1. INTRODUCTION

The evolution of Computer Vision and Machine Learning(ML) techniques have unlocked myriad possibilities in today's world. The success of deep neural networks and other ML algorithms in applications like autonomous vehicle driving, speech recognition, image recognitions etc. attests to the value of using such methods to tackle complex real-world problems.

One of the prominent feature of Computer Vision includes image recognition. In this, presence of any object in an image is confirmed and then attempt is made for recognition. Our project, "**Object Recognition and Image Enhancement for Night Vision Surveillance"** is a prominent example of Computer Vision application.

"**Object Recognition**" here means the ability of the system to isolate specific features from the image and then classify them in some predefined categories such that the presence of the object is indicated. "**Night Vision**" signifies that the images will be either underexposed or will be taken from special "Night Vision Camera" in complete darkness which will needs some form of enhancement such that the system is able to further process it.

1.1. Objectives

i. To implement image processing algorithms to enhance images taken in low light.
ii. To develop classification model for accurately identifying objects in the low light.
iii. To compare the classification results using different image processing algorithms.

2. LITERATURE REVIEW

The aim of this project is to generate a model that can enhance images taken in low-light situation and also correctly classify some objects in the frame of image using machine learning. The project also focuses on studying the effects of different enhancement algorithms on improving the classification accuracy of classifiers.
Prior studies in the field of enhancing images in low-light have provided with some methods that can be applied to improve characteristics of images.

2.1. Image Enhancement

Y. Tendero, S. Landeau, and J. Gilles, "Non-uniformity Correction of Infrared Images by Midway Equalization," Image Processing On Line, vol. 2, pp. 134–146, Dec. 2012.
Y. Tendero, S. Landeau, and J. Gilles (2012) derived method of correcting Non-Uniformity in infrared images. The non-uniformity is a time-dependent noise caused by the lack of sensor equalization. This method was designed to suit infrared images. Nevertheless, it can be applied to images produced for example by scanners, or by push-broom satellites. This single image method works on static images, is fully automatic, has no user parameter, and requires no registration. It needs no camera motion compensation, no closed aperture sensor equalization and is able to correct for a fully non-linear non-uniformity.

P. Getreuer, "Automatic Color Enhancement (ACE) and its Fast Implementation," Image Processing On Line, vol. 2, pp. 266–277, Jun. 2012.
P. Getreuer (2012) has provided with algorithm for Automatic Color Enhancement (ACE) which was introduced by Gatta, Rizzi, and Marini based on modeling several low level mechanisms of the human visual system. P. Getreuer has successfully reduced the computational complexity of ACE on an NxN from $O(N^4)$ to $O(N^2 \log N)$ using polynomic approximations of the slope function decomposing the main computation into convolutions.

"OpenCV: Histograms - 2: Histogram Equalization," OpenCV: Histograms - 2: Histogram Equalization. [Online]. Available:

http://docs.opencv.org/3.1.0/d5/daf/tutorial_py_histogram_equalization.html. [Accessed: 10-May-2017].

Methods such as Histogram Equalization and CLAHE (Contrast Limited Adaptive Histogram Equalization) can serve as valuable tool in boosting the contrast of foggy or nearly uniform contrast images.

2.2. Object Recognition and Classifier

A number of machine learning techniques have been studied and developed for the purpose of recognition and classification of image.

A popular traditional method was the classification of HOGs (Histogram of Oriented Gradients) using SVM (support vector machine) for the purpose of object recognition which attained modest accuracy and performance.

With the advent of ANNs (Artificial Neural Network) and more automated methods like CNNs (Convolutional Neural Network), better accuracy and computational speed was achieved for image classification.

Krichevsky et al. "ImageNet Classification with Deep Convolutional Neural Networks", 2012

Krichevsky et al. (2012) made the use of GPU for training of the tweaked CNN model on the subsets of ImageNet ILSVRC-2012 and achieved the best result. The model was named "AlexNet". The size of the model possessed a risk of over-fitting of the model even for the 1.2 million training tuples which was mitigated by using several data augmentation and regularization techniques. The final network contained five convolutional and three fully-connected layers, and this depth seems to be important: they found that removing any convolutional layer (each of which contains no more than 1 per cent of the model's parameters) resulted in inferior performance and also the training time is drastically reduced using RELU activation rather than conventional Tanh function.

Subsequent generations of state of art were based on the Krichevsky's AlexNet model. A fairly new approach for the CNN architecture was made by "GoogLeNet".

Szegedy et al. "Going Deeper with Convolution", 2015

C. Szegedy (Google Inc.) and team worked on a CNN architecture that become the winner of ILSVRC-2014. The model named "GoogLeNet" is 22-layer deep and involves use of 9 "Inception Modules" which is a framework containing multiple pooling and convolution operations in parallel. The inception essentially increased the network depth. The model however lacks fully connected layers as in more traditional CNNs and thus use about 12 times fewer parameters than the "AlexNet". The "GoogLeNet" was the first model to introduce the idea that CNNs do not always have to involve sequentially stacked layers and structure them in parallel for improved performance and computational complexity.In 2015, Google released an open source library called "TensorFlow" for numerical computation (involves mathematical computation functions for complex algorithms) using data flow graph (CNN structure is actually define in terms of graph). The library provides a CNN trained on the subset of ImageNet dataset; "Inception V3" model which is improvement over the initial "GoogLeNet" architecture. TensorFlow also provides method using the penultimate layers of this model as feature extractor which is known as "Transfer Learning".

"CS231n Convolutional Neural Networks for Visual Recognition".[Online].Available: http://cs231n.github.io/ [Accessed: 10th May, 2017]

An online notes that accompany the Stanford CS class CS231n prepared by Andrej Kaparthy. The notes have good explanation of the CNN layers and functions.

It is not practical to fully train a CNN from scratch (with random initializations) regarding data and computational limits. Transfer Learning is process of taking a pre-trained layer and optimize it with custom dataset. The CS231n notes define 2 methods: using ConvNet as feature extractor (where the final fully connected or prediction layers of CNN are replaced by fresh layers and retrained) or by Fine Tuning the ConvNet (where entire network or a subset of layers are fine-tuned with custom dataset rather than just the final layers).

3. METHODOLOGY

3.1. Tools Used

3.1.1. Infra-Red Cameras

Infra-red Cameras deal with imaging using Infrared Radiation Band (700 nm to 14 μm) rather than the Visible Band in Electromagnetic Spectrum (400 to 700 nm). The basic principle of an infrared camera is the detection of the infrared energy (which is the function of temperature) being emitted by the bodies in the surrounding and its conversion into electrical form. A brighter spot in the image taken from an IR camera implies the point or region with high temperature irrespective of degree of visibility of the surrounding. The images taken from a IR camera is monochromatic. Primarily, infrared imaging is used in night vision applications and building inspections.

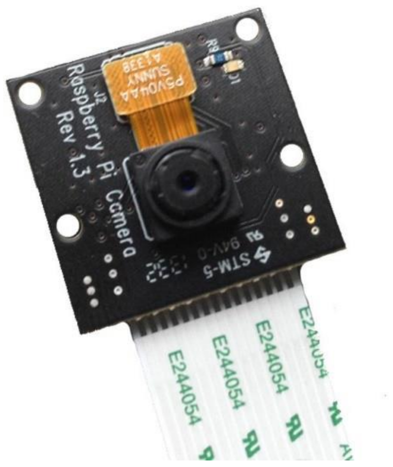


Figure 3. 1: IR Camera Module

3.1.2. Digital Image Processing

Image Processing refers to mathematical operations of signal processing where the input may be image, series of images, video or video frames. It basically deals with pixel-wise operation of images without extraction of meaningful information. The output of image processing may be either an image or a set of characteristics or parameters related to the image.

Digital Image processing is the use of computer algorithms to perform image processing on digital images.

3.1.3. Machine Learning using Neural Networks

Machine Learning enables computer programs to learn without being explicitly programmed. It evolved from the study of pattern recognition and computational learning theory in artificial learning.

In a broad sense, machine learning can be classified into two categories; supervised learning and unsupervised learning. Supervised Learning involves specific set of mathematical operations to minimize a cost function (or loss factor) determined by comparison of expected output to the actual output. Unsupervised Learning on the other hand, deals with extraction of pattern or meaningful information from a cluster of data.

The project aims to implement an appropriate neural network using machine learning for the detection (and also recognition) of a small set of objects.

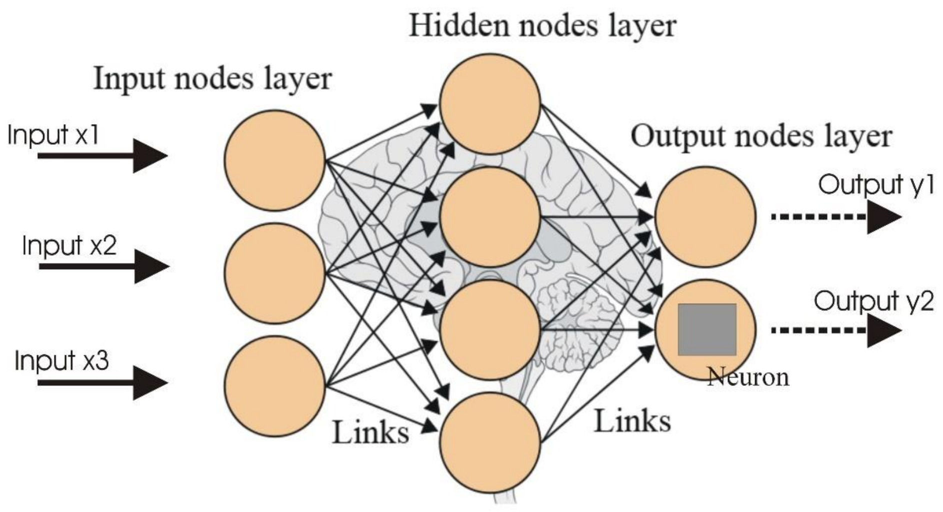


Figure 3. 2: A Three-Layered Neural Network

A neural network is an architecture of interconnected artificial neural nodes called “neurons” that exchange data between each other. The connections have numeric weights that are adjusted during the training process, so that a properly trained network will respond correctly when presented with an image or pattern to recognize. The network consists of multiple layers of feature-detecting “neurons”.

3.1.4. Raspberry PI

We have used Raspberry PI as our client unit. It is a lightweight computer system having ARM based processor. The specifications are:

- Name - Raspberry PI 3 Model B
- SoC: Broadcom BCM2837
- CPU: 4× ARM Cortex-A53, 1.2GHz
- GPU: Broadcom VideoCore IV
- RAM: 1GB LPDDR2 (900 MHz)
- Networking: 10/100 Ethernet, 2.4GHz 802.11n wireless
- Bluetooth: Bluetooth 4.1 Classic, Bluetooth Low Energy
- Storage: microSD
- GPIO: 40-pin header, populated
- Ports: HDMI, 3.5mm analogue audio-video jack, 4× USB 2.0, Ethernet, Camera Serial Interface (CSI), Display Serial Interface (DSI)
- Operating System - Raspbian (A debian based LINUX system)

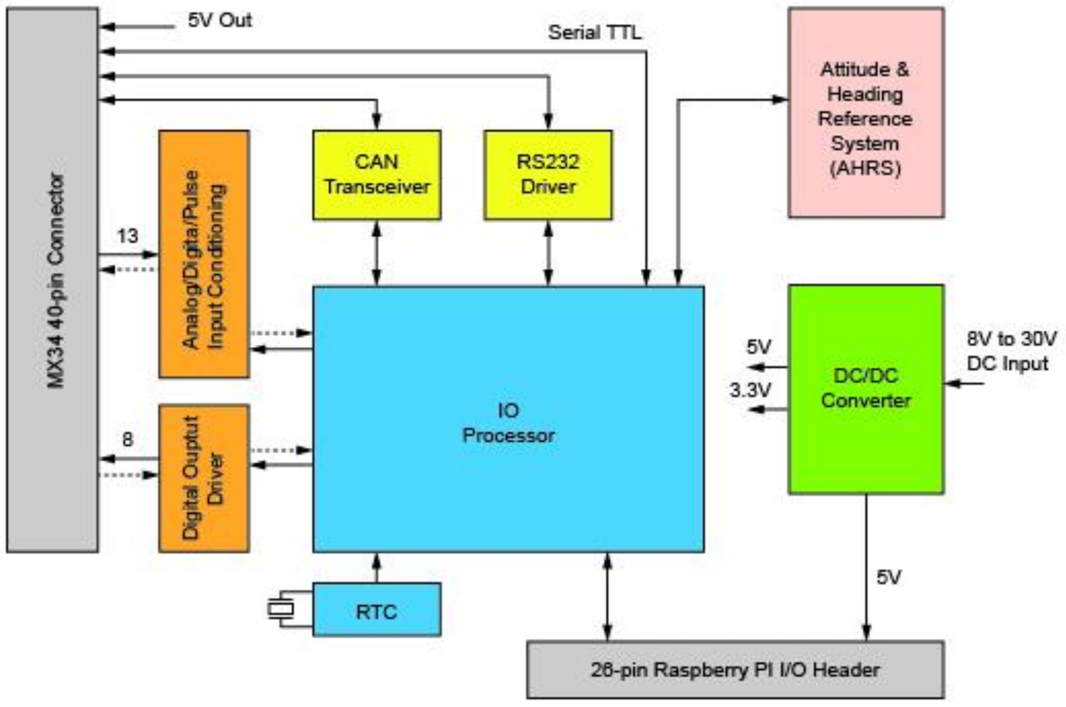


Figure 3. 3: Internal Block Diagram of RP

3.1.5. Computer Vision

Computer Vision (image understanding) refers to the techniques that mainly involves segmentation, recognition and reconstruction (3D models) which work together to give the ability of scene understanding to the computer. It deals with extraction of high-dimensional data from the real world in order to produce numerical or symbolic information. Therefore, meaningful information is extracted from image or image related parameters.

Computer Vision integrates both Image Processing and Machine Learning concepts. This enables it to accomplish many real-world applications with applicable and efficient methods which otherwise would involve myriad complications if needed to be hard coded.

After the Computer Vision techniques are applied, the end result will be detection of presence and if present, classification of the object. This classified data will be used to generate appropriate control signals which will be sent to the client interface. The interface will be designed to operate in accordance to the received control signal performing desired action.

3.1.6. GPU Processing

Implementation of machine learning algorithms in conventional serial processors is plagued by large number of challenges. The serial processors are extremely slow at processing multidimensional data such as matrix. For the efficient and real time application using machine learning algorithm, there is a need for parallel processing of large chunks of data. Graphics processors are very well suited for this operation.

Graphics cards are used in machine learning applications as there is increase in: the availability of massive amounts of training data, and powerful and efficient parallel computing provided by GPU computing. GPUs are used to train these deep neural networks using far larger training sets, in an order of magnitude less time, using far less datacenter infrastructure. GPUs are also being used to run these trained machine learning models to do classification and prediction in the cloud, supporting far more data volume and throughput with less power and infrastructure.

Figure 3. 4: GPU Chip

3.2.   System Architecture

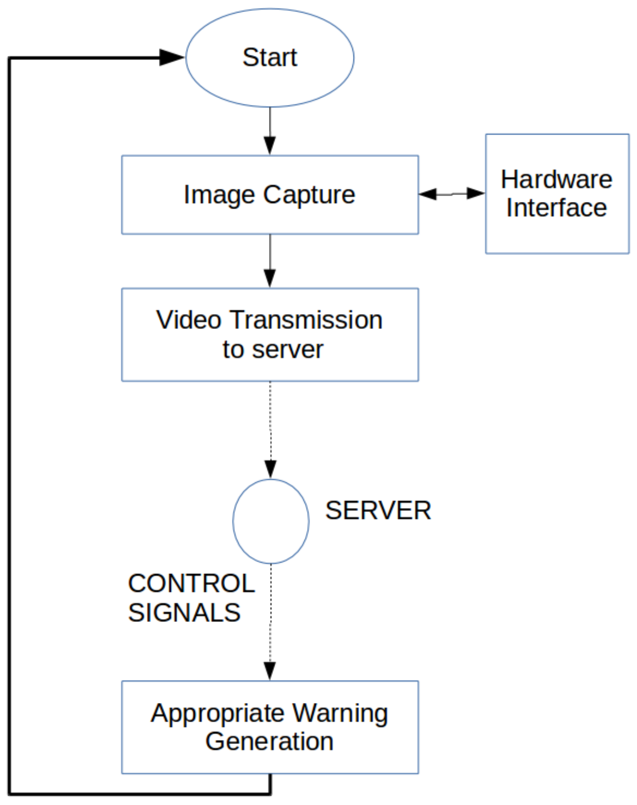


Figure 3. 5: Client Side Data Flow Diagram

The proposed model of the client-side architecture is shown in the diagram above. The client side constitutes of three major part, which are:

1. Camera interfacing with Raspberry pi.
2. Image transferring to the server for further manipulation.
3. Control signal reception.

The camera interfacing part and image transmission part has been successfully completed. A GUI (Graphical User Interface) has been developed in the client-side to easily control these two functions.

The image transmission part has been explained in detail in the following sections along with the block diagrams.

3.2.1. Image Transmission to Server

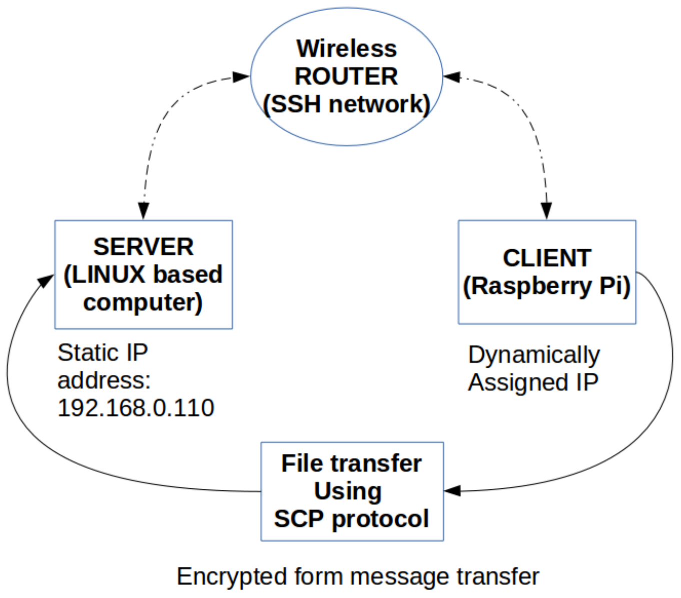

Wireless ROUTER (SSH network)
SERVER (LINUX based computer)
Static IP address: 192.168.0.110
CLIENT (Raspberry Pi)
Dynamically Assigned IP
File transfer Using SCP protocol
Encrypted form message transfer

Figure 3. 6: Image Transmission in Local Network

3.2.1.1. <u>Wireless Router</u>.

A wireless router is used to assign the client and server with IP addresses to enable file transmission between them. The server is assigned a static IP address to prevent the IP changing each time the server is connected to the router. The client however is allowed to have dynamic IP as we have only used one way transmission of file from client to server till now.

The client is accessed using SSH (Secure Shell) protocol. It is a network protocol that provides a secure way to access a remote computer. Secure Shell facilitates strong authentication and secure encrypted data communications between two computers connecting over an insecure network such as the Internet. Port 22 is used by default to run SSH services.

Steps involved in setting up SSH secure network:

1. Key-pair generation: Public key and private are generated in local computer. This is done using “keygen” command in linux. This generates a private key and sharable public key.

2. Transmitting public key to remote device: The public key obtained is transmitted to remote computer and stored in appropriate file location.
3. This will allow bypassing password with transmitting files over SSH network using SCP command.

<u>Client and Server Setup</u>.

The client and server are both LINUX based systems. The descriptions of the systems are:

▪ Client:

We have used Raspberry PI as our client unit. It is a lightweight computer system having ARM based processor. The specifications are:

- Name - Raspberry PI 3 Model B
- SoC: Broadcom BCM2837
- CPU: 4× ARM Cortex-A53, 1.2GHz
- GPU: Broadcom VideoCore IV
- RAM: 1GB LPDDR2 (900 MHz)
- Networking: 10/100 Ethernet, 2.4GHz 802.11n wireless
- Bluetooth: Bluetooth 4.1 Classic, Bluetooth Low Energy
- Storage: microSD
- GPIO: 40-pin header, populated
- Ports: HDMI, 3.5mm analogue audio-video jack, 4× USB 2.0, Ethernet, Camera Serial Interface (CSI), Display Serial Interface (DSI)
- Operating System - Raspbian (A debian based LINUX system)

A NoIR(No Infrared) camera of 8MP is interfaced with the client unit for image capture during night time.

▪ Server:

We have proposed on using a server having high computational capability for our project. However, as of now we are using our laptops as server until the proposed one is procured.

<u>SCP file transfer</u>.

Secure copy or SCP is a means of securely transferring computer files between a local host and a remote host or between two remote hosts. It is based on the Secure Shell (SSH) protocol. The SCP is a network protocol, based on the BSD RCP protocol, which supports file transfers between hosts on a network. SCP uses Secure Shell (SSH) for data transfer and uses the same mechanisms for authentication, thereby ensuring the authenticity and confidentiality of the data in transit. A client can send (upload) files to a server, optionally including their basic attributes (permissions, timestamps). Clients can also request files or directories from a server (download). SCP runs over TCP port 22 by default.

The command for transmitting file using SCP is:
scp SourceFile user@host:directory/TargetFile

To copy files, we use:
scp user@host:directory/SourceFile TargetFile
scp -r user@host:directory/SourceFolder TargetFolder

3.2.1. Camera Interfacing with Raspberry Pi

Based on the primary objective of the project, which is image enhancement and recognition during night time, a NoIR camera has been interfaced with Raspberry pi.
As the name suggests, a NoIR camera has no IR cut filter, which makes this camera unsuitable for daylight shots, but, if used with IR illuminators, we actually get Raspberry Pi night vision.
The camera interfaced in this project has some of the following hardware and software specifications:

- Small board size: 25mm * 24mm * 9mm (without IR illuminators)
- A 5MP omnivision sensor
- A pair of IR illuminators- one on each side
- Images- Upto 2592 x 1944 pixel static images
- Video Support - 1080p30, 720p60 and 640x480p60/90

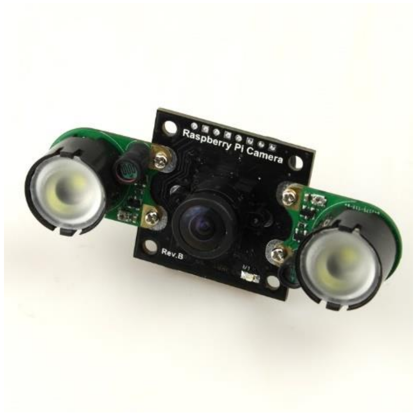


Figure 3. 7: Infra-Red Camera

The IR illuminator used with this camera module makes the camera capable of recording images at very low light, or even at absolute darkness. On that account, the quality of images captured also depends upon the intensity of IR light emitted by the illuminator. For the time being, this intensity has been kept constant and other external obstructions like light sources has been ignored.

Similar to the regular pi camera, this NoIR camera, along with the IR illuminators has been interfaced with the raspberry pi using the dedicated CSI interface. A unique feature of CSI interface includes extremely high data rate and transmission of pixel data.

<u>CSI Interface</u>.

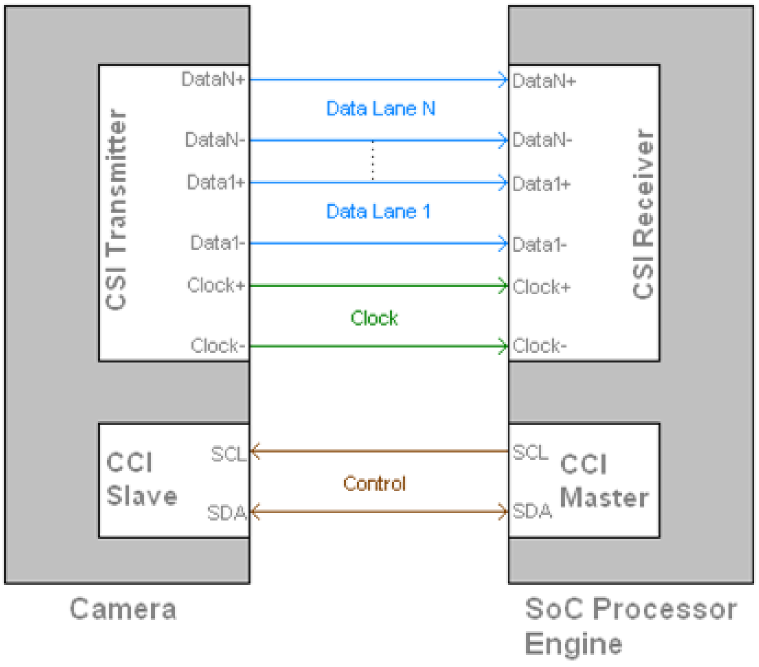


Figure 3. 8: Camera Serial Interface

The CSI specifications describes the physical layer known as D-PHY2. The signaling scheme of this physical layer, known as Low Voltage Differential Signaling (SubLVDS), is a modified version of the IEEE1596.3 LVDS specification. It is a system for low voltage 1.2 V applications, allowing data rates of up to 800 Mbps per lane with 1 Gbps set as a practical limit. In practice, the data rate can vary a lot and depends upon the quality of the interconnections. A maximum of four physical data lanes is allowable in this specification, however two are available for the Raspberry Pi.

This is a high-speed data communication bus and noise is of huge concern. Although this type of serial communication generates negligible crosstalk, the specification suggests using minimum clock rates for the camera module. The CSI transmission clock is source synchronous and the main processor may produce it instead to avoid noise interference on the camera module. The data transmission supports a wide range of data types such as RGB, RAW, YUV, generic, or byte based programmer defined.

For a 1 Gb bandwidth, the specification defines six pins consisting of two for data, two for control, and two for clock. For 2 Gb bandwidths, eight pins consisting of four for data (two lanes), two for control, and two for clock are required. There is also an additional pair of data lines for serial communication to set the bridge registers.

Raspberry Pi CSI Interface Connector Pinout

Table 3.1: Pin Configuration of Camera Serial Interface

S5 Pin	Name	Purpose
1	Ground	Ground
2	CAM1_DN0	Data Lane 0
3	CAM1_DP0	
4	Ground	Ground
5	CAM1_DN1	Data Lane 1
6	CAM1_DP1	
7	Ground	Ground
8	CAM1_CN	MIPI Clock
9	CAM1_CP	
10	Ground	Ground

11	CAM_GPIO	
12	CAM_CLK	
13	SCL0	I^{2}C Bus
14	SDA0	
15	+3.3 V	Power Supply

- CAM1_CN and CAM1_CP

  These pins provide the clock pulse for the MIPI data lanes for the first camera. They connect to the MIPI Clock Positive (MCP) and MIPI Clock negative (MCN) pins of the camera IC. These clock signals usually arrive from the camera module generated by the MIPI circuitry.

- CAM1_DN0 and CAM1_DP0

  These are the MIPI Data Positive (MDP), and MIPI Data negative (MDN) pins for the data lane 0 of camera 1.

- CAM1_DN1 and CAM1_DP1

  These are the MIPI Data Positive (MDP), and MIPI Data negative (MDN) pins for the data lane 1 of camera 1.

- SCL0 and SDA0

  A smaller serial bus consisting of SCL and SDA pins facilitates serial communication, which allows the user to control the camera functions such as selecting the resolutions. These pins connect directly to the SCCB slave interface inside the camera IC.
  The SCL pin provides a standard serial interface clock input, and SDA standard serial interface for data I/O.

Accessing this camera has been done using the inbuilt PiCamera python library. This library provides a pure python interface to the Raspberry pi camera module for almost all versions of Python.

Number of frames to capture, resolution of image, exposure, resizing of image are some of the image attributes that can be controlled from the PiCamera library.
For the time being, the camera resolution has been fixed to 480p using the command:
camera.resolution = (640, 480)

camera.capture()	Captures the image with the argument containing the path and name of the image captured.

Furthermore, delay has been predefined using the command
time.sleep()
Similarly, no. of frames to be captured in each snap can also be pre-defined, which, in this case has been set to five.
After the image has been captured and saved to the client, which is the raspberry pi itself, it is then transmitted to server, which is a laptop in this case.

3.2. Image Processing and Enhancing

3.3.1. Contrast Stretching

Contrast stretching or normalization is the process implemented for enhancing the contrast of a photograph. It is done by changing the range of pixel intensity value.
The following formula is used for contrast stretching which is also known as histogram stretching:

$$I_{new} = \frac{I_{old} - I_{min}}{I_{max} - I_{min}} * (2^k - 1) \qquad \text{---------------------------------------(3.1)}$$

Where,
I_{new}= Intensity after stretching
I_{old}= Intensity before stretching
I_{max}= maximum intensity of original image
I_{min}= minimum intensity of original image
k = bit depth of image

Here, the original picture is:

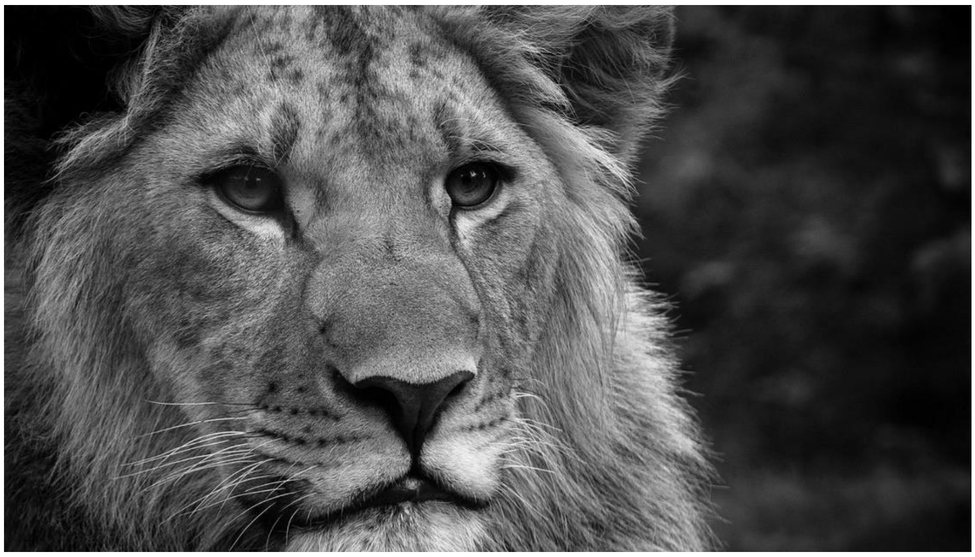


Figure 3. 9: Original Grey-Scale Image

Its histogram is:

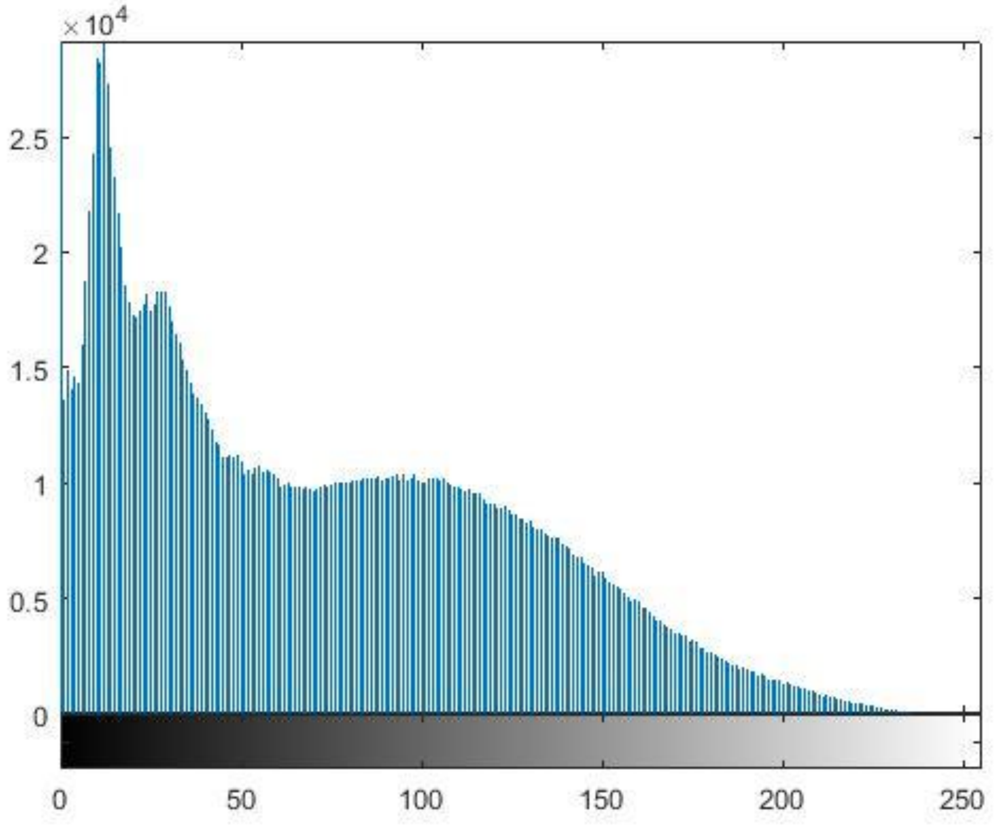


Figure 3. 10:Histogram of Original Image (No. of Pixels vs Intensity)

But after transformation:

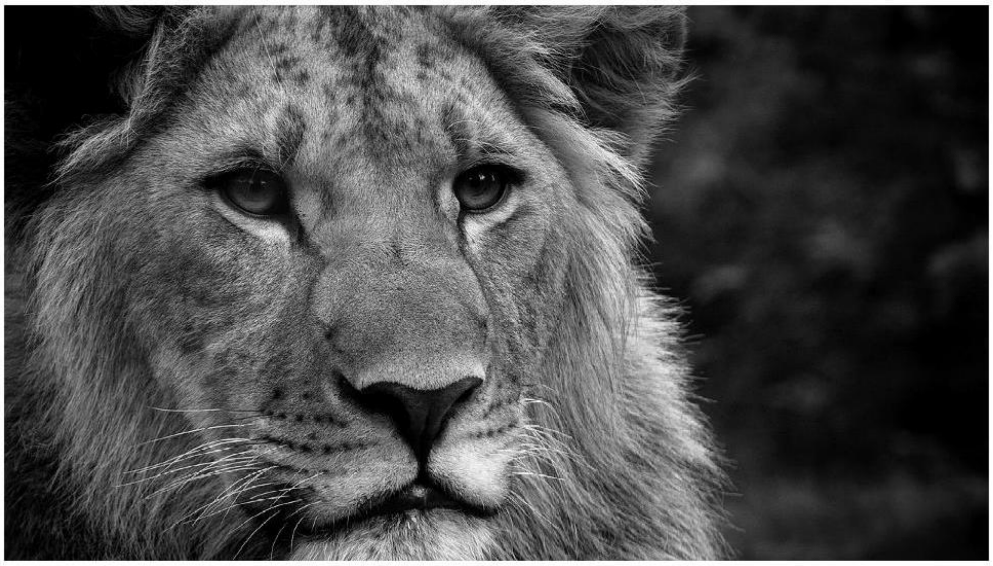


Figure 3. 11: Image After Contrast Stretching

Here is no significant difference but histogram shows that:

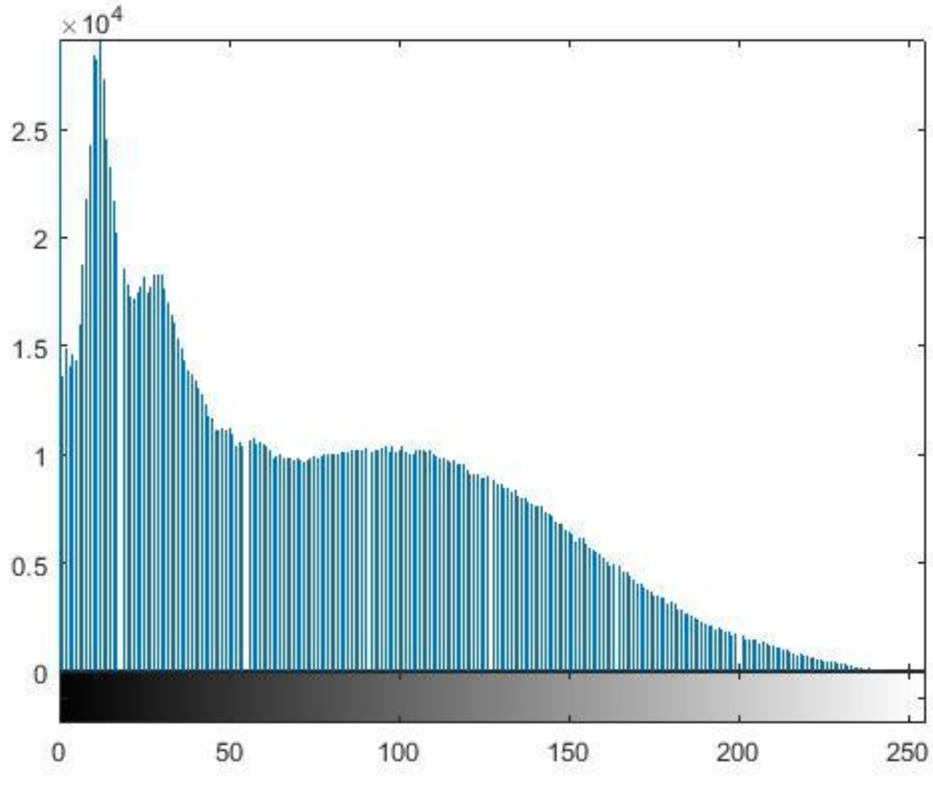


Figure 3. 12: Histogram of Contrast Stretched Image (No. of Pixels vs Intensity)

Which is stretched but the effect cannot be distinguished. It is because the maximum and minimum value of the pixels being near to 255 and 0 respectively. This causes the stretching saturation.

To overcome this limitation, we introduced histogram equalization.

3.3.2. Histogram Equalization

It is the process of adjusting the contrast of image by using the histogram and evenly distributing the probability.

For doing so the following steps are followed:

1. First the probability distribution of all the pixel's intensity is calculated.
2. The CDF table is then calculated
3. The following formula is used to normalize the histogram on the basis of probability:

$$I_{new} = \frac{cdf - cdf_{min}}{m*n-1} * (2^k - 1) \qquad \text{--(3.2)}$$

Where,

I_{new} = New pixel intensity

cdf = cumulative distribution function of the intensity value

cdf_{min} = minimum cdf

m * n = resolution of image

k = bit-depth

The original image, its histogram and cdf is given below,

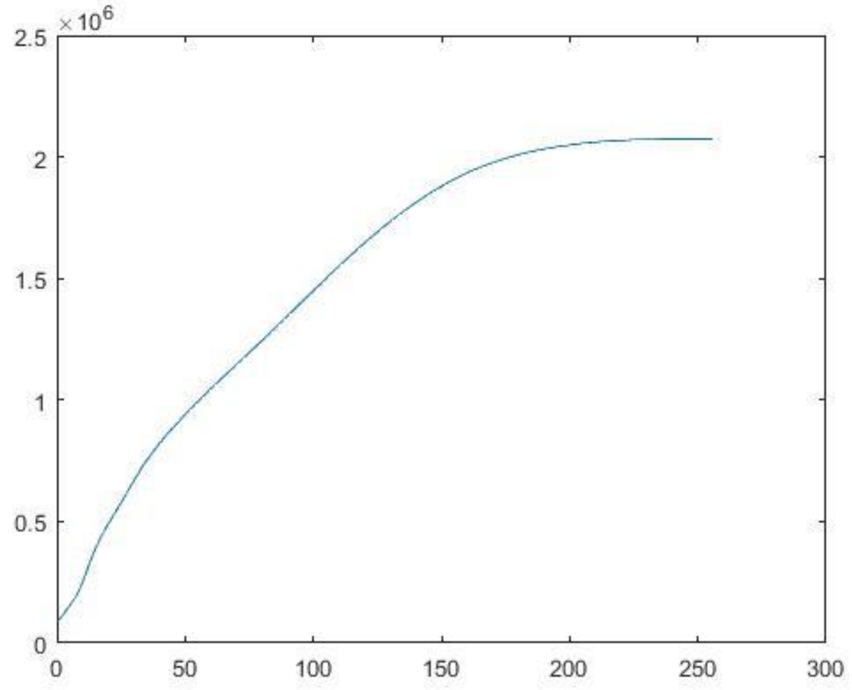


Figure 3. 13: CDF Distribution of Original Image (No. of Pixels vs Intensity)

And after equalization:

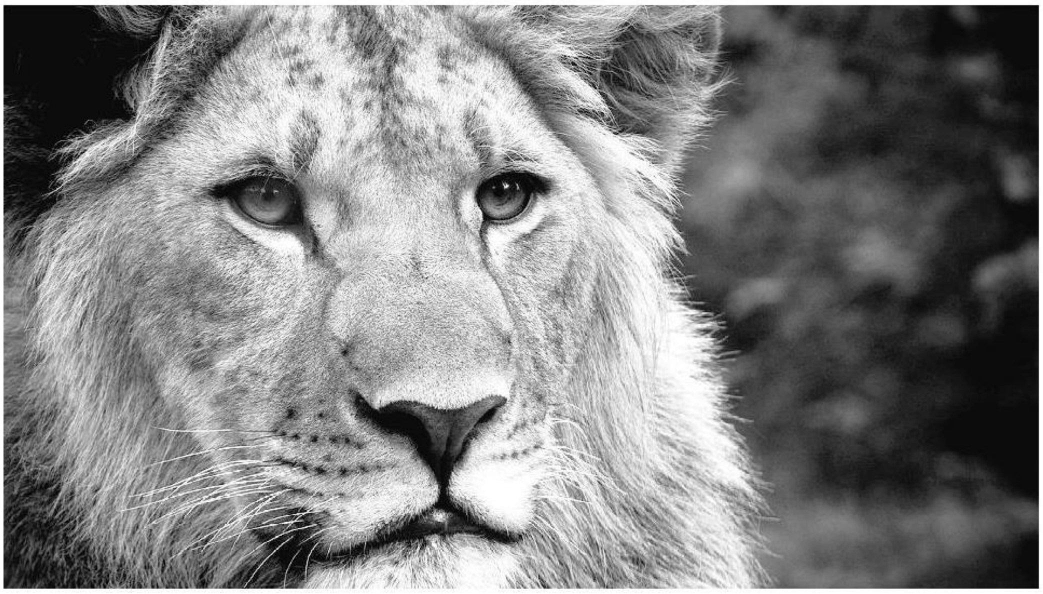


Figure 3. 14: Image After Histogram Equalization

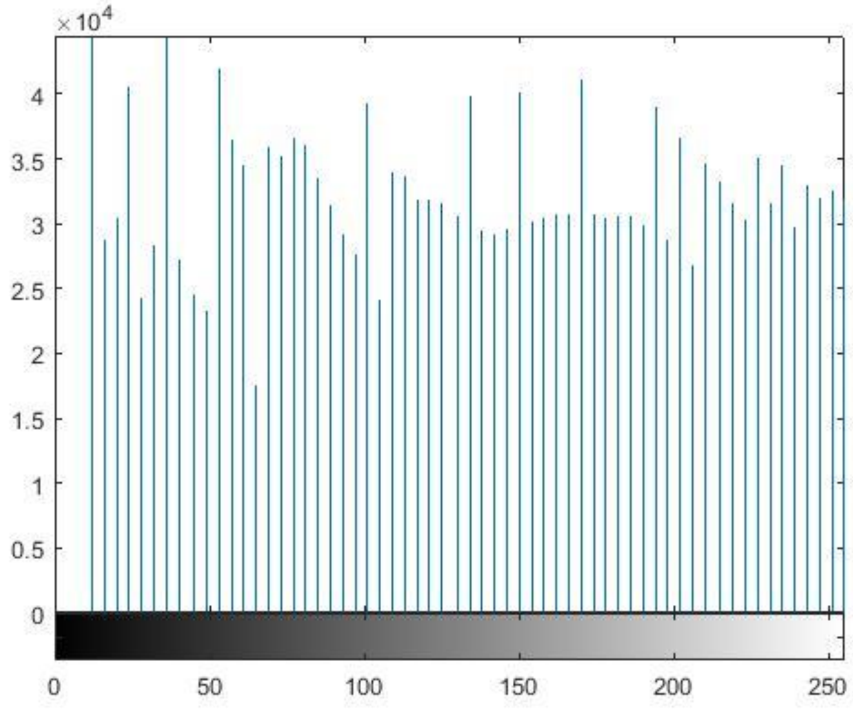


Figure 3. 15: Equalized Histogram (No. of Pixels vs Intensity)

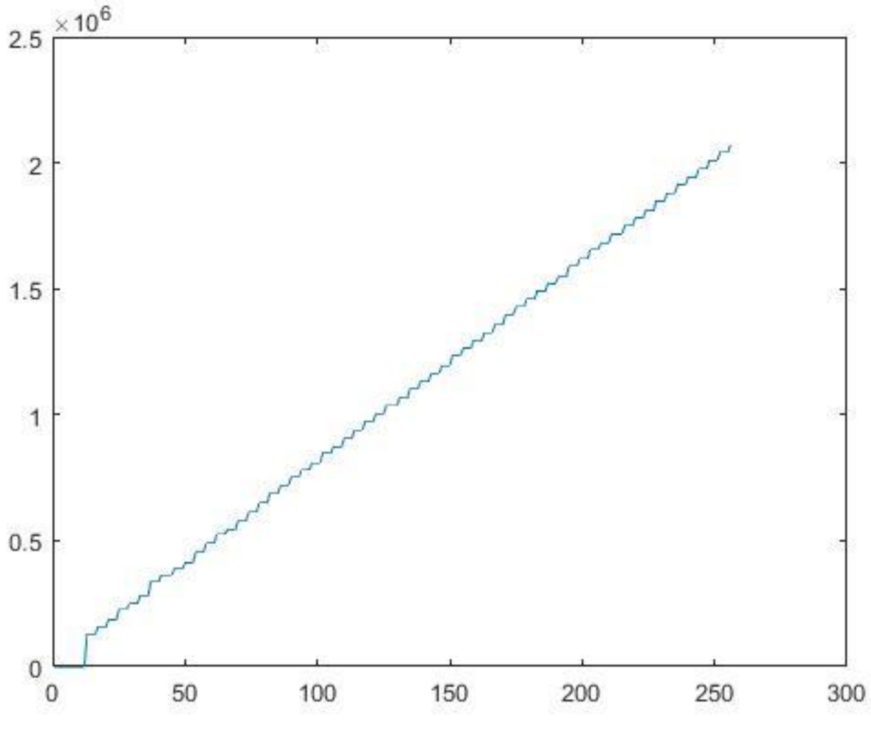


Figure 3. 16: CDF of Image After Histogram Equalization (No. of Pixels vs Intensity)

3.3.3. Canny Edge Detection Algorithm

After the contrast enhancement, the image is then subjected to edge detection. Canny edge detection algorithm is used for detection of edges

This algorithm contains 5 steps:

1. Application of Gaussian filter for noise reduction:

$$H_{ij} = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{(i-(k+1))^2+(j-(k+1))^2}{2\sigma^2}\right); 1 \le i,j \le (2k+1)$$

--------(3.3)

   Where, (2k+1) *(2k+1) is the size of filter kernel.

   Generally, a filter of size 5 * 5 is used and convoluted with the image:

$$\mathbf{B} = \frac{1}{159}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix}$$

--------------------------------(3.4)

2. Finding the intensity gradient

$$\mathbf{G} = \sqrt{{\mathbf{G}_x}^2 + {\mathbf{G}_y}^2}$$
$$\mathbf{\Theta} = \operatorname{atan2}(\mathbf{G}_y, \mathbf{G}_x),$$

--------------------------------(3.5)

   Where, G_x and G_y are the first derivative in x and y (horizontal and vertical) direction. The atan2 function is arc tangent function with 2 argument.

3. Non-maximum suppression

   The edge obtained from step ii might not be sharp rather it is blurred. So, the blurred part is subject to suppression. The blurred part is detected by comparing the edge strength of the current pixel with the edge strength of the pixel in the positive and negative gradient directions. If the value of edge strength of the current pixel with the edge strength of the pixel in the positive and negative gradient direction, the value is preserved otherwise suppressed.

4. Double thresholding

The edges are then sent for thresholding in as per the lower and upper threshold values. The edge pixel intensity is compared and is marked as strong edge, weak edge or suppressed.

5. Edge tracking by hysteresis

   The strong edges are untouched whereas the weak edges are tested for its 8 neighbor pixels. If there are any strong edges in the hysteresis, it is considered as edge else it is considered as the edge marked due to noise.

The above image after edge detection looks like:

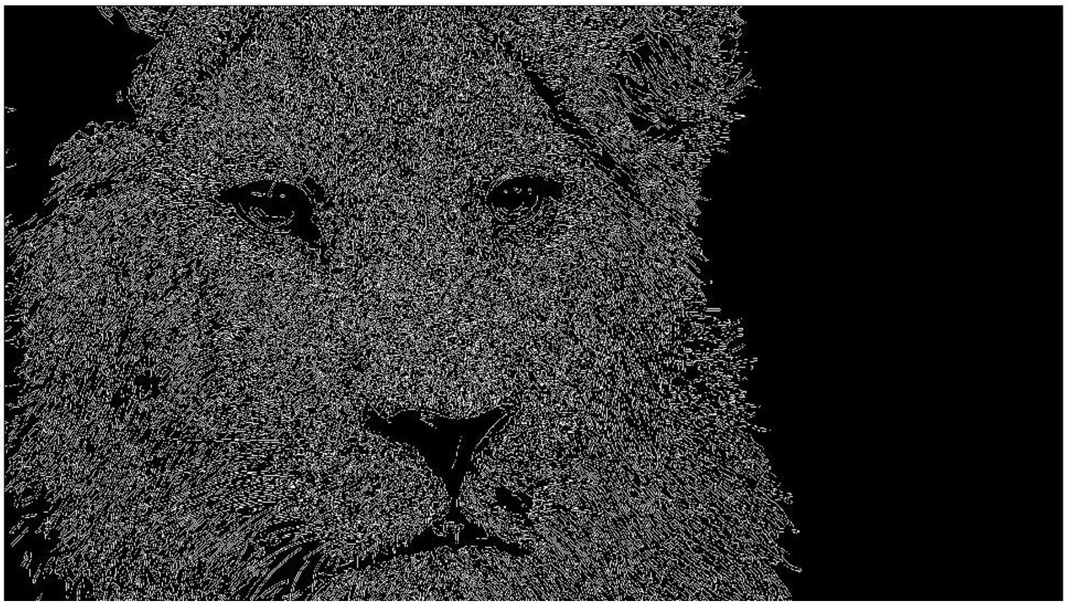


Figure 3. 17: Image After Canny Edge Detection

3.3.4. Object Isolation

In case of multiple objects in the frame, the neural network becomes unable to detect and recognize each object. Hence, opencv approach is used in order to separate objects.
This is done using opencv approach in python by ordering the object coordinates.
Initially, countors are set to identify the continuous coordinates of the borders of the objects. These coordinates are large in number, hence, simple contour methods is used to deduce the number of coordinates to four. This encloses the object inside a rectangular box. Any number of objects inside the frame can be detected via this process since the program is looped each and every time.

Canny edge detection is used for separating the borders of each object. To further improvise the canny edge detection, dilate and erode technque is applied to further enance the edges of the object.

For the instance, only the objects infornt of uniform background are separated. Errors are intordouces in case of non-uniform background. This can be further improved using image enhancement and noise reduction.

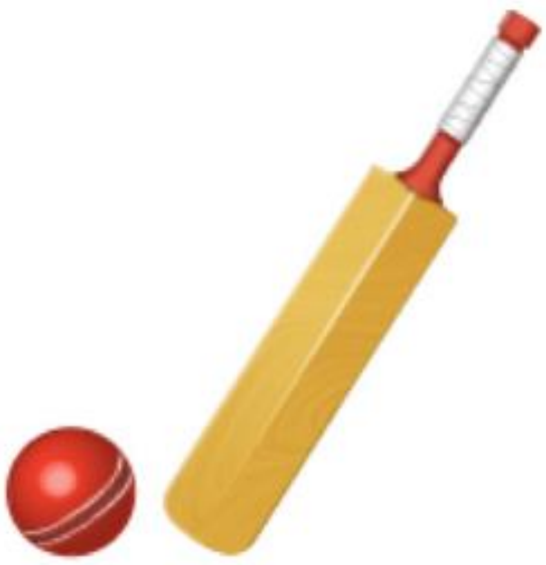
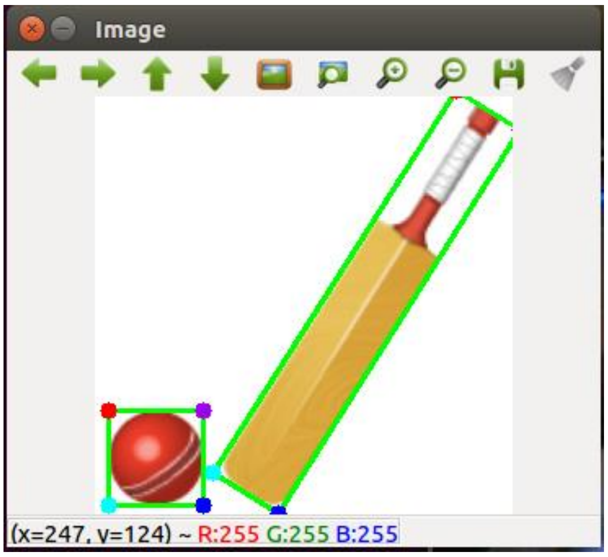


Figure 3. 18: Original Image Before Separation



(x=247, y=124) ~ R:255 G:255 B:255

Figure 3. 19: Image After Separation

3.4. Classifier Using Convolutional Neural Network

The TensorFlow is a google library for machine learning. It provides API for defining a machine learning framework and implements them separately in a session.

3.4.1. CNN for MNIST Dataset

The MNIST database (Mixed National Institute of Standards and Technology database) is a large database of handwritten digits created by remixing the samples from NIST's original datasets. The dataset contains 60,000 training images and 10,000 testing images each 28 by 28 pixels and single channel (grayscale).

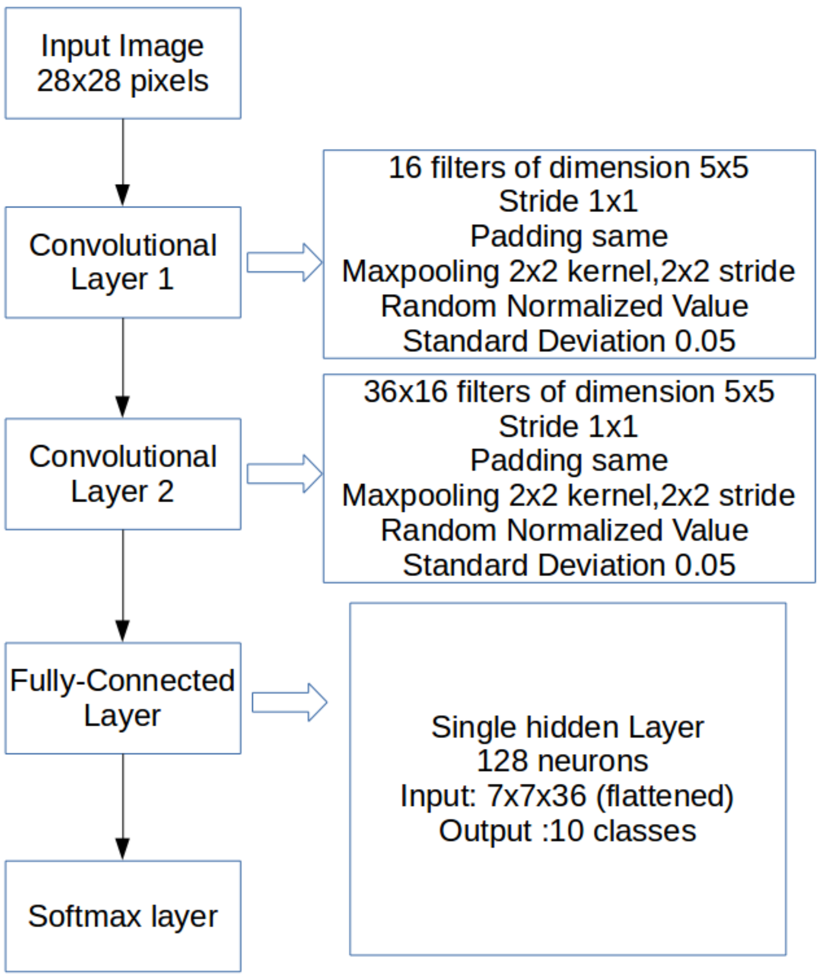


Figure 3. 20: CNN Model for MNIST Database

- Convolutional Layer

Two convolutional layers are used in this CNN. The first convolutional layer consists of 16 filters (or Kernels) of dimension 5x5 with stride of 1 in both x & y dimensions and zero padding is also used to keep the size of the output image same as the input image. These filters produce 16 new images of same dimension (due to zero padding) which are then subjected to max-pooling with a kernel of dimension 2x2 and a stride of 2x2 in the x-y plane. The kernels for both the convolution and max-pooling are matrices of aforementioned dimension initialized with a random normalized value having a standard deviation of 0.05.
The first convolutional layer resulted in a 16 channel which is fed into the second convolutional layer. This layer consists of 36 filters for each of the 16 channels with filter dimension 5x5 initialized as previous. This layer also involves max-pooling with previous specification and thus, this layer results into 36 images of 7x7 dimension.
Additionally, the outputs of each of these layers were subjected to Rectified Linear Unit(ReLU) activation to prevent saturation and dead neurons.

- Fully Connected Layer

The fully connected layer is the actual neural network classifier. The 7x7x36 features extracted from the previous convolutional layers are first expanded into two dimension (total_input_batch , 7x7x36) i.e. they are flattened layer, and then fed to the fully connected layer. The fully connected layer consisted 128 neurons in a single hidden layer and 10 classes in the output layer.
Fully connected layer is implemented as simple matrix multiplication of the flattened layer with weights and addition of bias which is then subjected to ReLU activation.

- Softmax Function

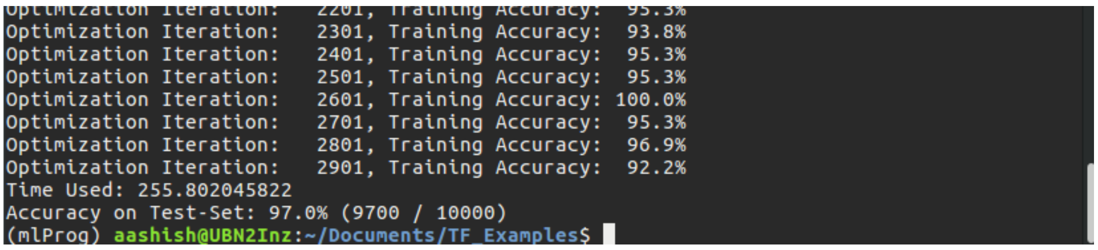
The output form the fully connected layers mayn't be interpretable. These estimates need to be normalized. The softmax function normalizes the result in such a way that score for each of 10 digit classes remain between 0 and 1 and they sum to 1.
The training dataset are passed through the convolutional neural network in bulks adding an extra-dimension. The data first undergo forward propagation through the network and then the error is back-propagated adjusting individual weights and biases. This process is iterated several times and is the actual "Training Phase" of the network.

- Optimization and training

Cost Function is a measure of how well the predicted value is close to the actual value. The cross-entropy is a continuous function that is always positive and if the predicted output of the model exactly matches the desired output then the cross-entropy equals zero. The goal of optimization is therefore to minimize the cross-entropy so it gets as close to zero as possible by changing the variables of the network layers. Once the cost function is achieved Gradient Descend algorithm is applied to minimize the cost function and thus make prediction more accurate. The key part of the Gradient Descend algorithm is Backpropagation which is a method to calculate gradients for the error with respect to each individual weights and biases.

- Outcome

The result showed test accuracy of 97.0% after 3000 iterations. On our typical machine (Laptop with 3rd gen core i5 CPU without GPU) the training period lasted about 256 seconds i.e. about 4.5 minutes. Varying results would be obtained in each try with minor variations in final outcome since the weights and biases are initialized on random. Also, major change in accuracy and error would be obtained if changes were made to the factors like Model Architecture and Number of Optimization Iterations.



```
Optimization Iteration:   2201, Training Accuracy:  95.3%
Optimization Iteration:   2301, Training Accuracy:  93.8%
Optimization Iteration:   2401, Training Accuracy:  95.3%
Optimization Iteration:   2501, Training Accuracy:  95.3%
Optimization Iteration:   2601, Training Accuracy: 100.0%
Optimization Iteration:   2701, Training Accuracy:  95.3%
Optimization Iteration:   2801, Training Accuracy:  96.9%
Optimization Iteration:   2901, Training Accuracy:  92.2%
Time Used: 255.802045822
Accuracy on Test-Set: 97.0% (9700 / 10000)
(mlProg) aashish@UBN2Inz:~/Documents/TF_Examples$
```

Figure 3. 21: Training for MNIST Database

3.4.2. CNN for CIFAR-10 Dataset

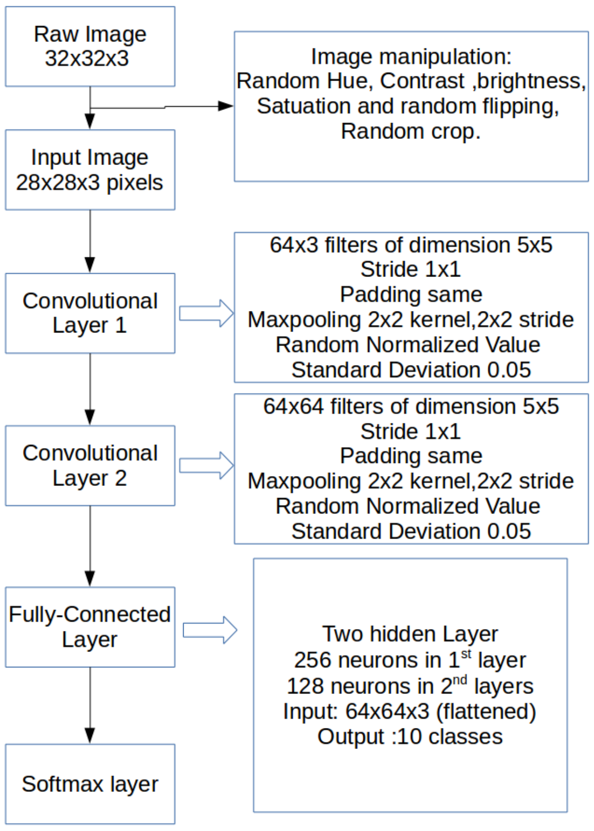

Raw Image 32x32x3
Image manipulation: Random Hue, Contrast ,brightness, Satuation and random flipping, Random crop.
Input Image 28x28x3 pixels
Convolutional Layer 1
64x3 filters of dimension 5x5
Stride 1x1
Padding same
Maxpooling 2x2 kernel,2x2 stride
Random Normalized Value
Standard Deviation 0.05
Convolutional Layer 2
64x64 filters of dimension 5x5
Stride 1x1
Padding same
Maxpooling 2x2 kernel,2x2 stride
Random Normalized Value
Standard Deviation 0.05
Fully-Connected Layer
Two hidden Layer
256 neurons in 1st layer
128 neurons in 2nd layers
Input: 64x64x3 (flattened)
Output :10 classes
Softmax layer

Figure 3. 22: CNN for CIFAR-10 Dataset

CIFAR-10 Dataset is actually a subject of a major 80 million tiny images dataset. The dataset comprises 60,000 images each of 32x32 pixels 3 channel (RGB) color images falling under one of 10 classes. It was collected by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. The 10 classes are “airplane”, “automobile”, “bird”, “cat”, “deer”, “dog”, “frog”, “horse”, “ship” and “truck”.

- Preprocessing

The 32x32 image dataset are first pre-processed i.e. they are subjected to random hue, contrast, brightness and saturation variations and randomly flipped so as to virtually increase the number of datasets. Also, the images are cropped into 24x24 pixel in random fashion. The pixels are also normalized to a value between 0 and 1 to prevent overflow. The subsequent architecture is devised to accommodate this 24x24 image and not the raw 32x32 image.

- Convolutional layers

This CNN model consists of 2 convolutional layers. The first layer consists of 64 filters of 5x5 dimension each for each of the 3 channels. Thus, total number of kernels would be 192. The layers involved functions similar to the one in the MNIST CNN model since this was built over it. The output of 1st layer is immediately subjected to a max-pooling layer which outputs 64 image channels (of the 64 filters) of dimension 12x12 (owing to max-pooling). The images are subjected to Local Response Normalization before passing into the ReLU function to bring the values to a common scale.
Also, to prevent overfitting during training, an effective regularization technique known as 'Dropout' was applied implicitly on each of the convolutional layers.
The 2nd convolutional layer has 64 5x5 kernels for each of the 64 channels resulting in a total of 4096 filters. The layer also comprises max-pooling layer and output would be 64 image channels of dimension 6x6. This 6x6x64 dimension image channel is stretched into a two dimension i.e. (? , 6x6x64) (here '?' accounts for the extra dimension added when batch processing) and then passed into fully connected layer.

- Fully Connected Layer and SoftMax Layer

The final layers are the fully connected layer. It consists of 2 hidden layers with 256 neurons in 1st and 128 neurons in 2nd. The output of the 2nd hidden layer is 10 classes. As in MNIST CNN model, the layers have implicit ReLU activation and the final output is passed into the SoftMax layer applying SoftMax function.
Using argmax function on the output of SoftMax classifier best prediction is taken.

- Optimization and training

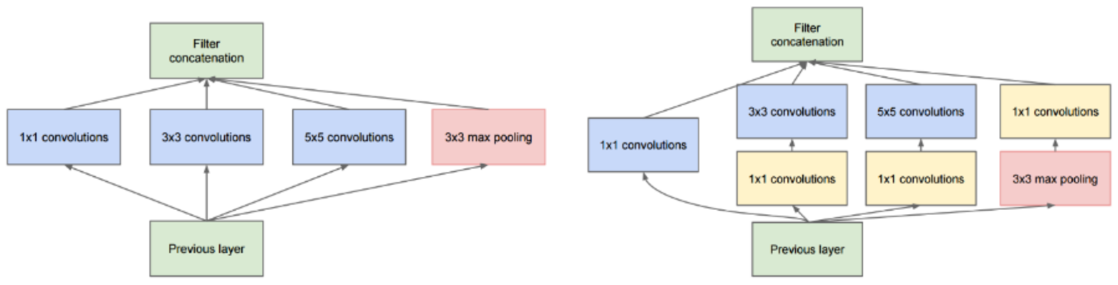
The cost function is calculated using cross-entropy function by comparing the predicted output with true output as done in MNIST model. The difference here is we used a more advanced form of Gradient Descent known as the Adam Optimizer to perform the optimizations.

3.4.3. Comparison of the MNIST CNN model vs CIFAR-10 CNN model

Out of the two model, latter being a more sophisticated architecture and additional image depth (channels), training period were lengthier and process itself was resource aggressive. While the whole 3000 training iterations over the MNIST CNN model was achieved under 5 minutes, the CIFAR-10 architecture took over 45 minutes to complete 1000 iterations. In terms of resource usage, the MNIST CNN model was pretty resource friendly, in the sense that the training could be minimized and the PC could be used for other purposes. But training the CIFAR-10 was extremely resource intensive (used all of available RAM plus addition 4 GB swap or virtual RAM) rendering the PC completely useless during the training duration.
We could see that training a practical CNN with more realistic images of higher resolution and even complex architecture was well beyond the capacity of our conventional hardware. The AlexNet (2012), devised by Krichevsky et. al. is said to have taken about a week to train on the state-of-art hardware.

3.4.4. Inception Module

A typical well-structured CNN involves multiple operations like multiple convolutions with different kernel sizes and pooling operations. These operations have proven to greatly enhance the overall performance of the CNN. Inception model in its basic form is a stack of such operations. The concept of inception model includes a 1x1 convolution, 1x1 convolution followed by 5x5 convolution, 1x1 convolution followed by 3x3 convolution and average pooling all of the operations acting on same input and stacked as a single module as shown in the figure (a). This is a naive approach and isn’t practical.



(a) Inception module, naïve version (b) Inception module with dimension reductions

Figure 3. 23: Inception Model

A practical inception module is shown in fig (b) which involves dimension reductions. The 1x1 convolutions involve dimension reduction in depth field while the spatial dimensions may remain same. Also, the 1x1 convolutions implicitly involves non-linearization.
The dimension reduced features are then convolved with higher dimension kernels reducing computational complexity.

3.4.5. GoogLeNet

GoogLeNet is was the state of art CNN architecture for the 2014 ILSVRC14 competition. The network involves use of 9 Inception modules, which were tweaked to obtain marginal improvement.

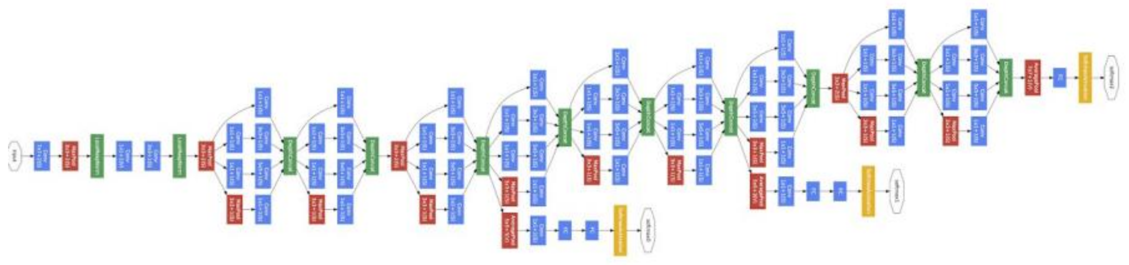


Figure 3. 24: GoogLeNet Architecture

Convolution
Pooling
Softmax
Other

Width of inception modules ranges from 256 filters (in early modules) to 1024 in top inception modules. (256, 480, 480, 512, 512, 512, 832, 832 & 1024 subsequently for each 9 inception modules). This model actually uses about 12 times fewer parameters than the 2012 state of art AlexNet.

3.4.6. Inception V3

The Inception v3 is the 2015 iteration of Google's Inception architecture for image recognition.
This CNN model has nearly 25 million parameters and uses 5 billion multiply-add operations for classifying a single image. On a modern PC without GPU this can be done in a fraction of a second per image.

3.4.7. Transfer Learning

The Inception v3 model is provided by the TensorFlow website. It is about 89mb. The model was trained on over a million images provided by ImageNet. The model can successfully predict 1000 classes of general object.
The architecture of the Inception v3 is such that it allows the final layer to be reset and retrained. In any CNN, the actual classification is done by the final (usually a fully connected) layer and the preceding convolutional layers serve as feature extractor. This final layer can then be retrained using our custom data which would be computationally less expensive due to fewer parameters to train (using the trained convolutional layers for feature extraction).

4. FUTURE PLAN

i. Training a practical CNN from scratch is beyond our hardware capacity and time limitation. More over the true-resolution dataset provided by ImageNet (only for educational and research purpose) was in the size of tera-bytes which again is well beyond our reach.

ii. The most feasible solution now would be to use a pre-trained state-of-art CNN model and use “Transfer-Learning” to classify our data. “Transfer-learning” in layman’s terms is simply taking a trained CNN model, stripping of its Fully Connected Layers and then Optimizing it with our needed limited dataset.

iii. The parameters of the Convolutional Layers are actually feature extractors. The initial layers extract simple features like edges, which are then subsequently passed into the deeper layers to extract features like shape and dimension and so on. The actual classification is done by the Fully Connected (neural network) classifier. Thus, it would make sense to use the pre-trained network and then insert a new fully connected layer optimizing it with our limited dataset.

iv. Fortunately, the TensorFlow framework provides a state-of-art CNN model, the Google Inception-V3.

v. We now plan to apply transfer-learning approach to this CNN model and then tweak it with our own images.

vi. Our next step would be to apply “Image-Enhancement” methods to create our own training set and then use it to tweak the Inception-V3 model.

5. REFERENCES

[1] Y. Tendero, S. Landeau, and J. Gilles, “Non-uniformity Correction of Infrared Images by Midway Equalization,” Image Processing On Line, vol. 2, pp. 134–146, Dec. 2012.

[2] P. Getreuer, “Automatic Color Enhancement (ACE) and its Fast Implementation,” Image Processing On Line, vol. 2, pp. 266–277, Jun. 2012.

[3] “OpenCV: Histograms - 2: Histogram Equalization,” OpenCV: Histograms - 2: Histogram Equalization. [Online]. Available: http://docs.opencv.org/3.1.0/d5/daf/tutorial_py_histogram_equalization.html. [Accessed: 10-May-2017].

[4] Krichevsky et al. “ImageNet Classification with Deep Convolutional Neural Networks”, 2012

[5] “CS231n Convolutional Neural Networks for Visual Recognition”.[Online].Available: http://cs231n.github.io/ [Accessed: 10th May, 2017]

[6] "Camera Serial Interface", En.wikipedia.org, 2017. [Online]. Available: https://en.wikipedia.org/wiki/Camera_Serial_Interface. [Accessed: 08- Mar- 2017].

[7] "Normalization (image processing)", En.wikipedia.org, 2017. [Online]. Available: https://en.wikipedia.org/wiki/Normalization_(image_processing). [Accessed: 08- Mar- 2017].

[8] "Histogram equalization", En.wikipedia.org, 2017. [Online]. Available: https://en.wikipedia.org/wiki/Histogram_equalization. [Accessed: 08- Mar- 2017].

[9] "OpenCV - Community Help Wiki", Help.ubuntu.com, 2017. [Online]. Available: https://help.ubuntu.com/community/OpenCV. [Accessed: 08- Mar- 2017].

[10] "Canny edge detector", En.wikipedia.org, 2017. [Online]. Available: https://en.wikipedia.org/wiki/Canny_edge_detector. [Accessed: 08- Mar- 2017].

[11] "Overlay Canny Edges on Original Image - OpenCV Q&A Forum", Answers.opencv.org, 2017. [Online]. Available: http://answers.opencv.org/question/74374/overlay-canny-edges-on-original-image/. [Accessed: 08- Mar- 2017].

[12] M. Peterson, "Hvass-Labs/TensorFlow-Tutorials", *GitHub*, 2017. [Online]. Available: https://github.com/Hvass-Labs/TensorFlow-Tutorials. [Accessed: 08- Mar- 2017].

[13] A. Geitgey, "Machine Learning is Fun! Part 3: Deep Learning and Convolutional Neural Networks", *Medium*, 2017. [Online]. Available: https://medium.com/@ageitgey/machine-learning-is-fun-part-3-deep-learning-and-convolutional-neural-networks-f40359318721#.5efrsb98v. [Accessed: 08- Mar- 2017].