Vehicle Crash Detection System

*Convolutional Neural Networks and Deep Learning

Jayanth Prathipati
Artificial Intelligence Branch
National Institute of Technology Karnataka
Surathkal, Karnataka
jayanthprathipati2003@gmail.com

Shashank Reddy Muppidi
Artificial Intelligence Branch
National Institute of Technology Karnataka
Surahtkal, Karnataka
shashanknitkai@gmail.com

***Abstract*—As the number of vehicle users continues to grow, it is increasingly important to have effective systems in place for identifying accidents in a timely manner. To address this problem, we propose using a deep learning model that utilizes computer vision and convolutional neural networks (CNN) to analyze image frames from vehicles and identify crash events. The CNN model processes the image frames and produces output indicating the location, impact, and severity of any damage. This approach has the potential to greatly improve the ability to quickly and accurately identify accidents, potentially saving lives.**

***Index Terms*—Deep Learning,Convolutional Neural Network,Hyper parameter Optimisation, Evolutionary Algorithm,Joint Optimisation for multiple convolutional networks,Genetic Algorithm Optimisation,,dash Camera.**

I. Introduction

As standard of living and economic development increases, the incidence of road accidents in our country is rising, causing significant loss of life and property. Traffic safety has become a major concern for the nation. One of the major causes of traffic fatalities is poor emergency response. Recognizing and classifying objects and environments is a task that humans are able to perform quickly and accurately, and this skill is developed from a young age. Similarly, computers can be trained to recognize and classify objects and environments by analyzing low-level features such as edges and curves using a technique called convolutional neural network (CNN). This approach, known as image recognition and classification, has the potential to improve emergency response and ultimately reduce traffic fatalities.

The focus of our project is to improve the precision of a basic convolutional neural network (CNN) structure by incorporating additional features. Our objectives for this paper are two-fold: first, we propose an efficient hyperparameter optimization strategy for the CNN network structure based on an evolutionary algorithm. Second, we extend this approach by using a committee of multiple CNNs. Hyperparameters, such as layer and kernel sizes, play an important role in determining the performance of a CNN. Hyperparameter optimization is the process of finding best set of hyperparameters that can provide

Identify applicable funding agency here. If none, delete this.

an accurate model in a reasonable time. In this work, we focus on optimizing the hyperparameters that describe the structure of a CNN. The goal is to identify a good model by optimizing a loss function over a graph-structured configuration space.

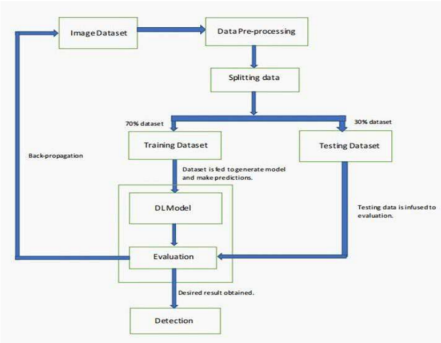

Fig. 1. Representation of Deep Learning for Image Dataset

II. Related Works

A. Search Optimisation Method

Grid search is a method used to find the best set of hyperparameters for a machine learning model. It works by

specifying a range of values for each hyperparameter, then testing all possible combinations of these values to find the set that gives the best performance. The advantage of using a grid search is that it can be easily parallelized, which means that multiple combinations of hyperparameters can be tested at the same time. However, if one combination of hyperparameters fails, it can cause all of the other jobs to fail as well.

One of the limitation of using grid search is that it becomes impractical as the number of hyperparameters increases. This is because the number of functions that need to be evaluated increases with each added hyperparameter. To overcome this limitation, machine learning practitioners often use a limited grid and then expand it, which makes the grid more efficient to configure the best set of hyperparameters while continually searching for new ones.

Another Recent Optimisation is, Bayesian optimization, a method for tuning hyperparameters in machine learning models. It uses a technique called Gaussian process modeling, which models the uncertainty of a function based on a set of observations. The idea is to find the set of hyperparameters that will give the best performance by using a probabilistic model to predict the performance of different sets of hyperparameters. The algorithm uses this model to select the next set of hyperparameters to test, by choosing the set that is most likely to improve the current best performance. However, this method can be computationally expensive when dealing with high-dimensional hyperparameters, which can limit its performance and applicability.

B. Genetic Algorithm

Genetic algorithms and evolutionary algorithms are optimization techniques that are inspired by the principles of natural selection and evolution.They both use a population-based approach to search for the optimal solution to a given problem.

Genetic algorithms use a binary representation of individuals, which makes it easy to implement the genetic operations of mutation and crossover. These operations allow for the generation of new candidate solutions that can be outside of the search space.

On the other hand, evolutionary algorithms use customized data structures, and the implementation of genetic operations depends on the problem at hand.

Genetic algorithms are useful when there is limited information about the gradient function and the optimization problem has multiple local minima or maxima. They can be parallelized easily, which allows for the exploration of multiple paths to the optimal solution simultaneously.

Additionally, they have the advantage of not getting trapped in suboptimal local maxima or minima, unlike local methods which are limited to the neighborhood of the current solution.

III. Methods

A. Hyperparameters Optimisation

Hyperparameters are parameters that are not learned from the data during the training process of a machine learning model, but are set before training. They are used to control the behavior of the model and are often chosen through a process called hyperparameter tuning.Hyperparameters can be divided into two types:

Structural hyperparameters: These are parameters that determine the overall architecture of the model, such as number of layers in a neural network or number of trees in a random forest.

Operational hyperparameters: These are parameters that control the training process, such as the learning rate or the number of iterations.

The process of finding the optimal values for these hyperparameters is known as hyperparameter tuning or hyperparameter optimization, and it is an important step in the machine learning workflow. Hyperparameter that determines the network structure such as:

- Kernel Size is the size of the filter.
- Kernel Type indicates values of the actual filter such as edge detection and sharpening.
- Activation functions allows the model to learn nonlinear prediction boundaries.
- Learning rate regulates the update of weight at the end of each batch.
- A number of epochs–the iterations of the entire training dataset to the network during training.
- Stride is the rate at which the kernel passes over input image.
- Padding adds layers of 0s to make sure the kernel to pass over the edge of images.
- Hidden layer are the layers that are hidden between input and output layers.

TABLE III. Network Structure Hyperparameters

Hyperparameter	Abbreviation	Range
Learning rate	learning_rate	[0.001, 0.003, 0.002,0.0015]
Batch Size	batch_size	[64]
Momentum	momentum	[0.9, 0.95, 0.99]
Optimizer	optimizer	['adam']
Number of Filters	filters1	[64, 96]
Kernal Size	ksize1	[4, 5]
Number of Filters	filters2	[96, 128]
Kernal Size	ksize2	[4, 5]
Number of Filters	filter3	[96, 128]
Kernal Size	ksize3	[4, 5]
Hidden Layer	full_hidd1	[100, 125]
Hidden Layer	full_hidd2	[100, 125]
Activation	activation	['lrelu']

A particular model can have more than 10 hyperparameters and finding the best combination is the search problem.The right choice of hyperparameter values can affect the performance of the model.

Optimization can be explained in a simple way in which given function that accepts inputs and returns numerical output, how can it efficiently find the inputs, or parameters, that maximize the functions output?. Hyper paramter Optimisation is formally defined as :

- $\mathcal{A}$ machine learning algorithm λ is a mapping $\mathcal{A}_\lambda$
- $\mathcal{D} \rightarrow \mathcal{M}$ where $\mathcal{D}$ is the set of all datasets and $\mathcal{M}$ is the space of all models
- $\lambda \in \Lambda$ is the chosen hyperparameter configuration with $\Lambda = \Lambda_1 \times \cdots \times \Lambda_P$ being the is P-dimensional hyperparameter space.
- The learning algorithm estimates a model $M_\lambda \in \mathcal{M}$ that minimizes a regularized loss function $\mathcal{L}$ (e.g. misclassification rate):

$$\mathcal{A}_\lambda\left(\mathcal{D}^{(train)}\right) := \underset{\mathcal{M}_\lambda \in \mathcal{M}}{\arg\min} \mathcal{L}\left(\mathcal{M}_\lambda, \mathcal{D}^{(train)}\right) + \mathcal{R}\left(\mathcal{M}_\lambda, \lambda\right)$$

Our task of hyper-parameter optimization is to find the optimal hyper-parameter configuration lambda using a validation set,

$$\lambda^* := \underset{\lambda \in \Lambda}{\arg\min}\, \mathcal{L}\, \underbrace{\left(\mathcal{A}_\lambda, \mathcal{D}^{(train)}, \mathcal{D}^{(valid)}\right)}_{=: \mathcal{F}_\mathcal{D}(\lambda)}$$

B. Hyper-parameter optimisation using Evolutionary Algorithm

Our goal is to find an optimal Convolutional Neural Network structure which will be encoded as a set of hyper-parameters. Therefore hyper-parameter can be defined as h = hconv, hf c as the set of structural parameters composed by hconv, the parameters of the convolutional layers, and hf c, the parameters of the fully connected layers. The convolutional layer parameter set is given by where li = (kcount, ksize) refers

$$\vec{\mathbf{h}}_{conv} = \{\mathbf{l}_0, \ldots, \mathbf{l}_{n-1},\}$$

to the configuration tuple of the i th layer, i.e. kcount number of kernels and ksize kernel size

The solution that we infer is discrete and the error function is discontinuous and non differentiable, so traditional optimization approaches such as gradient-decent are not applicable in this case. Therefore we propose to use evolutionary algorithms (EAs) Evolutionary algorithms (EAs) are a type of optimization method that are inspired by the process of natural selection. They work by creating population of potential solutions to a problem, and then repeatedly applying genetic operators (such as crossover and mutation) to the population to generate new, improved solutions. The best solutions are then selected to move on to the next generation, while the weaker solutions are discarded. This process is repeated until a satisfactory solution is found or a stopping criterion is met. EAs are particularly useful for solving problems where the solution space is discrete and the error function is not continuous or differentiable, as they are able to navigate this complex and rugged landscape effectively. Furthermore, EAs can avoid getting stuck in local minima by simultaneously exploring different parts of the solution space.

In this paper,we are using evolutionary algorithms to optimize the hyperparameters of a neural network.We define an ”individual” by the hyperparameters h, which include the layers and their characteristics (such as the number of filters or the size of the fully connected layers). To narrow down the solution space and make the optimization process more efficient, they impose a specific ordering on the layers: the convolutional layers are sorted first based on their kernel size and then based on the number of filters, and the fully connected layers are sorted based on their size. This ordering scheme is intended to help focus the search on more plausible network configurations and exclude ones that are less likely to be optimal. By doing this the researchers are trying to limit the solution space, so that the algorithm can explore more effectively. Here lambda denotes the number of offspring

$$P^t = \{\vec{\mathbf{h}}_0, \ldots, \mathbf{h}_{\mu-1}\}.$$

individuals evolved by mutation and crossover. The lambda off springs are equally generated by applying crossover and mutation to a set of randomly sampled individuals from the parent population. We apply a one-point crossover method which exchanges all genes after a randomly selected position in the genome of two individuals. The mutation contains two operations: the variation and the elimination or creation of a gene. The variation is implemented by the polynomial bounded mutation function of the NSGA-II algorithm. The value x of a gene is updated to x 0 with a certain probability pm as follows:

$$x' = \lfloor \min(\max(x + \delta_q(x_u - x_l), x_l), x_u) \rfloor \quad (1)$$

with

$$\delta_q = \begin{cases} 2\delta_1 - 1 & , u < 0.5 \\ 1 - 2\delta_2 & , u \geq 0.5 \end{cases}$$

$$\delta_1 = \left(u + (0.5 - u)\left(1 - \frac{x_i - x_l}{x_u - x_l}\right)^{\eta+1}\right)^{\frac{1}{\eta+1}}$$

$$\delta_2 = \left(1 - u + (u - 0.5)\left(1 - \frac{x_u - x_i}{x_u - x_l}\right)^{\eta+1}\right)^{\frac{1}{\eta+1}}$$

In the work being described, the evolutionary algorithm uses two types of operators: elimination and creation. These operators are used to modify the hyperparameters of an individual, with the goal of improving its performance. The elimination operator removes a layer or a part of it, while the creation operator adds a new layer or increases the size of an existing one. These operators are applied with a probability of pm and are subject to constraints, such as the maximum and minimum allowed sizes for the layers. If the hyperparameters of an individual are not modified by the operators, the weights of the CNN will not be retrained. In other words, if the structure of the network does not change, the algorithm does not need

to spend computational resources to retrain it. This helps to save computational resources and time.

IV. Convolution Neural Network

Convolutional Neural Networks (CNNs) are a type of neural network that are particularly well-suited for processing data that has a grid-like structure, such as images. In an image, the neurons are arranged in three dimensions: width, height, and depth (also referred as channels). For example, in the CIFAR-10 dataset, the images have a volume of 32x32x3 (width, height, and depth).

A traditional neural network is composed of input layer, one or more hidden layers, and output layer. CNNs, on the other hand, arrange the neurons into three dimensions of width, height, and depth, and each layer will transform the 3D input to a 3D output volume of neuron activations. In this case, the input layer holds the image, its dimensions are width and height, and the depth is the three channels (red, green and blue).

CNNs architectures have three main types of layers: Convolutional Layer, Pooling Layer, and Fully Connected Layer. The convolutional layer applies a set of filters to the input, which are used to extract features from the image. Pooling layers are used to reduce the spatial dimensions of the image, which helps to reduce the computational cost and prevent overfitting. Fully connected layers are used to classify the image based on the features extracted by the convolutional layers and the pooling layers. These layers help to improve the performance of the CNNs in image classification tasks.

- RELU(Rectified Linear Unit) is an activation function that converts all negative pixel values to 0.
- INPUT will hold on the raw pixel value of images.
- CONV will compute output of neurons.
- POOL will pass over sections of the image and pool them into the highest value in the section.
- FC (fully-connected) layer will calculate the class, and finally, each neuron will be connected to all number in the previous volum.

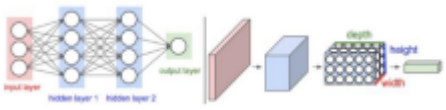

Fig. 1. A regular 3-Layer Neural Network vs CNN

A. Layers in Convolutional Neural Networks

1.Convolutional Layer:

A convolutional layer in a convolutional neural network (CNN) is used to extract features from an input image. It does this by performing a mathematical operation called convolution, where a small matrix called a filter is moved across the image, taking the dot product between the filter and the parts of the image that it overlaps. This process is repeated to create a feature map, which contains information about different aspects of the image, such as edges, corners, and lines. The feature map is then passed to subsequent layers in the network to extract more advanced features. The convolutional layer is beneficial in CNNs because it preserves the spatial relationship between the pixels in the image, allowing the network to learn more meaningful features.

2.Pooling Layer:

A pooling layer in a convolutional neural network (CNN) is used to reduce size of feature maps produced by convolutional layer. This is done to reduce computational costs and improve the efficiency of the network. There are several types of pooling operations, such as max pooling, average pooling, and sum pooling.

Max pooling selects the largest element from a section of feature map, while average pooling calculates average of the elements in that section. Sum pooling calculates the total sum of the elements in the section. These pooling operations reduce the feature map by summarizing features extracted by the convolutional layer.

The pooling layer serves as a bridge between the convolutional layer and the fully connected layer, which is the last layer in the CNN. It helps the network to recognize features independently and generalize the features extracted from the convolutional layer. By reducing the size of the feature maps, the pooling layer also reduces the number of computations required in the network, making it more efficient.

3. Fully Connected Layer:

The fully connected (FC) layer is the last layer in a convolutional neural network (CNN) and it is used to connect the neurons between different layers. This layer is composed of weights, biases, and neurons. The input image from the previous layers is flattened and then fed to the fully connected layer.

In this layer, the flattened vector of all pixels undergoes several more FC layers where mathematical operations take place. This is where the classification process begins. The purpose of using multiple fully connected layers is that it can improve the performance of the network compared to using a single fully connected layer. The multiple layers allow the network to extract more complex and abstract features from the input image and increase the accuracy of the network.

B. Using Pyramid Pooling Module

The problem with traditional CNN layers is that they cannot always incorporate enough global context information to accurately identify high-density objects. To address this issue, a method called PPM (Pyramid Pooling Module) is used. PPM is a global and subregional context module that allows CNN to be more robust to changes in scale. It does this by combining features from different levels of resolution, called pyramid scales.

The four pyramid scales used in PPM are 1x1, 2x2, 3x3 and 6x6. These scales allow the network to extract information from different levels of detail in the image. By combining these features, PPM allows the network to better understand the overall context of the image and improve its ability to identify high-density objects. In summary, PPM is a module

that allows CNN to incorporate both global context information and sub-regional context information to extract features from an input image and make the network more robust to changes in scale.

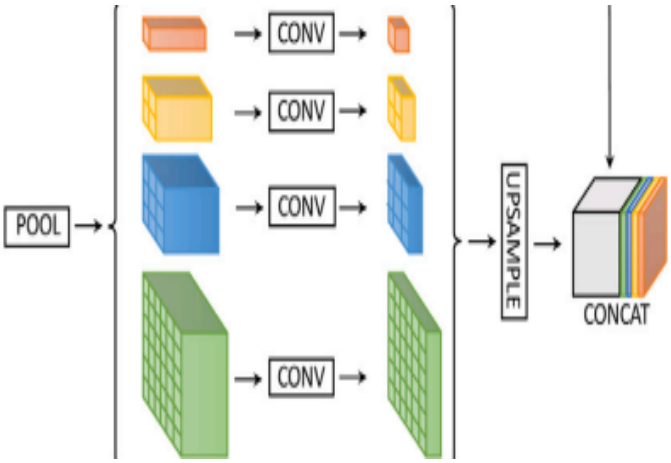

Fig. 2. Pyramid Pooling Module

The highest level, shown in orange, applies a global max pooling operation to create a 1x1 feature map that describes complete context of the image, such as the number of detected objects. The other levels divide the input map into subregions, creating a grouped representation of the image with their subcontext information such as dense or sparse regions.

Each level of the PPM contains feature maps with various sizes. To ensure that these feature maps can be combined effectively, a 1x1 convolution layer with 512 filters is used after each level. The feature maps are then upsampled to the same size as the input map using bilinear interpolation. Finally, these feature maps are concatenated with input map to form an improved description of the image. This step ensures that small object information is not lost in the PPM phase and that necessary information is present in the pooled image.

The PPM module was proposed for semantic segmentation, but it has also been shown to be effective in counting objects in experiments. By allowing information at different scales and its global context to be grouped with the feature map, the PPM improves the detection performance of the CNN. In summary, the PPM is a module that combines features from different levels of resolution and global context information to improve the CNN performance in identifying high-density objects.

V. Joint Optimsation of Multiple CNN

A committee of multiple CNNs is being used to improve classification performance. The classification is done by combining the scores from multiple trained CNNs, and using the average of these scores. Ideally, the classification errors made by each of the individual CNNs in the committee should not be related to each other, so that the overall average error is minimized. A novel fitness function has been proposed which considers the overall classification error of the group of CNNs in order to achieve this goal.We propose a novel

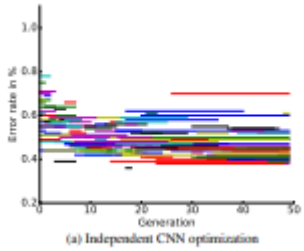

Fig. 3. Distrubution of individuals during Hyper paramters optimisation Independent CNN Optimisation (Error rate is represented by the line

fitness function that takes the global classification error into account:

$$f(\mathbf{h}_m) = \sum_{j}^{N} e_j(\mathbf{h}_m) \cdot \left(\sum_{i}^{\mu+\lambda} e_j(\mathbf{h}_i) \right)^{k}, \ \mathbf{h}_m, \mathbf{h}_i \in P^{t}_{\mu+\lambda}$$

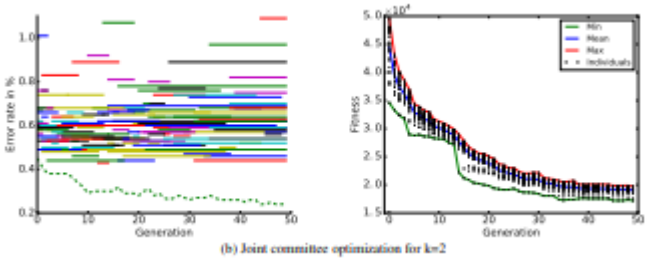

Fig. 4. Distrubution of individuals during Hyper paramters optimisation Joint Committee optimisation(Error rate is represented by the line

VI. Experiments and Results

A. Data Preprocessing

The data preprocessing phase helps to improve the model efficiency.In simpler terms, the data preprocessing phase is a step taken to prepare the images for use in a machine learning model. The images are converted into a 3D matrix with an RGB format and then grouped together in an array. The pixel values of the images are then scaled down by dividing them by 255, which results in pixel values that are between 0 and 1. The dataset used for training and testing the model contains a total of 60000 images, each of which has a resolution of 32x32 pixels. Since the images are in RGB format, the input shape of the images is (32,32,3) which means 32 pixels wide, 32 pixels high and 3 channels (R,G,B).

B. Approach

The design of the Convolutional Neural Network consists of three parts mainly model construction, model training, and model testing.

First Architectire that contains 3 Conv layers with max pooling in the first two layers, and two Fully Connected layers.

Table 1 : Configuration of the Arch1

Layer name	Kernel size	Output size	Number of parameters	Number of FP operations (MFLOPS)
Conv1	3x3	32x32x32	896	1.769504
MaxPool1	2x2	16x16x32		
Conv2	3x3	16x16x64	18496	9.437248
MaxPool2	2x2	8x8x64		
Conv3	3x3	8x8x64	36928	4.718656
Fc1		128	524416	1.048704
Fc2		10	1290	0.00257
Total			**582 026**	**16.976682**

Second Architecture architecture that contains 8 Conv layers containing 4 max-pooling layers, and 3 Fully Connected layers and added 2 high-level (256 filters) convolution layer to the second Architecture.

Table 2 : Configuration of the Arch2

Layer name	Kernel size	Output size	Number of parameters	Number of FP operations (MFLOPS)
Conv1	3x3	32x32x32	896	1.769504
Conv2	3x3	32x32x32	9248	18.8744
MaxPool1	2x2	16x16x32		
Conv3	3x3	16x16x64	18496	9.437248
Conv4	3x3	16x16x64	36928	18.874432
MaxPool2	2x2	8x8x64		
Conv5	3x3	8x8x128	73856	9.437312
Conv6	3x3	8x8x128	147584	18.874496
MaxPool3	2x2	4x4x128		
Fc1		2048	2098176	4.195328
Fc2		512	524800	1.049088
Fc3		10	5130	0.01025
Total			**2915114**	**82.522058**

Third Architecture the model achieved 78.81 validation accuracy in 34 epochs. Based on the three Different Architectures we can get an optimized of hyperparamter values. The Output set of values are given below:

Evaluating the parameters such as accuracy,precision,recall,f1:

Table 3 : Configuration of the Arch3

Layer name	Kernel size	Output size	Number of parameters	Number of FP operations (MFLOPS)
Conv1	3x3	32x32x32	896	1.769504
Conv2	3x3	32x32x32	9248	18.8744
MaxPool1	2x2	16x16x32		
Conv3	3x3	16x16x64	18496	9.437248
Conv4	3x3	16x16x64	36928	18.874432
MaxPool2	2x2	8x8x64		
Conv5	3x3	8x8x128	73856	9.437312
Conv6	3x3	8x8x128	147584	18.874496
MaxPool3	2x2	4x4x128		
Conv7	3x3	4x4x256	295168	9.43744
Conv8	3x3	4x4x256	590080	18.874624
MaxPool4	2x2	2x2x256		
Fc1		2048	1049600	2.098176
Fc2		512	524800	1.049088
Fc3		10	5130	0.01025
Total			**2751786**	**108.73697**

TABLE I. PARAMETERS EVALUATION FOR SEVERITY OF CAR CRASH

Algorithms	Parameters					
	Accuracy in %	*Precision score in %*	*Recall score in %*	*F1 Score in %*	*Sensitivity in %*	*Specificity in %*
CNN	68.9	66 55 78	70 43 85	68 48 81	70.37 42.85 84.88	65.51 54.55 77.65

VII. CONCLUSION

We came up with idea of appending two furthur optimisation sot our cnn model in which Evolutionary Algorithm based Hyper parameter optimisation is used in case of commities of multiple CNN with mathematical application.Another Optimisation is using Pyramid Pooling Technique in the pooling layer instead of other basic methods.In some cases there can be a chance of time lag as our model takes time to analyize the best hyperparameter in training Dataset.

TABLE II. PARAMETERS EVALUATION FOR POINT OF IMPACT

Algorithms	Parameters					
	Accuracy in %	*Precision score in %*	*Recall score in %*	*F1 Score in %*	*Sensitivity in %*	*Specificity in %*
CNN	73.97	79 66 72	81 67 68	80 67 70	80.95 67.30 68	79.16 66.03 77.34

TABLE III. PARAMETER EVALUATION TO CHECK IF THE CAR IS DAMAGED OR NOT.

Algorithms	Parameters					
	Accuracy in %	*Precision score in %*	*Recall score in %*	*F1 Score in %*	*Sensitivity in %*	*Specificity in %*
CNN	89.93	89 90	90 88	90 89	90.45 88.10	88.88 89.77

VIII. Acknowledgement

We would like to express our profound gratitude to Mr. Dinesh Naik, Assistant Professor,Information Technology department, NITK for their contribution to the completion of our project.Your useful advice and suggestions were really helpful to us during the project's completion.

IX. References

- Navneet Dalal and Bill Triggs, "Histograms of Oriented Gradients for Human Detection," in IEEE Computer Vision and Pattern Recognition, 2005, pp. 886–893.
- Assi, K., 2020, December. Prediction of traffic crash severity using deep neural networks: a comparative study. In 2020 International Conference on Innovation and Intelligence for Informatics, 0Computing and Technologies (3ICT) (pp. 1-6). IEEE.
- E. Bochinski, T. Senst, and T. Sikora, "Hyperparameter Optimization For Convolutional Neural Network Committees Based on Evolutionary Algorithms," I2017 IEEE Int. Conf. Image Process., pp. 3924–3928, 2017.
- L. Xie and A. Yuille, "Genetic CNN," in Proceedings of the IEEE International Conference on Computer Vision, 2017, vol. 2017-October, pp. 1388–1397.
- M. Wistuba, N. Schilling, and L. Schmidt-Thieme, "Hyperparameter optimization machines," in Proceedings - 3rd IEEE International Conference on Data Science and Advanced Analytics, DSAA 2016, 2016.
- Q. V Le, J. Ngiam, A. Coates, A. Lahiri, B. Prochnow, and A. Y. Ng, "On Optimization Methods for Deep Learning," Proc. 28th Int. Conf. Int. Conf. Mach. Learn., pp. 265–272, 2011.
- Domhan, J. T. Springenberg, and F. Hutter, "Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves," in IJCAI International Joint Conference on Artificial Intelligence, 2015.