EVENT DETECTION BASED ON IMAGES

Abstract

Multimedia event-detection had acquired a great deal of interest in regards with progress of video-technology and increase in multi-media data. The video content complexities including repeated individuals interaction, different scenes and noisy overlapping becomes tedious to characterise the event concepts and subjects. However in assuming, dynamic data nature, imbalanced data issue, higher volume in image streams, it becomes impractical to determine and classify the events based on features.The study propounded to determine such events from multiple image classes using convolutional neural networks, ReLu and Softmax activation functions.

Keywords: Convolutional neural network, Perceptron, Edge Detection, Pooling

Authors

Dr. P. Venkateshwarlu
Assistant Professor
Department of Computer Science
Vaageswari College of Engineering
Karimnagar,Telangana,India
venkateshwarlupurumala@gmail.com

Dr. B. Manjula
Assistant Professor
Department of Computer Science
University College of Science
Warangal,Telangana,India.
manjulabairam@gmail.com

I. INTRODUCTION

In the present day era most of the multimedia streams of data is being generated by various social networks and mobile devices due to the increase in photographs taken by almost everyone on daily basis in recent years than ever before[1]. The major task in the present day scenario is to how better we can organize a huge gallery of personal photos which are to be assigned to respective album based on the event type based on the specific environment[2]. For example events similar to “Concert”, “Exhibition”, “Fashion”, “Graduation”, “Mountain Trip”, “Sea Holiday, “Sport”, “Wedding”[3]. The process of assigning labels to these eight eventalbum is performed either manually or through other aspects that rely over any specific algorithm being implemented such as location etc.Though most of the analysis is performed is with respect to the contentbased image analysis[4]which has been introduced in recent times to organize photos by implementing some process of selection the images related to a particular event to facilitate nice memories of our lives[5] based on the specific interests images are classified.

Image classification tends to provide an important indexing procedure over any image related database or the dataset which requires performing accurate and dependable classification whose task is to significantly decrease the image similarity measure in terms of time over a huge database or a dataset. Most of the proposed hybrid approaches that are proposed till date will combine some of the existing approaches for performing the classification and it is open fact that these proposed hybrid methods have not been tested over huge datasets due to hurdle of availability of resources.

For providing an example both Fitz and Green implemented frequency spectrum based approach over 40 images for performing classification and similarly Kawagoeand Tojoimplemented another approach which is based on structure using 94 images. The majority of work being performed inimage classification is merely based on the aspect of supervised learning using assignment of discrete classesassigned based on knowledgefeatures.

Event based classification of images are performed thorough two ways[6]. In the initial event task, all the images are categorized based on the event till whole album is classified by considering all the sequence of photos that exist in a dataset, though the classification of images by assigning them to a specific event type into its related album is unknown when implemented.

Therefore, in this chapter the major focus is on the second task which performs event identification and classification CIFAR-10 a standard dataset used for performing multiclass classification. As to identify a specific event over anyconvoluted scene by means of largerdiscrepancy in visual manifestation[7]“Deeplearning techniques”[8] are extensively usedto tweak active “Convolutional Neural Networks (CNNs)”over any specific event image datasets[9] .

Consequently in this chapter a slighter distinct methodology is considered while using the “conventional usage of a CNN”for empowering to be a discriminative model while designing the classifier[10]in accordance to improvise the generative model that specifically performs a specific activity on the input imagesthat are related to other domain based on the active methods implemented for performing image captioning[11] for generating labels for images. Our main contribution is a demonstration that the generated descriptions can be fed to the input of a classifier in an ensemble in order to improve the event recognition accuracy of traditional methods. Though the proposed visual representation is not as rich as features extracted by fine-tuned CNNs as they are better than the outputs generated for object detectors . As our approach is completely different than traditional CNNs which has the potential to combined with them into an ensemble that possesses high diversity and consequence towards obtaining maximum accuracy.

II. SOFTMAX ACTIVATION FUNCTION

The term “Softmax” is in general a mathematical function that translates a given vector of values into a vector of probabilities generated for each of the value that s comparative over a qualified scale of each of the vector values. And this function is commonly implemented on “machine learning” for implementing the activation function such that configuration of network is performed using N output values where each class performs classification task for normalizing the outputs and translating the weightedsum into sum of probabilities being interpreted as the membership probability over each class.

“Softmax activation function” characterizesthe flattening version of the activation model which comprises of a possible unit along with maximum input and output +1 for the rest of the units which will have value 0. We can further implement the same as a method which takes a list of numeral values for returning the multinomial or “softmax” probability distribution over the provided list consists of multi class classification issues as the activation function requires membership of more than two class labels whose data must be prepared as the target variable comprises of class labels that initially encodes the labels by applying a integer value over each class label that ranges from 0 to N-1 class labels.

The error rate generated concerning the anticipated and predicted multinomial probability distribution based on the evaluated value denoted by cross-entropy as per the rate of error generated over the updated model and moreover the cross-entropy is also called as the loss function.

1. **Edge Detection is Helpful in Extracting the Features of the Images.:** In “digital image processing” is the aspect of edge detection which is considered to be a technique used in computer vision for identifying the boundaries of an image in a specific photograph as a methodology that implements the algorithm which searches for discontinuities in attaining the brightness pixel in an image being converted to grayscale picture for detecting the features or objects in a photograph by performing segmentation and extraction techniques in an algorithm as the change in image points leads to change in brightness or change in sharpness of an image leads to generation of curved line segments denoted as edges.

   “Edge detection techniques”are implemented based on the isolated features which detect the objects based on the boundaries of an image after recognizing the model must analyze the image and perform the identification of the object as per the requirements of the prepared model to observe the delineation of image along with its boundaries for extracting the image contents being recognized through the object by using the false positives or through errors being identified for refining the algorithm to enhance the accuracy of image.

2. **Canny Edge Detection Algorithm:** Canny Edge Detection is a popular multi-stageedge detection algorithm which performs the following steps for detecting edges of an image and most widely used:

   "Noise Reduction" is the process of edge detection which is inclined to noise that may be available in an image where the initial step is to eliminate noise inside the image with a 5x5 Gaussian filter.

   "Identification of level of intensity in image gradient"is the process of smoothing an image is by performing filtering using Sobel kernel in both horizontal (Gx)and vertical direction (Gy)for obtaining the initial derivative from thetwo images for identifying the edge gradient based on the direction for each of the pixel using:

$$Edge\_Gradient\ (G) = \sqrt{G_x^2 + G_y^2} \qquad (1)$$

$$Angle\ (\theta) = \tan^{-1}\left(\frac{G_y}{G_x}\right) \qquad (2)$$

   The direction of gradient is always perpendicular to edges as they are rounded to any one of four angles that represents "vertical, horizontal and two diagonal directions".

   "Non-maximum Suppression" is performed after obtaining the gradient magnitude and direction after performing the full scan of image which is performed to eliminate the unnecessary pixels that comprises of the edge and to obtain this process every pixel is verified with the local maximum value and the neighborhood over the direction of gradient.

   "Hysteresis Thresholding" verifies which one is an edgeand which one is not based on the two threshold values denoted as minVal and maxVal foridentifying the gradient intensity as when the maxVal is greater in terms of gradient value then it is sure to be an edge and when minVal is lesser in terms of gradient value then it is a non-edge. The threshold value generated while classifying based on the gradient intensity level which lies between minVal and maxVal threshold values represents the connectivity between edges and non-edges and when they are connected then "sure-edge" pixels are represented to be edges.

3. **Bounding Box Regressor for Crop an Image:** The "Bounding Box Regressor (BBR)"[12]is used in object detection and leads to crop the image based on the anchor boxes comprising of fixed positions along with their aspect rations for refining the nearby objects much precisely due to which BBR is tightly coupled with other components of an image while performing object identification.

   The default boxes or the anchor boxes are the initially defined bounding boxes which have been sampled over a standard grid that comprises of fewer chosen scales of aspect ratios used to estimate the object boxes through training data which is well harmonized over other components performing object detection using pre-defined object classes with characteristics of anchor boxes.

An object box comprises of entire image segment instead of a portion of it is used to draw the boundingboxes comprising of various image segments generated through "hierarchical image segmentation methods" without any supervision over the object location leads to unsupervised technique which is often fail to generate segmentation results as they comprises of inadequate aspects in terms of recall and localization accuracy. Whereas supervised techniques leads to better performance.

The "Convolutional Neural Network (CNN)" is used to implementlower level layers for performing large scale image classification which tends to transfer the process of classification over distinct domains along with distinct visual identification tasks as the process of lower level image transformation representation is considered to be a regular technique for avoiding overfitting while performing "object detection" as well as "semantic segmentation" by transferring the knowledge acquired through segmentation to various objectclasses whose segmentation based annotations are not obtained.

III. POOLING PROCESS

When a intense network or the multi layer perceptron model is considered which comprises of distinct neurons in an individual layer are further connected with various other neurons in preceding layer based on the weight of params generated in a network for performing the cross product of numerous neurons that are available in connected layers for processing an image consisting of maximum resolution needs elevated computational power to scale over superior images by training millions of params over insignificant images.

Thesekinds of issues can be resolved by implementing convolution neural network as the anticipated result is associated with the nearer input neurons in association with preceding layer as the convolution and pooling processes will lead to attain it.

The general process performed in convolution is the process of matrix multiplication over an image where initially every image will be converted into a directed numeric array representing pixel values associated with an image will be multiplied with a portion of image that relies on the filter size to place a filter over another one till all the pixels in the image are covered. This convolution process will reduce the image size drastically as we have matrix of numeric values than pixels and it leads to easy transformation of an image for performing various operations on it.

The process of Pooling is implemented to extract diverse features for generating output of an image over a convolution layer by following a similar process that slides over an image using a particular pool size or the kernel size and there exists two types of pooling: a) Max Pooling (MP) and b)Average Pooling (AP). MP is used most often as it stores maximum number in pool and AP will store average number in the pool and either in case of MP or AP the rest of the values will be discarded and further based on the each pool value significant features in an image will be selected and rest of the features based on their values when compared with the each pool value will not be selected in the model.

$$[(n+2p-f/s)+1] *[(n+2p-f/s)+1] \tag{3}$$

In equation 3 size of image is denoted n and padding is denoted by p and stride is denoted with s and the MP or AP process will be implemented using following phases:

1. Initially choose an appropriate sliding window size
2. Select a stride to perform jump across the pixels
3. Move the window over the filtered images
4. Obtain the maximum value from each window (max pooling)

IV. DATA AUGMENTATION

Practically when the amount of data improves the performance of the deep learning model also enhances as it requires huge amount of data which is merely a challenging aspect as in most of the cases it is not feasible to supply huge amount of data to feed the deep learning model as it leads to wrong interpretations while learning the patterns from the data provided may lead to unintended performance. The only way to deal with this kind of issue s through Augmentation of an image.

The process implemented for performing image augmentation will modify the training data for creation of some more data while performing training process which artificially enhances the existing dataset related to same class that trains the deep learning model. The process of transformation includesspecific operations over the image modification such as “shifts, flips, zooms, and much more”.

The main intention is to enhance the training dataset providing dissimilarities in the training image set images are possible seen by the model by performing horizontal flip of any specific inputimage may make sense since the photo might have been taken either from left or right hence the vertical flip over the image may not make much sense to the implemented model. While choosing any of the particular data augmentation methodologies it required to train the dataset is to be carefully selected based on the training dataset perspective along with the problem domain for implementing data augmentation methods for verifying the model performance that improvises the model.

The present day deep learning algorithms similar to “Convolutional Neural Network (CNN)” has the capability to learn various features which are alternative based on the image location being augmentation as an aid while performing the process of transformation that learns the invariant approach for learning the features while transforming the model for example left to right to toptobottom arrangement with slighter levels in any of the images as image data augmentation is not performed over validation or test dataset but merely on training dataset. This process is entirely discrete from preparation of data by performing image resizing along with the process of pixel scaling being performed time after time over all datasets for interconnecting the chosen model using keras library in python.

V. DATA PREPROCESSING

CNN comprises of deep learning algorithms that are incredibly powerful in terms of analyzing the images by initial preprocessing it for generating the training dataset comprising of distinct sizes as in most of the cases images are resized earlier to creation of inputs. Initially square and rectangle images are to be resized to a specific pixel shape based on the requirement like 256×256 pixels or a specific side to 256 pixels based on the shortest side of an image for further cropping the image obtained through training augmentation.

The obtained mean pixel value generated after performing subtraction through each of the pixel for obtaining the centering process based on the generated per channel as these values are estimated in terms of pixels in training dataset in terms of “red, green, and blue” channels over the colour images.

1. **Edge Detection**: It is the process where edges are the curves where we can expect sudden changes in terms of attained brightness which leads to change in shadow lines that seems to be the process of discontinued surface reflection properties required to identify the level of discontinuity over a specific image based on its brightness along with image derivatives will produce the using “Laplacian of Gaussian (LoG)” for performing edge detection.

2. **Boxing:** It describes the spatial position in an object as the bounding box is a specific rectangular representation of an area that is being determined by x and y coordinates initiating from the upper left corner of a rectangle followed till reaching the lower right corner being commonly represented through bounding box using (x,y)

3. **Cropping:** is the initial type of augmentation performed was horizontal flips of a smaller cropped square image that was expanded to the required side using horizontal reflections within the image. As a regular process all images need to preprocess to perform robust learning as there may exists distinct processing method and similarly crop is a required pre processing in terms of the object position comprising of larger variance based on which input images are prepared for the training data part.

VI. IMAGE CLASSIFICATION

The dataset used in this chapter to perform classification is “CIFAR-10” which is a small image classification problem as “CIFAR-10” is considered to be a standard dataset merely implemented for performing tasks related to computer vision using deep learning though “CIFAR-10” dataset can be efficiently resolved byimplementing over the basis of learning and practicing the entire development process which is further evaluated and usedfor generating image classification using convolution deep learning neural networks from scratch.

The acronym “CIFAR” stands for “Canadian Institute for Advanced Research” and the dataset used in thisstudy is “CIFAR-10”which is a subset of “CIFAR-100” dataset and the “CIFAR-10” dataset comprises of 60000 colour images with 32 × 32 pixel with objects of 10 classes like “Concert”, “Exhibition”, “Fashion”, “Graduation”, “Mountain Trip”, “Sea Holiday”, “Sport” , “Wedding”.

When we compare other datasets with respect to “CIFAR-10” dataset there are small images in the dataset compared with a standard photograph being considered by a computer vision researcher. Though the “CIFAR-10” dataset is a peculiar one being widely utilized by most of the computer vision algorithms considered as a benchmark for implementingML area by achieving over 80% of straight forward classification accuracy.

The “Rectified Linear Activation Function (ReLU)” comprises of a piecewise “linear function” which will output the input precisely if it is positive or else the output will be zero and moreover ReLu has became as the default activation function for most of the categories of neural networks (NN) as a specific model for utilizing and training the training process to be easier for achieving better performance.

While performing training to the deepNN with multiple layers is considered to be more challenging and sensitive while configuring random weights while implementing a learningalgorithm. The main issue with distribution of inputs in deepNN is the tendency to change based on mini batches based on the weights being updated as they rely on moving target along with the distribution of inputlayers in a network.

The “batch normalization” is a process that performs training over very deepNNthat standardizes various inputs over a layer for each one of the mini batch that effectsthe soothing the learning process for reducing dramatically number of training epochs based on the needed train deep networks.

In the proposed classification framework being proposed in figure 1 comprises of classification model being updated by imparting hidden layers related to CNN by implementing “CIFAR-10” dataset andto illustrate the enhanced results in experimental study.

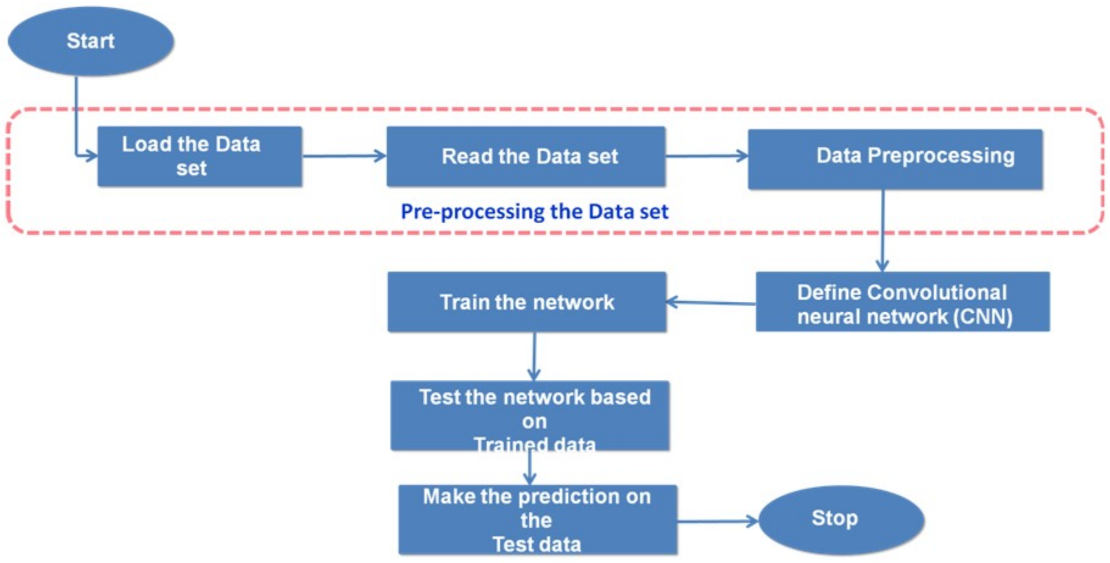

Start
Load the Data set
Read the Data set
Data Preprocessing
Pre-processing the Data set
Train the network
Define Convolutional neural network (CNN)
Test the network based on Trained data
Make the prediction on the Test data
Stop

Figure 1: Image Classification frame work

In the proposed image classification framework as illustrated in figure 1 is initiated by loading the dataset as in this chapter we are using “CIFAR-10” data set through which data is read and data is being pre processed by performing "Edge detection", "Bounding box", "Cropping of image" as required over all the training dataset then CNN is defined with four hidden layers along with kernel size of 3 and max pool window size is of 2 and the final filter size is of 32 are used to train the network then further network is tested using testing data over trained data and the framework is ended by predicting the accuracy based on the test data provided.

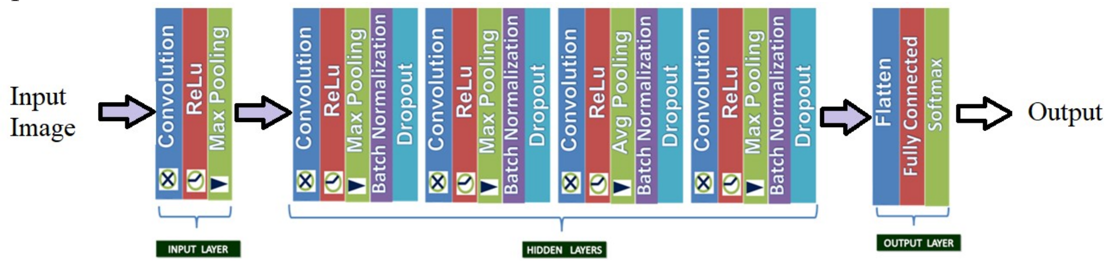


Figure 2: Internal structure of proposed CNN

Figure 2 illustrates that the proposed CNN architecture comprises of one input layer along with one output layer and with four hidden layers. Input layer comprises of convolution, ReLu and Max Pooling and the output layer comprises of Flatten, Fully connected and Softmax and four hidden layers are used in the CNN architecture where each of the hidden layer comprises of Convolution, ReLu, Batch Normalization and Dropout.

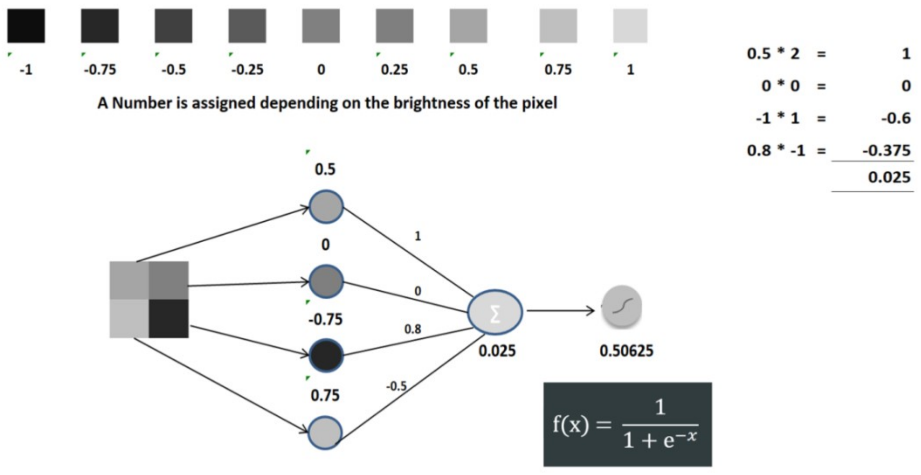
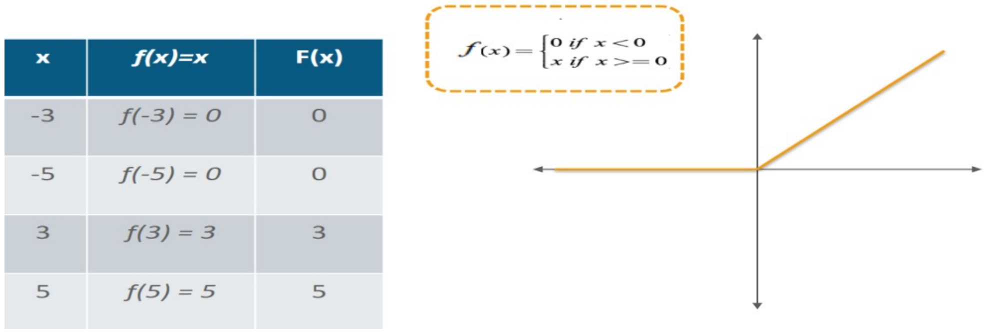


Figure 3: Perceptron Example with 4 pixels

Perceptron is a basic NN building block as the training of perceptron comprises of supply it to numerous training samples for evaluating the output for each of them where each of the sample comprises of w weights andare further modified such that will lead to reduction of "*output error*" denoted by *"mean squared error"* at a higher levelfor identifying the variation between the target and the actual outputs such that the training remains same.

Figure 3 illustrates single input along with single output and four hidden layers as the units and neurons are compatible as each of the unit comprises of a specfcperceptron where the input layer serves as input to hidden layers and generates data related to output layer where each of the connection among two neurons comprises of w weight with each unit of layer t is connected with each unit in the previous layer (t-1) with weight 0 and the outputs are calculated based on the activation function and denotes the result of the output layer is denoted as the output of the network.

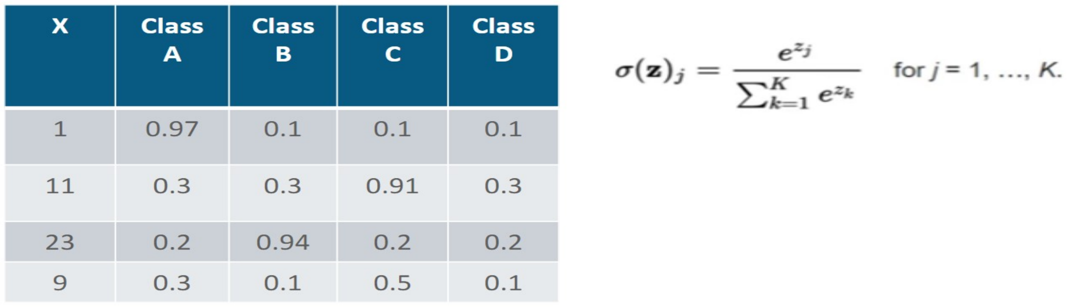


x	f(x)=x	F(x)
-3	f(-3) = 0	0
-5	f(-5) = 0	0
3	f(3) = 3	3
5	f(5) = 5	5

$$f(x) = \begin{cases} 0 \text{ if } x < 0 \\ x \text{ if } x >= 0 \end{cases}$$

Figure 4: ReLu transformation function implementation based on threshold

The ReLUtransformation function is illustration in figure 4 which will activate the node when the provided inputwhich is beyond certain measure as the input which is lesser than zero and the output value is zero and further as the input values is raised above evident threshold based on the linear association with dependent variables.



X	Class A	Class B	Class C	Class D
1	0.97	0.1	0.1	0.1
11	0.3	0.3	0.91	0.3
23	0.2	0.94	0.2	0.2
9	0.3	0.1	0.5	0.1

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{for } j = 1, \ldots, K.$$

Figure 5: Implementation of softmax function using four hidden layers

As illustrated in figure 5 illustrates when there exits numerous classes of outputs being implemented with Softmax function based on the probability distribution of each of the useful findings of a class comprising of maximum probability to identify the classifiers output layer.

Table 1: Description of Dataset Used

Attribute	Particulars
Name	CIFAR-10
Link	http://loki.disi.unitn.it/~used/
No. of Training Images	12000
No. of Testing Images	2400
No. of Events	8

Each of the image in the dataset comprises of 150 × 150 pixels RGB images in jpg file format are used where figure 6 represents the sample of the images being used

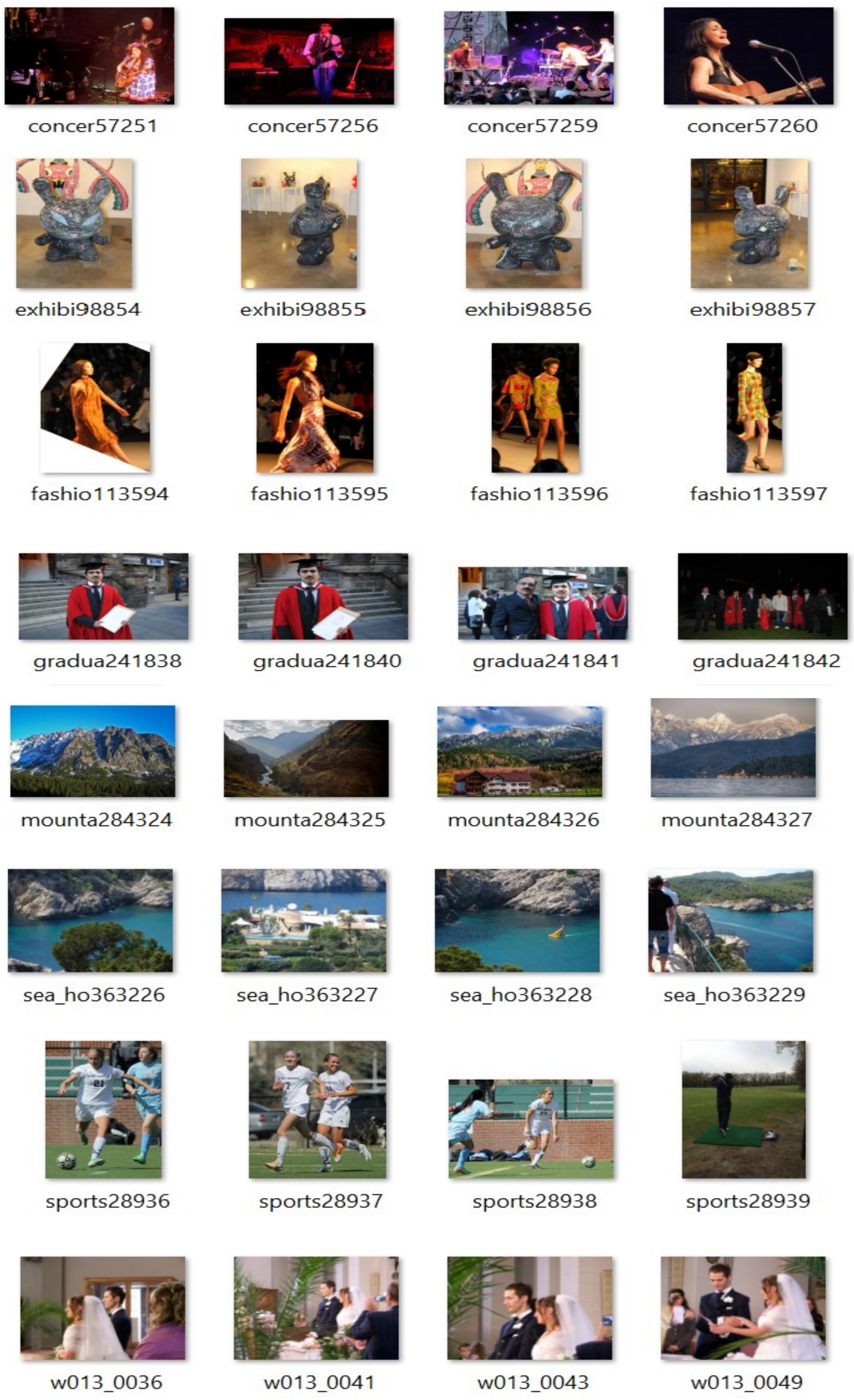


Figure 6: Sample images considered

```
8 classes with 14400 images in total
```

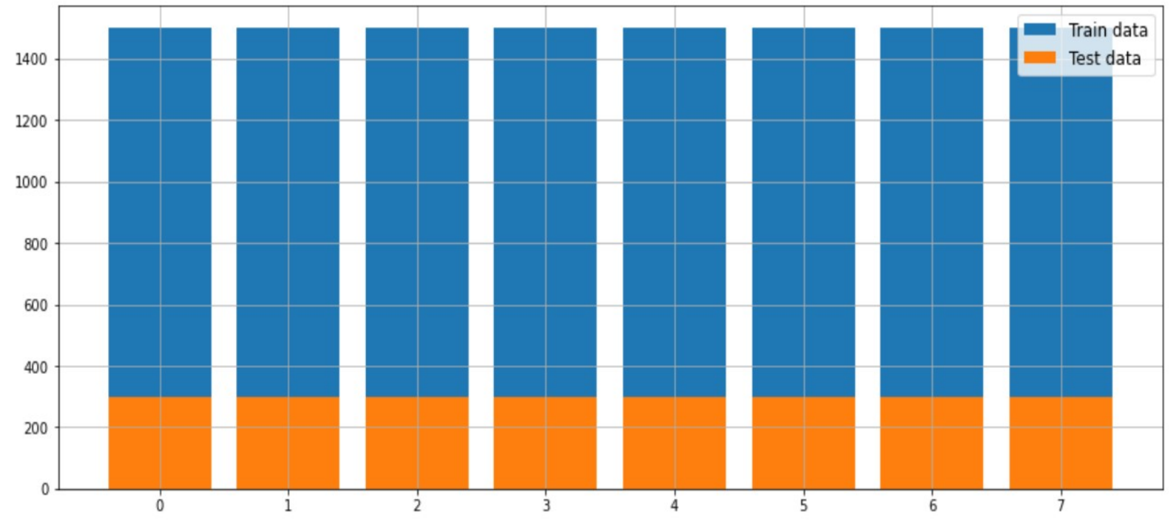


Figure 7: Visualization for division of dataset for train data and test data

Figure 7 represents the visualising of division of dataset being taken for training and testing of the data where 8 different classes have been considered for the study and 14400 images in total are considered and the shape of the train images is 12000×150×150×3 where 12000 denotes labels then 150 represents the pixels and 3 denotes the RGB values associated with each of the image and the training sample considered is 80% and testing sample space is of 20% being considered for obtaining effective results.

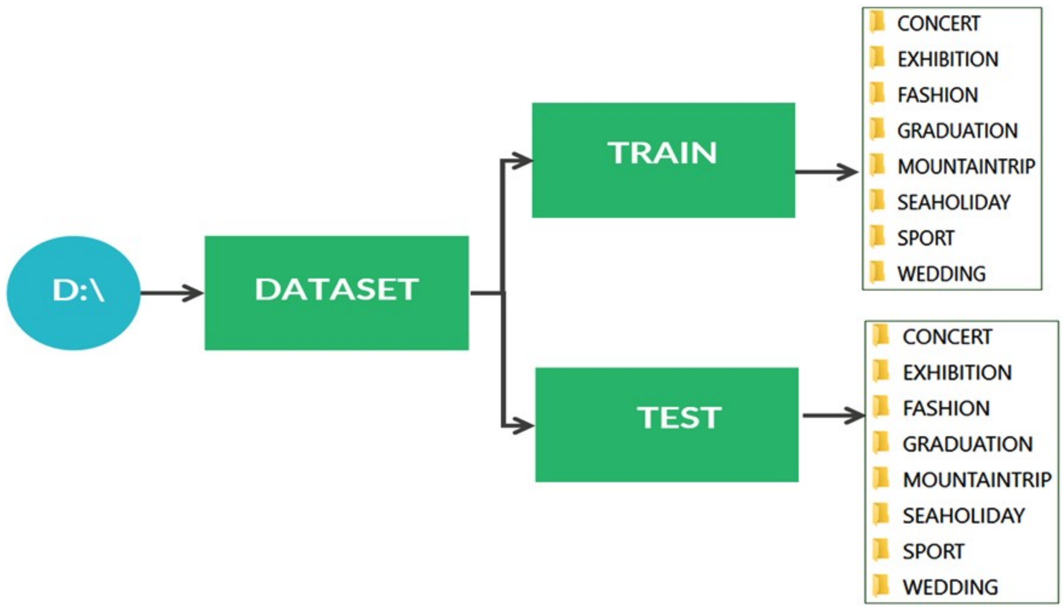


Figure 8: Visualization loading of train data and test data

Figure 8 represents the process adopted for loading the “CIFAR-10” dataset which is acquired on 06th March 2022 for implementing the proposed algorithm as the model implemented is placed on a local disk by organizing into two folders as illustrated in figure.

By implementing the python methodos.walk() which generates the list of filenames available in the folder over the specified path in either bottom up or top down manner is navigated in order to load all the list of images.

In the process of reading the dataset each of the image name is interpreted one after another and further attached to the generated list by resizing the image comprising of 150×150 pixels by adding the labels along with the index value over each of the resized image and is further transformed into a numpy array by setting the title class as depicted in figure 9.

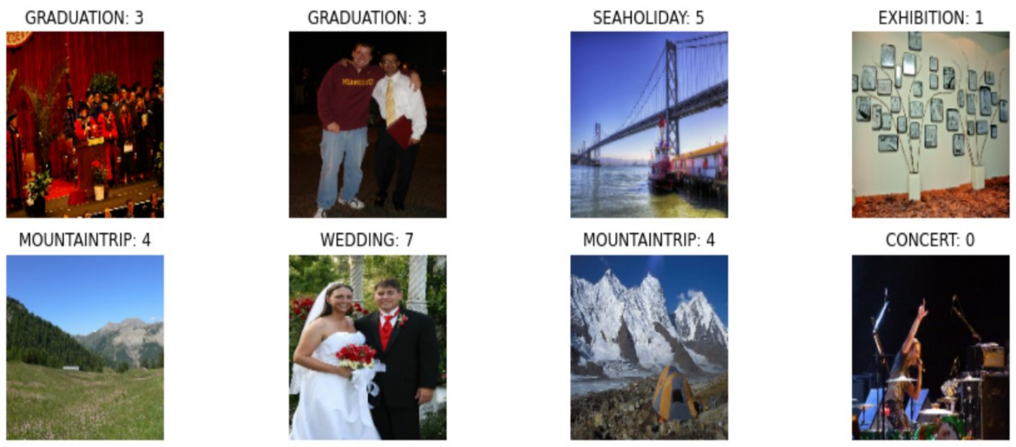


Figure 9: Loading of image sample that are resized, indexed and further labeled

The data is further processed based on the labels assigned and index values initially edge detection is performed on the images by extracting the features over RGB images and storing them in numpy array based on 12000 labels assigned and then “Canny edge detection (CED)” algorithm is implemented for performing the edge detection process by converting the image to gray image from RGB image and one of the major disadvantage of CED is it is not accurate in detecting all the features in an image which leads to loss.

This loss can be uncovered by implementing bounding boxes where nonzero edges are identified and cropping of mage is performed by generating new $\{x_1,x_2,y_1,y_2\}$ values of image which illustrates the cropping process of a region in an image by replacing the numpy array of cropped mage with the original numpy array without replacing the original images and further it is resized to image width and image height of 150×150 pixels and this entire process resultant is illustrated in figure 10 and this entire process is implemented on the machine which consumed memory limit of 268435456 in bytes and incarnation of 895853056964357322 where incarnation is the process of allocation of resources to concrete servant classes through adapter class.

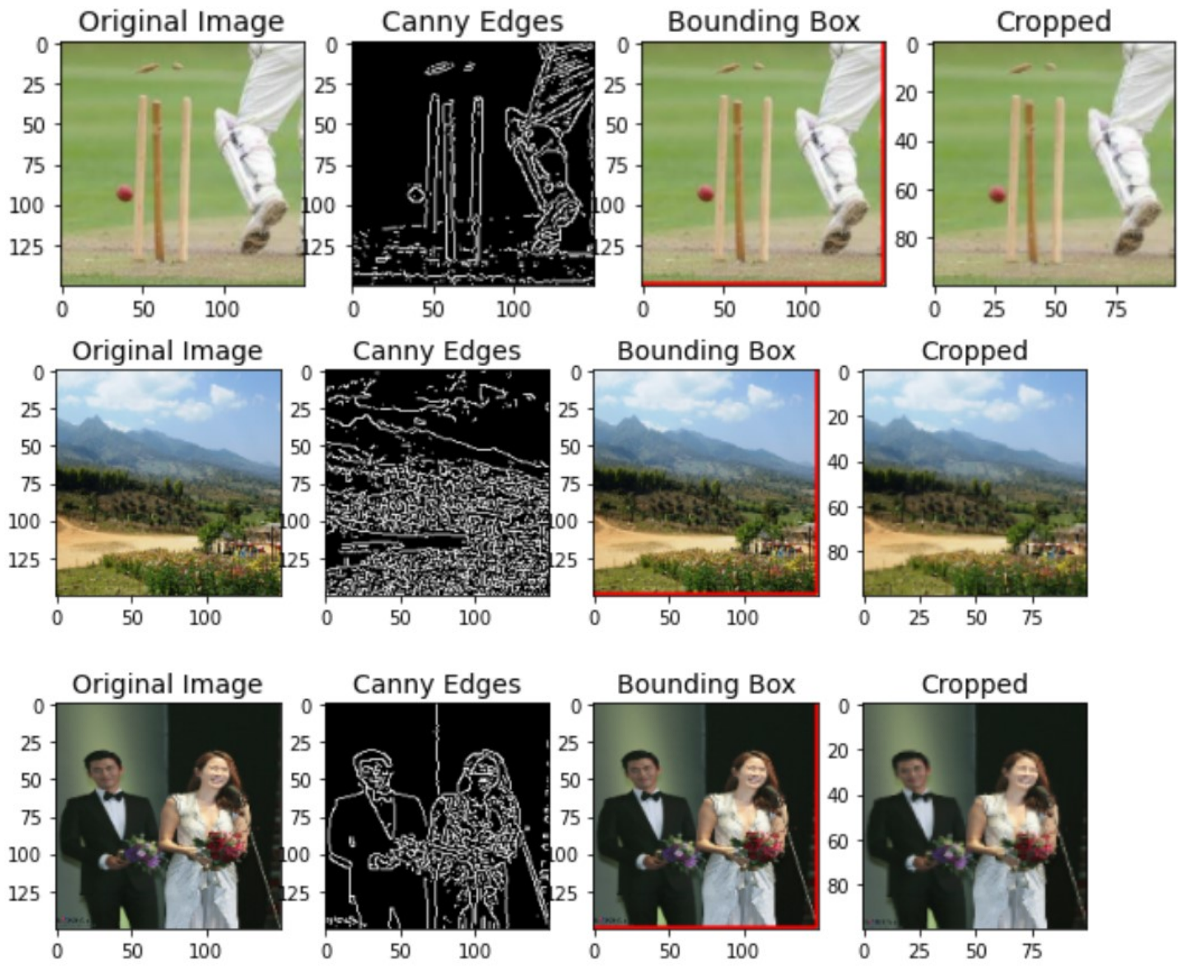


Figure 10: Sample of results generated through processed mages

VII. CREATION OF MODEL

The steps taken for creation of model are initiated by splitting the data interms of training and validation using the sklearn module of python which is most useful library in ML used for performing statistical modelling and comprises of efficient tools for predicting data analysis through various aspects. The entire dataset considered is categorized into 80% of data as training and rest of the 20% of the data is utilized for testing or as a validation Set with 32 as the filter size which performs filtering process which generates the dimensionality of the output space for identifying any specific feature. The anticipated environment will implement the proposed model with 3 to be the kernel size which is the length of the two dimensional convolution window and with 2 as the size of window for performing max pooling window.

1. **Pooling:** The process of pooling will tend to merge the image data used for diminishing the data size by setting the pooling window size which is set to value 2 in the implementation which is a major aspect which creating the model as pooling process will show direct impact on the loss function. The pooling process is initiated by selecting the appropriate sliding window size which is 3 in the implementation from which a stride is selected by performing walk thorough over the window diagonally over the filtered images from each of the window using the maximum pooling value which is set to 2 in the implementation.

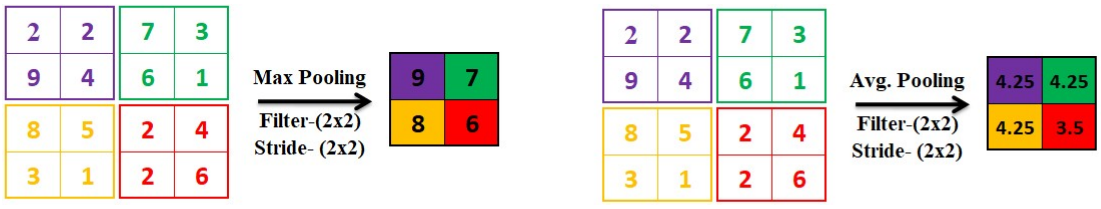


Figure 11: Maxpooling and Average pooling results

2. **Epoch vs. Batch Size vs. Iteration:** The "Gradient Descent" is a specific iterative optimization algorithm which implements ML for obtaining best results where the term Gradient denotes the degree of "inclination" or "declination" of a slope in an image and "descent" means the instance of "descending" and the algorithm in an iterative manner generates results by executing multiple times for obtaining optimal result.

   The proposed algorithm we have 50000 samples of training data with a batch size of 1000 images hence 50 iterations are required to finish one epoch which internally uses "Gradient descent" to complete single epoch by performing forward or backward pass for loading the entire dataset by performing 50 batches to load such that the data must not fall in either overfitting or in underfitting but it must be optmum.

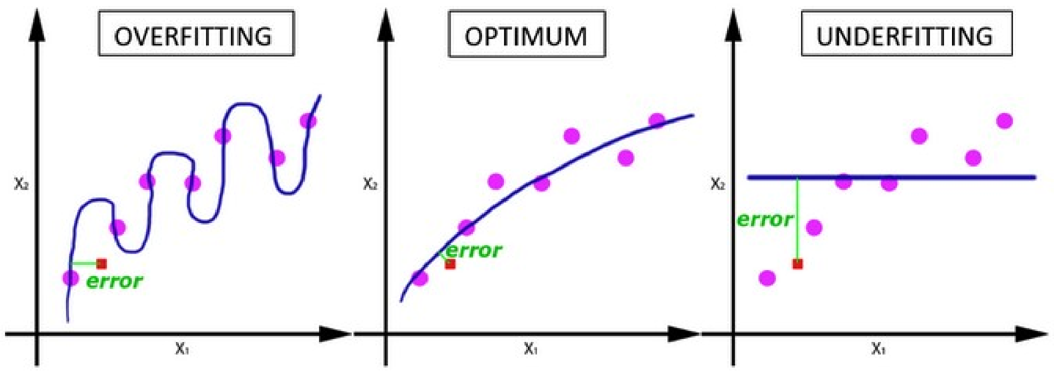


Figure 12: Epochs Illustration

   When the number of epochs enhances in terms of additional number of times due to which the weights are changed over a neural network and this leads to impact over the curve that leads to any one of the situation such as "underfitting" to "optimal" to "overfitting" curve as illustrated in figure 12. But there is not specific value or threshold for these three terms as these values changes from one dataset to another and also based on the number of features considered.

3. **Data Augmentation:** To develop a potential image classifier "Image Augmentation" is required which artificially generates training images based on distinct methods for processing the images by rotating the images or shifting the images as per the requirements using "ImageDataGenerator" class from keras which fits the model as specified in the algorithm implemented comprising the rotation_range of 25 then zoom_rangethen the width_shift_range attribute and the height_shift_range properties are set to 0.1 with the shear_range of 0.2 with the property horizontal_flip set to value true leads to the execution result as depicted in figure 13.

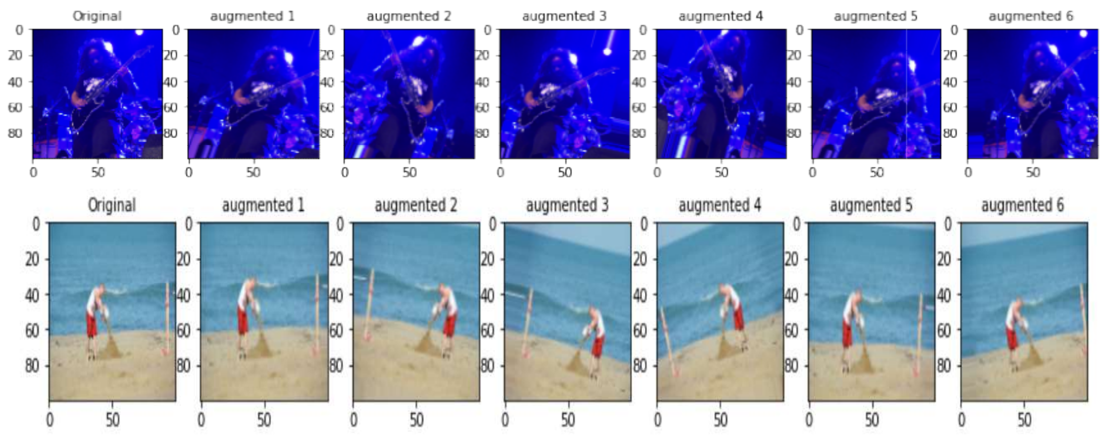


Figure 13: Sample images of implementation of augmentation in dataset

VIII. PROPOSED ALGORITHM

1. **Algorithm:** Classification of images

2. **Input:** image for which class is to be predicted

3. **Output:** class of image to which it belongs

```
Step 01: Start
Step 02: Load the images of different events through single epoch.
Step 03: split images for training and validation
Step 04: Read image name and append to a list
Step 05: resize the image
Step 06: label and index the image based on its event category
Step 07: perform edge detection
Step 08: covert image into numpy array
Step 09: set the filtersize for the dimensionality of the output space
Step 10: set the kernel_size for length of the 2D convolution window
Step 11: implement max_pool for assigning size of the max pooling windows
Step 12: define number of EPOCHS
Step 13: define batch_size #number of samples for training
Step 14: iter_per_epoch = len(X_train)   // batch_size
Step 15: val_per_epoch = len(X_val) // batch_size
Step 16: model = Sequential()
Step 16.1: add convolutional_layer
Step 16.2: add internal layers with pooling batch normalization & increased dropout and add output layer
Step 16.3: perform data augmentation
Step 16.4: fit the model
Step 16.5: evaluate the model
Step 16.6: validate the output
Step 17: Stop
```

The proposed algorithm will perform classification of images with all the images based on their classes to predict the accuracy is considered as input to the algorithm and the class of image to which it belongs to is considered as the output of the proposed algorithm.

The tasks of algorithm are initiated by loading the images from distinct events using an iterative single epoch then perform splitting operation over dataset to generate training and testing or validation set and then perform reading of labels and classify them to specific part of the list then resize the images to 150×150 pixels and further perform labeling and creation of index based on event type.

Then the algorithm is continued by performing edge detection then convert the image into numpy array then set the filter size based on the dimensionality of the output space provided with the kernel size being set to 2D convolution then invoke max_poolfunction for assigning the size to various max pooling windows.

After performing all these steps number of epochs are set that is to 120 is applied in the execution and then further batch size is assigned in the implementation it is set to 50 and these 50 iterations are performed to obtain the epoch while performing the training process and testing or validation process and the model is generated with these configurative steps being performed.

Further the model is implemented in a sequential manner by adding convolutional layer after which internal layers along with the pooling of batch normalization is performed with increased dropout for generating the output layer then on this data, data augmentation is performed then the model is fit and the evolution model is generated then the results are validated.

IX. RESULTS

The results are obtained by implanting the proposed algorithm as illustrated in section 3.8 using “Anaconda: Jupter Notebook” software with

Table 2: Trial 1 Proposed Model Summary with Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 100, 100, 32)	896
max_pooling2d (MaxPooling2D)	(None, 50, 50, 32)	0

conv2d_1 (Conv2D)	(None, 50, 50, 64)	18496
max_pooling2d_1 (MaxPooling2	(None, 25, 25, 64)	0
dropout (Dropout)	(None, 25, 25, 64)	0
conv2d_2 (Conv2D)	(None, 25, 25, 128)	73856
max_pooling2d_2 (MaxPooling2	(None, 12, 12, 128)	0
dropout_1 (Dropout)	(None, 12, 12, 128)	0
flatten (Flatten)	(None, 18432)	0
dropout_2 (Dropout)	(None, 18432)	0
dense (Dense)	(None, 512)	9437696
dense_1 (Dense)	(None, 8)	4104

Total params: 9,535,048
Trainable params: 9,535,048
Non-trainable params: 0

Table 3: Trial 2 Proposed Model Summary with Model: "sequential"

Layer (type)	**Output Shape**	**Param #**
conv2d (Conv2D)	(None, 100, 100, 32)	896
max_pooling2d (MaxPooling2D)	(None, 50, 50, 32)	0
conv2d_1 (Conv2D)	(None, 50, 50, 64)	18496
max_pooling2d_1 (MaxPooling2	(None, 25, 25, 64)	0
batch_normalization (BatchNo	(None, 25, 25, 64)	256
conv2d_2 (Conv2D)	(None, 25, 25, 128)	73856
max_pooling2d_2 (MaxPooling2	(None, 12, 12, 128)	0
batch_normalization_1 (Batch	(None, 12, 12, 128)	512
flatten (Flatten)	(None, 18432)	0
dense (Dense)	(None, 512)	9437696
dense_1 (Dense)	(None, 8)	4104

Total params: 9,535,816
Trainable params: 9,535,432
Non-trainable params: 384

Table 4: Trial 3 Proposed Model Summary with Model: "sequential"

Layer (type)	**Output Shape**	**Param #**
conv2d (Conv2D)	(None, 100, 100, 32)	896
max_pooling2d (MaxPooling2D)	(None, 50, 50, 32)	0

conv2d_1 (Conv2D)	(None, 50, 50, 64)	18496
max_pooling2d_1 (MaxPooling2	(None, 25, 25, 64)	0
batch_normalization (BatchNo	(None, 25, 25, 64)	256
dropout (Dropout)	(None, 25, 25, 64)	0
conv2d_2 (Conv2D)	(None, 25, 25, 128)	73856
max_pooling2d_2 (MaxPooling2	(None, 12, 12, 128)	0
batch_normalization_1 (Batch	(None, 12, 12, 128)	512
dropout_1 (Dropout)	(None, 12, 12, 128)	0
conv2d_3 (Conv2D)	(None, 12, 12, 128)	147584
average_pooling2d (AveragePo	(None, 6, 6, 128)	0
batch_normalization_2 (Batch	(None, 6, 6, 128)	512
dropout_2 (Dropout)	(None, 6, 6, 128)	0
conv2d_4 (Conv2D)	(None, 6, 6, 128)	147584
max_pooling2d_3 (MaxPooling2	(None, 3, 3, 128)	0
batch_normalization_3 (Batch	(None, 3, 3, 128)	512
dropout_3 (Dropout)	(None, 3, 3, 128)	0
flatten (Flatten)	(None, 1152)	0
dropout_4 (Dropout)	(None, 1152)	0
dense (Dense)	(None, 512)	590336
dense_1 (Dense)	(None, 8)	4104

Total params: 984,648
Trainable params: 983,752
Non-trainable params: 896

Table 5: Trial 4 Proposed Model Summary with Model: "sequential"

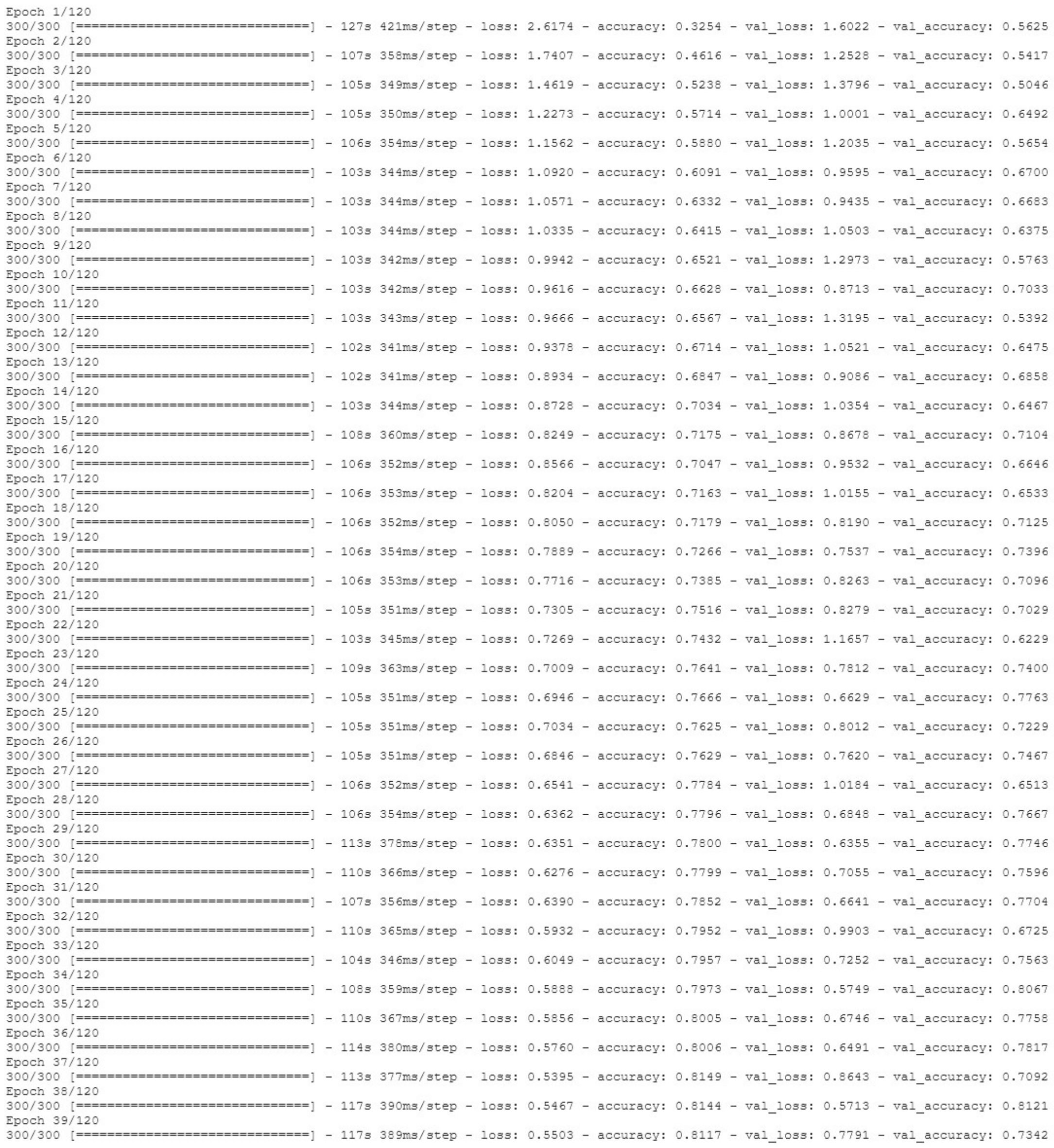
Layer (type)	**Output Shape**	**Param #**
conv2d (Conv2D)	(None, 100, 100, 32)	896
max_pooling2d (MaxPooling2D)	(None, 50, 50, 32)	0
conv2d_1 (Conv2D)	(None, 50, 50, 64)	18496
max_pooling2d_1 (MaxPooling2	(None, 25, 25, 64)	0
batch_normalization (BatchNo	(None, 25, 25, 64)	256
dropout (Dropout)	(None, 25, 25, 64)	0
conv2d_2 (Conv2D)	(None, 25, 25, 128)	73856
max_pooling2d_2	(None, 12, 12, 128)	0

(MaxPooling2		
batch_normalization_1 (Batch	(None, 12, 12, 128)	512
dropout_1 (Dropout)	(None, 12, 12, 128)	0
conv2d_3 (Conv2D)	(None, 12, 12, 128)	147584
average_pooling2d (AveragePo	(None, 6, 6, 128)	0
batch_normalization_2 (Batch	(None, 6, 6, 128)	512
dropout_2 (Dropout)	(None, 6, 6, 128)	0
conv2d_4 (Conv2D)	(None, 6, 6, 128)	147584
max_pooling2d_3 (MaxPooling2	(None, 3, 3, 128)	0
batch_normalization_3 (Batch	(None, 3, 3, 128)	512
dropout_3 (Dropout)	(None, 3, 3, 128)	0
flatten (Flatten)	(None, 1152)	0
dropout_4 (Dropout)	(None, 1152)	0
dense (Dense)	(None, 512)	590336
dense_1 (Dense)	(None, 8)	4104

```
Total params: 984,648
Trainable params: 983,752
Non-trainable params: 896
```

To fit the model following is the code snippet:

```
m = model.fit_generator(
train_gen,
steps_per_epoch= iter_per_epoch,
      epochs=EPOCHS,
validation_data = val_gen,
validation_steps = val_per_epoch,
      verbose = 1 # Verbosity mode. 0 = silent, 1 = progress bar, 2 = one line per epoch.)
```



```
Epoch 1/120
300/300 [==============================] - 127s 421ms/step - loss: 2.6174 - accuracy: 0.3254 - val_loss: 1.6022 - val_accuracy: 0.5625
Epoch 2/120
300/300 [==============================] - 107s 358ms/step - loss: 1.7407 - accuracy: 0.4616 - val_loss: 1.2528 - val_accuracy: 0.5417
Epoch 3/120
300/300 [==============================] - 105s 349ms/step - loss: 1.4619 - accuracy: 0.5238 - val_loss: 1.3796 - val_accuracy: 0.5046
Epoch 4/120
300/300 [==============================] - 105s 350ms/step - loss: 1.2273 - accuracy: 0.5714 - val_loss: 1.0001 - val_accuracy: 0.6492
Epoch 5/120
300/300 [==============================] - 106s 354ms/step - loss: 1.1562 - accuracy: 0.5880 - val_loss: 1.2035 - val_accuracy: 0.5654
Epoch 6/120
300/300 [==============================] - 103s 344ms/step - loss: 1.0920 - accuracy: 0.6091 - val_loss: 0.9595 - val_accuracy: 0.6700
Epoch 7/120
300/300 [==============================] - 103s 344ms/step - loss: 1.0571 - accuracy: 0.6332 - val_loss: 0.9435 - val_accuracy: 0.6683
Epoch 8/120
300/300 [==============================] - 103s 344ms/step - loss: 1.0335 - accuracy: 0.6415 - val_loss: 1.0503 - val_accuracy: 0.6375
Epoch 9/120
300/300 [==============================] - 103s 342ms/step - loss: 0.9942 - accuracy: 0.6521 - val_loss: 1.2973 - val_accuracy: 0.5763
Epoch 10/120
300/300 [==============================] - 103s 342ms/step - loss: 0.9616 - accuracy: 0.6628 - val_loss: 0.8713 - val_accuracy: 0.7033
Epoch 11/120
300/300 [==============================] - 103s 343ms/step - loss: 0.9666 - accuracy: 0.6567 - val_loss: 1.3195 - val_accuracy: 0.5392
Epoch 12/120
300/300 [==============================] - 102s 341ms/step - loss: 0.9378 - accuracy: 0.6714 - val_loss: 1.0521 - val_accuracy: 0.6475
Epoch 13/120
300/300 [==============================] - 102s 341ms/step - loss: 0.8934 - accuracy: 0.6847 - val_loss: 0.9086 - val_accuracy: 0.6858
Epoch 14/120
300/300 [==============================] - 103s 344ms/step - loss: 0.8728 - accuracy: 0.7034 - val_loss: 1.0354 - val_accuracy: 0.6467
Epoch 15/120
300/300 [==============================] - 108s 360ms/step - loss: 0.8249 - accuracy: 0.7175 - val_loss: 0.8678 - val_accuracy: 0.7104
Epoch 16/120
300/300 [==============================] - 106s 352ms/step - loss: 0.8566 - accuracy: 0.7047 - val_loss: 0.9532 - val_accuracy: 0.6646
Epoch 17/120
300/300 [==============================] - 106s 353ms/step - loss: 0.8204 - accuracy: 0.7163 - val_loss: 1.0155 - val_accuracy: 0.6533
Epoch 18/120
300/300 [==============================] - 106s 352ms/step - loss: 0.8050 - accuracy: 0.7179 - val_loss: 0.8190 - val_accuracy: 0.7125
Epoch 19/120
300/300 [==============================] - 106s 354ms/step - loss: 0.7889 - accuracy: 0.7266 - val_loss: 0.7537 - val_accuracy: 0.7396
Epoch 20/120
300/300 [==============================] - 106s 353ms/step - loss: 0.7716 - accuracy: 0.7385 - val_loss: 0.8263 - val_accuracy: 0.7096
Epoch 21/120
300/300 [==============================] - 105s 351ms/step - loss: 0.7305 - accuracy: 0.7516 - val_loss: 0.8279 - val_accuracy: 0.7029
Epoch 22/120
300/300 [==============================] - 103s 345ms/step - loss: 0.7269 - accuracy: 0.7432 - val_loss: 1.1657 - val_accuracy: 0.6229
Epoch 23/120
300/300 [==============================] - 109s 363ms/step - loss: 0.7009 - accuracy: 0.7641 - val_loss: 0.7812 - val_accuracy: 0.7400
Epoch 24/120
300/300 [==============================] - 105s 351ms/step - loss: 0.6946 - accuracy: 0.7666 - val_loss: 0.6629 - val_accuracy: 0.7763
Epoch 25/120
300/300 [==============================] - 105s 351ms/step - loss: 0.7034 - accuracy: 0.7625 - val_loss: 0.8012 - val_accuracy: 0.7229
Epoch 26/120
300/300 [==============================] - 105s 351ms/step - loss: 0.6846 - accuracy: 0.7629 - val_loss: 0.7620 - val_accuracy: 0.7467
Epoch 27/120
300/300 [==============================] - 106s 352ms/step - loss: 0.6541 - accuracy: 0.7784 - val_loss: 1.0184 - val_accuracy: 0.6513
Epoch 28/120
300/300 [==============================] - 106s 354ms/step - loss: 0.6362 - accuracy: 0.7796 - val_loss: 0.6848 - val_accuracy: 0.7667
Epoch 29/120
300/300 [==============================] - 113s 378ms/step - loss: 0.6351 - accuracy: 0.7800 - val_loss: 0.6355 - val_accuracy: 0.7746
Epoch 30/120
300/300 [==============================] - 110s 366ms/step - loss: 0.6276 - accuracy: 0.7799 - val_loss: 0.7055 - val_accuracy: 0.7596
Epoch 31/120
300/300 [==============================] - 107s 356ms/step - loss: 0.6390 - accuracy: 0.7852 - val_loss: 0.6641 - val_accuracy: 0.7704
Epoch 32/120
300/300 [==============================] - 110s 365ms/step - loss: 0.5932 - accuracy: 0.7952 - val_loss: 0.9903 - val_accuracy: 0.6725
Epoch 33/120
300/300 [==============================] - 104s 346ms/step - loss: 0.6049 - accuracy: 0.7957 - val_loss: 0.7252 - val_accuracy: 0.7563
Epoch 34/120
300/300 [==============================] - 108s 359ms/step - loss: 0.5888 - accuracy: 0.7973 - val_loss: 0.5749 - val_accuracy: 0.8067
Epoch 35/120
300/300 [==============================] - 110s 367ms/step - loss: 0.5856 - accuracy: 0.8005 - val_loss: 0.6746 - val_accuracy: 0.7758
Epoch 36/120
300/300 [==============================] - 114s 380ms/step - loss: 0.5760 - accuracy: 0.8006 - val_loss: 0.6491 - val_accuracy: 0.7817
Epoch 37/120
300/300 [==============================] - 113s 377ms/step - loss: 0.5395 - accuracy: 0.8149 - val_loss: 0.8643 - val_accuracy: 0.7092
Epoch 38/120
300/300 [==============================] - 117s 390ms/step - loss: 0.5467 - accuracy: 0.8144 - val_loss: 0.5713 - val_accuracy: 0.8121
Epoch 39/120
300/300 [==============================] - 117s 389ms/step - loss: 0.5503 - accuracy: 0.8117 - val_loss: 0.7791 - val_accuracy: 0.7342
```

Figure 14: Instance of the model Execution

Table 6: Results Obtained for Trail 4

epoch	Train Loss	Training Accuracy	Validation Loss	Validation Accuracy
Epoch 1	2.6339	0.3219	1.5816	0.5117
Epoch 2	1.78	0.4532	1.1585	0.5996
Epoch 3	1.3763	0.5284	1.794	0.3717
Epoch 4	1.2575	0.5579	0.9993	0.6396
Epoch 5	1.1548	0.5946	1.3852	0.4925
Epoch 6	1.072	0.622	0.9695	0.6433
Epoch 7	1.0537	0.6297	0.9069	0.6758
Epoch 8	0.9847	0.6499	1.0334	0.6562
Epoch 9	0.9757	0.6522	1.2586	0.5483

epoch	Train Loss	Training Accuracy	Validation Loss	Validation Accuracy
Epoch 10	0.9739	0.6547	0.8879	0.6825
Epoch 11	0.9233	0.6744	1.3096	0.5742
Epoch 12	0.9064	0.6854	0.8135	0.7183
Epoch 13	0.8651	0.7002	0.865	0.6963
Epoch 14	0.8685	0.7036	0.7813	0.7333
Epoch 15	0.824	0.7171	0.8198	0.7133
Epoch 16	0.7993	0.7256	0.7605	0.7383
Epoch 17	0.8097	0.7178	0.9414	0.6687
Epoch 18	0.7675	0.7352	0.7191	0.7554
Epoch 19	0.7767	0.7321	0.6853	0.7575
Epoch 20	0.7444	0.7494	1.168	0.5888
Epoch 21	0.7355	0.7473	0.9079	0.6862
Epoch 22	0.6991	0.7577	0.8447	0.7208
Epoch 23	0.7127	0.755	0.8651	0.7075
Epoch 24	0.7095	0.7535	0.7418	0.7492
Epoch 25	0.6899	0.7665	0.8187	0.7138
Epoch 26	0.6681	0.7689	1.029	0.6454
Epoch 27	0.654	0.7798	1.042	0.6321
Epoch 28	0.6387	0.7791	0.5752	0.8071
Epoch 29	0.6168	0.7871	0.8333	0.7058
Epoch 30	0.5961	0.7981	0.5845	0.8042
Epoch 31	0.6341	0.7824	0.6042	0.785
Epoch 32	0.5969	0.7977	1.0609	0.6308
Epoch 33	0.603	0.7951	0.7234	0.7563
Epoch 34	0.5923	0.795	0.5991	0.8033
Epoch 35	0.5903	0.8018	0.5221	0.8225
Epoch 36	0.5553	0.8091	1.0563	0.6692
Epoch 37	0.554	0.8058	0.5399	0.8179
Epoch 38	0.5333	0.8192	0.6113	0.7971
Epoch 39	0.5289	0.8205	0.5519	0.8104
Epoch 40	0.5595	0.8114	1.0869	0.6704
Epoch 41	0.5382	0.822	0.5985	0.7992
Epoch 42	0.5421	0.8209	0.5146	0.8304
Epoch 43	0.5066	0.8275	0.847	0.7271
Epoch 44	0.5319	0.8191	0.6967	0.7679
Epoch 45	0.5187	0.8244	0.6784	0.7696
Epoch 46	0.4999	0.8308	0.636	0.7937
Epoch 47	0.4944	0.8285	0.5459	0.8142
Epoch 48	0.507	0.8259	0.6995	0.7696
Epoch 49	0.4846	0.835	0.5105	0.8342
Epoch 50	0.4857	0.8356	0.6345	0.7908
Epoch 51	0.4969	0.8326	0.6305	0.7992

epoch	Train Loss	Training Accuracy	Validation Loss	Validation Accuracy
Epoch 52	0.4813	0.8327	0.8517	0.7417
Epoch 53	0.4729	0.8473	0.6311	0.7925
Epoch 54	0.4706	0.8389	0.5609	0.8183
Epoch 55	0.4694	0.8427	0.6692	0.7908
Epoch 56	0.49	0.8365	0.5264	0.8225
Epoch 57	0.4636	0.8392	0.602	0.7992
Epoch 58	0.4446	0.847	0.5378	0.8279
Epoch 59	0.4662	0.8386	0.61	0.8058
Epoch 60	0.4437	0.8556	0.4955	0.84
Epoch 61	0.4437	0.8504	0.6466	0.7971
Epoch 62	0.4577	0.846	0.5432	0.8254
Epoch 63	0.4338	0.8515	0.5066	0.8358
Epoch 64	0.4246	0.856	0.4839	0.8492
Epoch 65	0.4197	0.8565	0.5452	0.8225
Epoch 66	0.4508	0.8467	0.5254	0.8221
Epoch 67	0.4489	0.8521	0.554	0.8117
Epoch 68	0.4271	0.8599	0.5779	0.8104
Epoch 69	0.4172	0.8596	0.5263	0.84
Epoch 70	0.433	0.8526	0.4808	0.8496
Epoch 71	0.4261	0.858	0.5788	0.8133
Epoch 72	0.4262	0.8529	0.6132	0.8112
Epoch 73	0.4144	0.8593	0.483	0.8458
Epoch 74	0.3972	0.8672	0.5111	0.8371
Epoch 75	0.4185	0.8603	0.5285	0.8396
Epoch 76	0.4162	0.8613	0.4888	0.8471
Epoch 77	0.3848	0.866	0.5006	0.8408
Epoch 78	0.3889	0.8719	0.5524	0.825
Epoch 79	0.3911	0.8657	0.6269	0.8104
Epoch 80	0.3901	0.8643	0.525	0.8283
Epoch 81	0.3804	0.8689	0.5681	0.8313
Epoch 82	0.3868	0.8704	0.5835	0.8188
Epoch 83	0.3915	0.8662	0.5027	0.8354
Epoch 84	0.3889	0.8745	0.6683	0.7987
Epoch 85	0.3922	0.8657	0.4821	0.85
Epoch 86	0.3731	0.8726	0.5398	0.8367
Epoch 87	0.3624	0.8809	0.558	0.8246
Epoch 88	0.3765	0.8663	0.4285	0.8683
Epoch 89	0.3628	0.8746	0.4616	0.8575
Epoch 90	0.355	0.8819	0.5077	0.8425
Epoch 91	0.3898	0.8654	0.6623	0.7929
Epoch 92	0.3656	0.8764	0.4645	0.855
Epoch 93	0.3629	0.8783	0.5904	0.8025

epoch	Train Loss	Training Accuracy	Validation Loss	Validation Accuracy
Epoch 94	0.3677	0.8774	0.5499	0.8354
Epoch 95	0.3763	0.8725	0.5361	0.8329
Epoch 96	0.3652	0.8763	0.5771	0.8292
Epoch 97	0.3392	0.8869	0.7828	0.7688
Epoch 98	0.365	0.8759	0.7318	0.7829
Epoch 99	0.3589	0.8774	0.6036	0.8158
Epoch 100	0.3466	0.878	1.1669	0.6942
Epoch 101	0.3494	0.8786	0.5358	0.8321
Epoch 102	0.3544	0.8767	0.6673	0.8025
Epoch 103	0.3424	0.8858	0.5444	0.8379
Epoch 104	0.3512	0.8795	0.529	0.8475
Epoch 105	0.3356	0.8852	0.5627	0.825
Epoch 106	0.3338	0.887	0.4998	0.8583
Epoch 107	0.3299	0.8877	0.5959	0.82
Epoch 108	0.3454	0.8822	0.5267	0.8396
Epoch 109	0.3253	0.8908	0.5881	0.8213
Epoch 110	0.3096	0.8953	0.4979	0.8546
Epoch 111	0.3503	0.8831	0.5069	0.8413
Epoch 112	0.3143	0.8947	0.4623	0.8592
Epoch 113	0.3288	0.8854	0.4361	0.8696
Epoch 114	0.3167	0.8896	0.4739	0.8508
Epoch 115	0.3225	0.8919	0.4361	0.8696
Epoch 116	0.3242	0.8921	0.4223	0.8796
Epoch 117	0.3239	0.893	0.4201	0.8802
Epoch 118	0.3111	0.8966	0.4199	0.8911
Epoch	0.298	0.9064	0.4061	0.8996

epoch	Train Loss	Training Accuracy	Validation Loss	Validation Accuracy
119				
Epoch 120	0.29	0.9143	0.4011	0.9101

Table 7: Accuracy of the Model

Model	CIFAR-10 Dataset Result	Proposed Model Accuracy	
		Training	Validation
Baseline + Dropout	83.450%	76.84%	78.25%
Baseline + Data Augmentation	84.470%	96.02% (Over fitting)	73.83%
Baseline + Dropout + Data Augmentation	85.880%	89.23%	81.92%
Baseline + Increasing Dropout + Data Augmentation + Batch Norm	88.620%	90.15%	90.66%

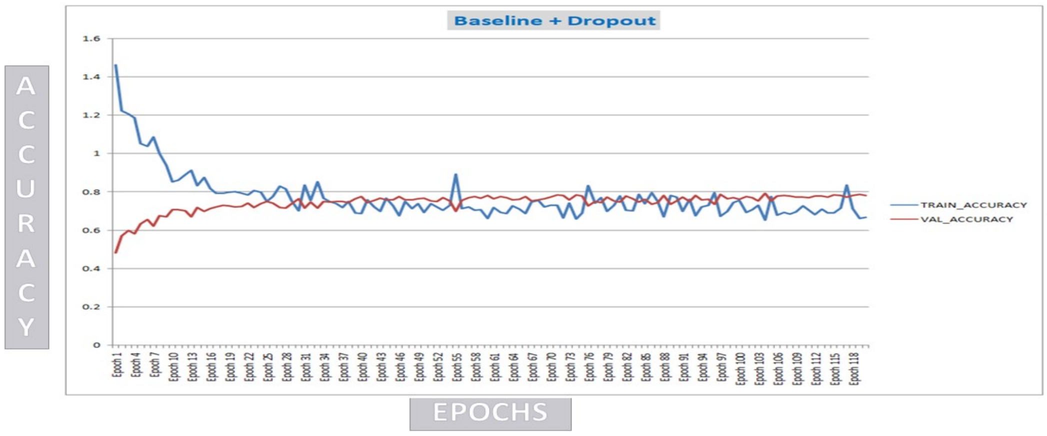


Figure 15: Accuracy Visualization (Baseline + Dropout Trial-1 results)

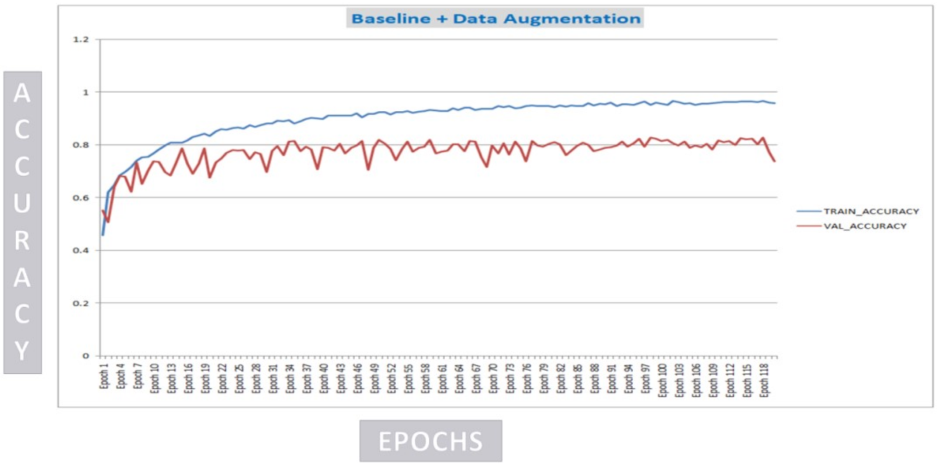


Figure 16: Accuracy Visualization (Baseline + Data Augmentation Trial-2 results)

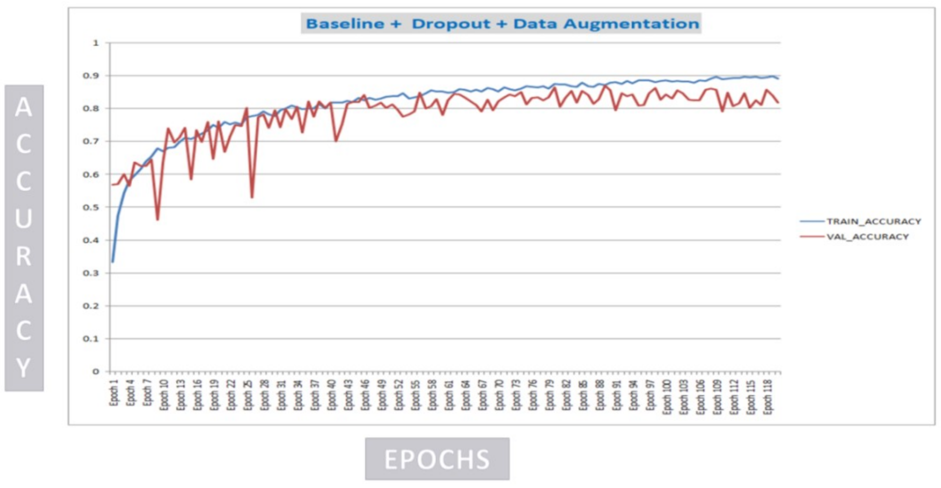


Figure 17:Accuracy Visualization (Baseline + Dropout + Data Augmentation Trial-3 results)

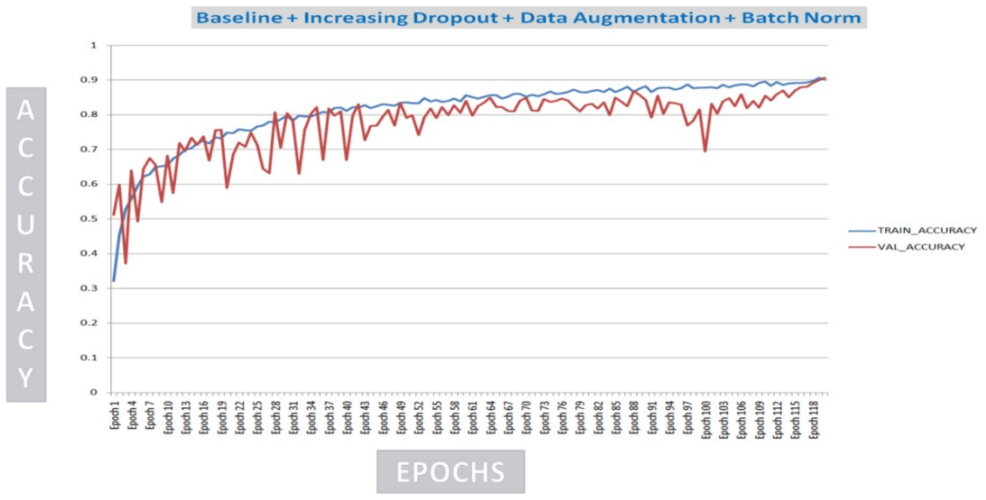


Figure 18: Accuracy Visualization (Baseline + Increasing Dropout + Data Augmentation + Batch Norm Trial-4 results)

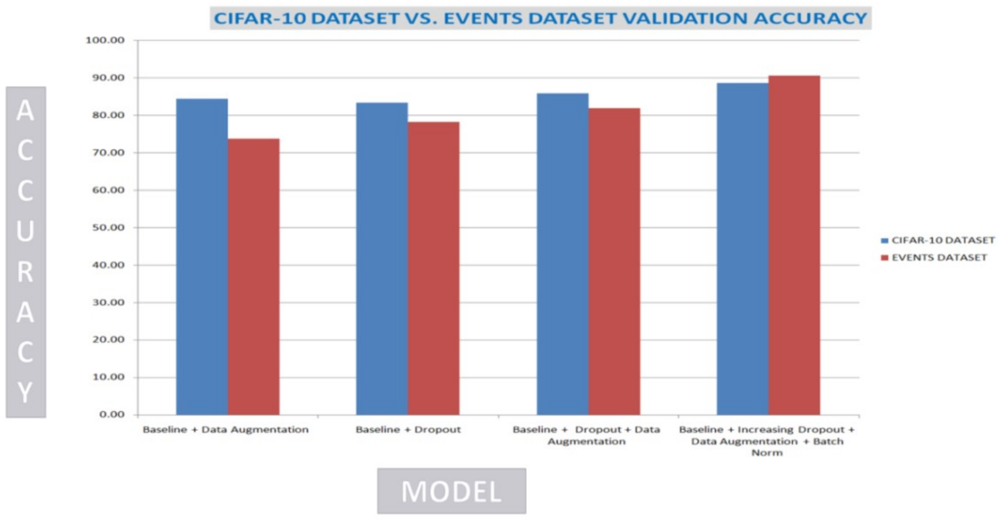


Figure 19: Proposed CNN model Accuracy Comparison

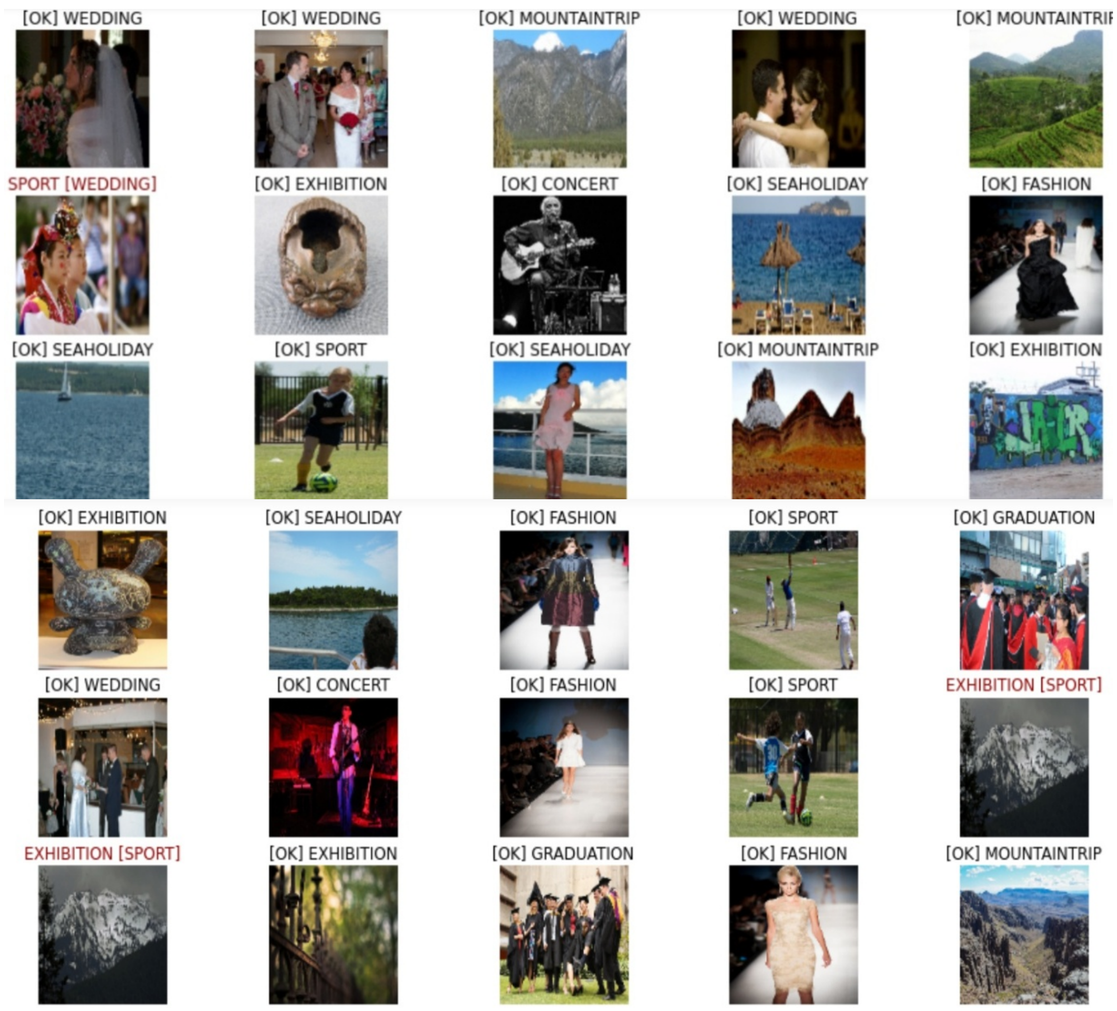


Figure 20: Output of the Proposed Image Classification CNN Model.

X. SUMMARY

This chapter illustrates the process of classification of images based on the event and in this process the base model is implemented with convolution, Relu and Pooling which is identified that the batch normalization process increaseswhen the dropout is identified to be improved with the result of training accuracy and validation accuracy and the proposed model worked better than the existing model.

This chapter is implemented based on "CIFAR-10" dataset which is utilized by most of the researchers for performing multi-class classification. Here 8 events of photos are used for classifying images using the modified CNN is used with 4 hidden layers are used. In every model there will be negligible outliers SPORT[WEDDING] and EXHIBITION[SPORT]obtained incorrect result as the model predicted photo of sport as wedding and one photo of exhibition as sport as some photos comprises of confusion even for a human eye is also confusing in such cases the proposed model has failed.

REFERENCES

[1] Ogu, Reginald &Ikerionwu, Charles &Ayogu, Ikechukwu. (2021). Leveraging Artificial Intelligence of Things for Anomaly Detection in Advanced Metering Infrastructures. 16-20. 10.1109/CYBERNIGERIA51635.2021.9428792.

[2] D. Parikh and C. L. Zitnick, "The role of features, algorithms and data in visual recognition," 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2010, pp. 2328-2335, doi: 10.1109/CVPR.2010.5539920.

[3] Capra, M. et al. An updated survey of efficient hardware architectures for accelerating deep convolutional neural networks. Future Internet 12, 113 (2020).

[4] Kouretas, I. &Paliouras, V. Hardware implementation of a softmax-like function for deep learning. Technologies 8, 46 (2020).

[5] Xiong, Y., Zhu, K., Lin, D., Tang, X.: Recognize complex events from static images by fusing deep channels. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1600 1609 (2015)

[6] S. Y. Chaganti, I. Nanda, K. R. Pandi, T. G. N. R. S. N. Prudhvith and N. Kumar, "Image Classification using SVM and CNN," 2020 International Conference on Computer Science, Engineering and Applications (ICCSEA), 2020, pp. 1-5, doi: 10.1109/ICCSEA49143.2020.9132851.

[7] D. Wang, H. Yu, D. Wang and G. Li, "Face Recognition System Based on CNN," 2020 International Conference on Computer Information and Big Data Applications (CIBDA), 2020, pp. 470-473, doi: 10.1109/CIBDA50819.2020.00111.

[8] Savchenko A.V., Miasnikov E.V. (2020) Event Recognition Based on Classification of Generated Image Captions. In: Berthold M., Feelders A., Krempl G. (eds) Advances in Intelligent Data Analysis XVIII. IDA 2020. Lecture Notes in Computer Science, vol 12080. Springer, Cham.

[9] Sandler, M., Howard, A.G., Zhu, M., Zhmoginov, A., Chen, L.: MobileNetV2: inverted residuals and linear bottlenecks. In: Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4510 4520. IEEE (2018)

[10] Goodfellow, Ian; Bengio, Yoshua; Courville, Aaron (2016). "6.2.2.3 Softmax Units for Multinoulli Output Distributions". Deep Learning. MIT Press. pp. 180 184. ISBN 978-0-26203561-3.

[11] R. Doon, T. Kumar Rawat and S. Gautam, "Cifar-10 Classification using Deep Convolutional Neural Network," 2018 IEEE Punecon, 2018, pp. 1- 5, doi: 10.1109/PUNECON.2018.8745428.

[12] K. Ishizaki, K. Saruta and H. Uehara, "Detecting Keypoints for Automated Annotation of Bounding Boxes using Keypoint Extraction," 2020 International Conference on Computational Science and Computational Intelligence (CSCI), 2020, pp. 1691-1694, doi: 10.1109/CSCI51800.2020.003