Conference (HCI). https://doi.org/10.14236/ewic/hci2017.39

[4] Engel, J. (2005). "Equipped for Murder": The Paxton boys and "the spirit of killing all Indians" in Pennsylvania, 1763-1764. Rhetoric & Public Affairs, 8(3): 355-381. https://doi.org/10.1353/rap.2005.0053

[5] Gautam, A., Singh, S. (2019). Trends in video object tracking in surveillance: A survey. 2019 Third International conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), pp. 729-733. https://doi.org/10.1109/I-SMAC47947.2019.9032529

[6] Jang, H., Yang, H.J., Jeong, D.S., Lee, H. (2015). Object classification using CNN for video traffic detection system. 2015 21st Korea-Japan Joint Workshop on Frontiers of Computer Vision (FCV), pp. 1-4. https://doi.org/10.1109/fcv.2015.7103755

[7] Minarno, A.E., Setiawan Sumadi, F.D., Wibowo, H., Munarko, Y. (2020). Classification of batik patterns using k-nearest neighbor and support vector machine. Bulletin of Electrical Engineering and Informatics, 9(3): 1260-1267. https://doi.org/10.11591/eei.v9i3.1971

[8] Nannia, L., Ghidoni, S., Brahnam, S. (2020). Ensemble of convolutional neural networks for bioimage classification. Applied Computing and Informatics, 17(1): 19-35. https://doi.org/10.1016/j.aci.2018.06.002