**Googledrive URL:** [**https://drive.google.com/drive/folders/1KdyoYzSAGYQ4DzKPbZp-MVqLuxqAL2ou?usp=sharing**](https://drive.google.com/drive/folders/1KdyoYzSAGYQ4DzKPbZp-MVqLuxqAL2ou?usp=sharing) **Github URL :** [**https://github.com/overtunned/Object-classification-using-cnn-motorcycle.git**](https://github.com/overtunned/Object-classification-using-cnn-motorcycle.git) **Lab Exercise Set-5**

**Roll No : CB.EN.P2AID20002**

**Name : Abhishek Gopinath K**

**================================================================================**

**Title : Object Identification in CNN – Motor Bike**

**Image level Description**

|  |  |
| --- | --- |
| **Feature** | **Description** |
| **Size** | **< 1mb** |
| **Resolution** | **96dpi** |
| **Bit Depth** | **24 (8 R, 8 G, 8 B)** |
| **Type** | **JPG** |

**Dataset Description**

URL of the dataset: http://www.vision.caltech.edu/html-files/archive.html

Dataset contains 1755 positive images which contains a motorbike and 2315 negative images in which motorbike is absent.

* The images are of different dimensions and hence we need to normalize it to feed into our model. Here, the all the images are divided by 255 to scale the values of the pixels to 0 to 1. The images are also resized to 224 x 224 x 3.
* The images of the dataset had only side images so have to apply data augmentation with transformations.

Then the images are split into train, test and validation sets in a ratio of **7:2:1**, respectively.

No of training images : (2849, 227, 227, 3)

No of testing images : (814, 227, 227, 3)

No of Validation images : (407, 227, 227, 3)

Class level details : There are only 2 classes

* 0 - No Motorbike
* 1 - Motorbike

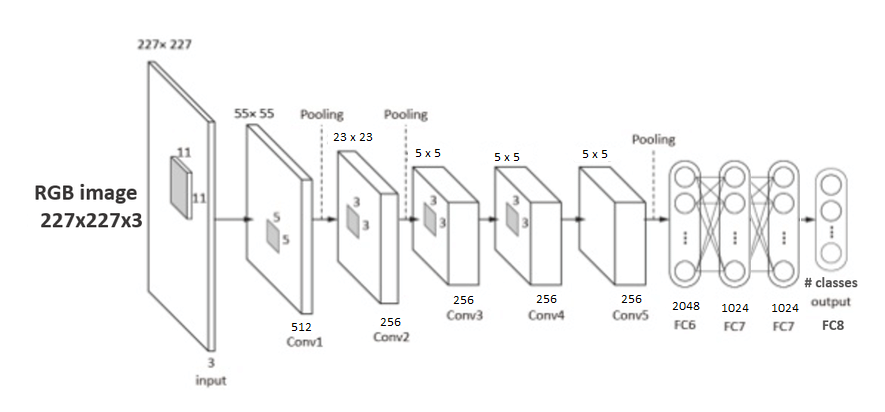
**Provide a description of scene images where this object will occur:**

The images consist of scene where there is motorbike parked somewhere, someone riding it, a rider doing stunts on it. As for negative images it consists of images that contains something that is similar toa motor bike like a cycle or wheels which is either stationary or being rode on.

Sample images of scene:

|  |  |
| --- | --- |
| **Scene Image 1** | **List the Questions that can be answered from the image** |
|  | Color of the cycle?  Is someone riding it?  Is it upright?  Is it lying on the floor?  How many wheels? |
|  | Color of the cycle?  Is someone riding it?  Is it upright?  Is it lying on the floor?  How many wheels? |
|  | Color of the cycle?  Is someone riding it?  Is it upright?  Is it lying on the floor?  How many wheels? |
|  | Color of the cycle?  Is someone riding it?  Is it upright?  Is it lying on the floor?  How many wheels? |
|  | Color of the cycle?  Is someone riding it?  Is it upright?  Is it lying on the floor?  How many wheels? |
|  | Color of the cycle?  Is someone riding it?  Is it upright?  Is it lying on the floor?  How many wheels? |

**Convolution Neural Network**



**Working principles of CNN**

**Convolutional layer**:

A convolution is a mathematical term that describes a dot product multiplication between two sets of elements. Within deep learning the convolution operation acts on the filters/kernels and image data array within the convolutional layer. Therefore, a convolutional layer is simply a layer of the houses of convolution operations that occurs between the filters and the images passed through a convolutional neural network.

**Batch Normalisation layer**:

Batch Normalization is a technique that mitigates the effect of unstable gradients within a neural network through the introduction of an additional layer that performs operations on the inputs from the previous layer. The operations standardize and normalize the input values, after that the input values are transformed through scaling and shifting operations.

**MaxPooling layer**:

Max pooling is a variant of sub-sampling where the maximum pixel value of pixels that fall within the receptive field of a unit within a sub-sampling layer is taken as the output. The max-pooling operation below has a window of 2x2 and slides across the input data, outputting an average of the pixels within the receptive field of the kernel.

**Flatten Layer**:

Takes an input shape and flattens the input image data into a one-dimensional array.

**Dense Layer**:

A dense layer has an embedded number of arbitrary units/neurons within. Each neuron is a perceptron.

**Dropout Layer**:

The Dropout layer randomly sets input units to 0 with a frequency of rate at each step during training time, which helps prevent overfitting. Inputs not set to 0 are scaled up by 1/(1 - rate) such that the sum over all inputs is unchanged.

**Program Environment:**

**Framework :** Keras and Tensorflow

**Programming Language :** Python

**Editor :** Google Colaboratory

**Libraries used :** keras, math, os, numpy, cv2, glob, random, matplotlib, sklearn

**Definition:**

Write the definition of different parameters and hyperparameters with its units and expected value.

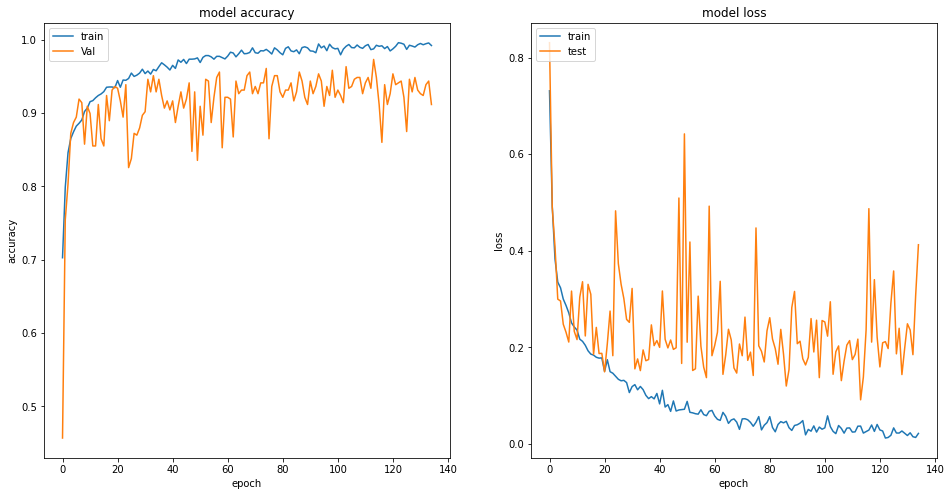
**Tabulation of Hyperparameters**

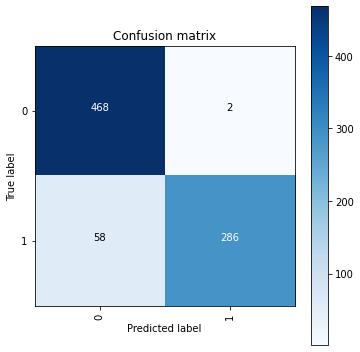
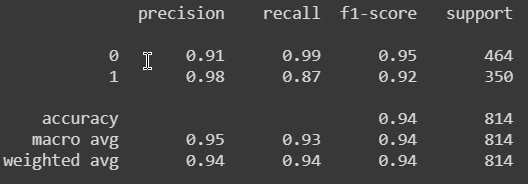
|  |  |
| --- | --- |
| **Hyperparameter name** | **Set Value in Architecture**  **(Units)** |
| No of hidden layers | 5 conv2d and 3 dense layers |
| Activation function | Relu and softmax |
| Drop out | 3 with 0.5 probability |
| Learning Rate | 0.0001 |
| Epochs | 135 |
| Batch size | 64 |

**Model performance**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Model No** | **Hyperparameter Estimates**  **(For ex: no of conv layer=2,**  **Activation function=3…)** | **Execution Time(unit)**  **Seconds per epoch** | **Accuracy** | **class** | **Precision** | **Recall** | **F1-score** |
| **1** | **Batch size = 10**  **Learning rate = 0.001**  **Epochs = 10** | **8s** | **0.86** | **0**  **1** | **0.80**  **0.99** | **1.00 0.66** | **0.89 0.80** |
| **2** | **Batch size = 20**  **Learning rate = 0.0001**  **Epochs = 30** | **7s** | **0.93** | **0**  **1** | **0.96**  **0.91** | **0.93**  **0.94** | **0.94**  **0.92** |
| **3** | **Batch size = 30**  **Learning rate = 0.001**  **Epochs = 50** | **6s** | **0.94** | **0**  **1** | **0.93**  **0.96** | **0.97**  **0.90** | **0.95**  **0.93** |
| **4** | **Batch size = 40**  **Learning rate = 0.0001**  **Epochs = 100** | **6s** | **0.94** | **0**  **1** | **0.94**  **0.95** | **0.96**  **0.92** | **0.95**  **0.93** |
| **5** | **Batch size = 50**  **Learning rate = 0.0001**  **Epochs = 135** | **26s** | **0.93** | **0**  **1** | **0.90 0.98** | **0.99 0.85** | **0.94**  **0.91** |

**Performance Measures**

Training accuracy: 0.9924 and Training loss: 0.0224  
Validation accuracy: 0.9115 and Validation loss: 0.4125  
Test accuracy: 0.9262 and Test loss: 0.2659

**Confusion matrix:**  
  
**Classification report:**  


**List of Similar works done in same dataset in internet**

<https://github.com/bhavul/Caltech-101-Object-Classification>  
<https://github.com/ronghuaiyang/faster-rcnn-caltech-pedestrian-pytorch>

**References**

1.Dataset references  
<http://www.vision.caltech.edu/Image_Datasets/Caltech256>  
<http://www.vision.caltech.edu/Image_Datasets/Caltech101/101_ObjectCategories.tar.gz>

2.Other Web reference  
<https://ieeexplore.ieee.org/document/7539822>  
<https://www.researchgate.net/publication/327279970_Motorcycle_Classification_in_Urban_Scenarios_using_Convolutional_Neural_Networks_for_Feature_Extraction>  
<https://www.researchgate.net/publication/326892213_Motorcycle_detection_and_classification_in_urban_Scenarios_using_a_model_based_on_Faster_R-CNN>