VISVESVARAYA TECHNOLOGICAL UNIVERSITY

BELAGAVI-590018

FINAL YEAR PROJECT

ON

CROWD DENSITY ESTIMATION AND ANOMALY DETECTION

Submitted in partial fulfillment of the requirements for the VIII semester

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE & ENGINEERING

Submitted By

MANCHALA ANUDEEP	1MJ17CS094
ASLAM BASHA NAYEEM BASHA	1MJ17CS027
KISHORE N	1MJ17CS077
HABBEB ULLA KHAN	1MJ17CS717

Under the Guidance of

Dr.SUDHAN M.B,

Associate Professor,

Department of AI&ML

COLLEGE OF ENGINEERING
Since 1982
Engineering A Better Tomorrow

An Autonomous Institute
Approved by AICTE, New Delhi
Affiliated to VTU, Belagavi
Recognized by UGC under 2(f) & 12(B)
Accredited by NBA & NAAC

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

MVJ COLLEGE OF ENGINEERING

Whitefield, Near ITPB

BANGALORE-67

Academic Year 2021-22

MVJ COLLEGE OF ENGINEERING

Near ITPB, Whitefield, Bangalore-67

Department of Computer Science and Engineering

COLLEGE OF ENGINEERING
Since 1982
Engineering A Better Tomorrow

An Autonomous Institute
Approved by AICTE, New Delhi
Affiliated to VTU, Belagavi
Recognized by UGC under 2(f) & 12(B)
Accredited by NBA & NAAC

Certificate

This is to certify that the project work, entitled "**Crowd Density Estimation and Anomaly Detection**" is a bona fide work carried out by

MANCHALA ANUDEEP	**1MJ17CS094**
ASLAM BASHA NAYEEM BASHA	**1MJ17CS027**
HABEEB ULLA KHAN	**1MJ17CS717**
KISHORE N	**1MJ17CS077**

in partial fulfillment of the requirement for the VIII semester, Bachelor of Engineering in **Computer Science & Engineering** of the Visvesvaraya Technological University, Belgaum during the academic year 2021-2022.

It is certified that all the corrections/suggestions indicated for Internal Assessment have been incorporated in the Report.

Signature of the Guide
(Dr.Sudhan M B)
Associate Professor,
Dept of AI&ML

Signature of the HOD
(Prof. Tamilarasi)
Assistant Professor &HOD,
Dept of CSE

Signature of the examiners

Internal

External

MVJ COLLEGE OF ENGINEERING

Whitefield, Near ITPB, Bangalore-67

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

COLLEGE OF ENGINEERING
Since 1982
Engineering A Better Tomorrow

An Autonomous Institute
Approved by AICTE, New Delhi
Affiliated to VTU, Belagavi
Recognized by UGC under 2(f) & 12(B)
Accredited by NBA & NAAC

DECLARATION

We, **MANCHALA ANUDEEP(1MJ17CS094), ASLAM BASHA NAYEEM BASHA(1MJ17CS027) ,HABEEB ULLA KHAN(1MJ17CS717),KISHORE N(1MJ17CS077)** hereby declare that the entire project work titled **"Crowd Density Estimation and Anomaly Detection"** embodied in this report has been carried out by us during VIII semester of B.E degree at MVJCE Bangalore under the guidance of **Dr.Sudhan M B**, **(Associate Professor, Dept. of AI&ML, MVJCE)** affiliated to **VISVESVARAYA TECHNOLOGICAL UNIVERSITY, BELAGAVI**. The work embodied

in this report is original & it has as not been submitted in part or full for any other degree in any University.

MANCHALA ANUDEEP (1MJ17CS094) ________________

ASLAM BASHA NAYEEM BASHA(1MJ17CS027) ________________

HABEEB ULLA KHAN(1MJ17CS717) ________________

KISHORE N(1MJ1CS077) ________________

Date:

Place:

ACKNOWLEDGEMENT

It gives me a great pleasure to acknowledge with thanks the assistance and contribution of many individuals who have been actively involved at various stages of this project to make it a success.

Firstly, I am grateful to this esteemed institution MVJ College of Engineering for providing us an opportunity to pursue our degree course.

I thank our Principal, **Dr. P. Mahabaleswarappa** for their constant support andguidance.

I thank our Vice Principal, **Dr.M. Brindha** for their constant support and guidance.

We also like to express our sincere gratitude to our **Dr.M A Lourdu Antony Raj,** registrar and controller for examinations ,MVJCE ,Bengaluru for persistent guidance

I wish to place on record my grateful thanks to our beloved **Prof. Tamilarsi**, HOD/CSE for her constant encouragement and all the support provided during this project work.

I consider it a privilege and honor to express my sincere gratitude to my guide **Dr.Sudhan.M.B**, Associate Professor**,** Department of AI&ML for his valuable guidance throughout the tenure of this mini project work, and whose support and encouragement made this work possible.

I would also like to sincerely thank all our teaching and non-teaching staff, and my family for all their love and encourage.

ABSTRACT

Crowd density estimation and anomaly detection from crowded images have a wide range of application such as crime detection, congestion, public safety, crowd abnormalities, visual surveillance and urban planning. The purpose of crowd density analysis is to calculate the concentration of the crowd in the videos. Pattern recognition technique helps to estimate the crowd detection count and density by using face detection. The job of detecting a face in the crowd is complicated due to its variability present in human faces including color, pose, expression, position, orientation, and illumination. Detection based approach is one of the methods used for crowd detection. However, it performs poorly in highly congested scenes. since most of the targeted objects are obscured. The drawback can be overcome by using 'Density based approach'. The images captured undergo pre-processing before being fed to the Convolutional Neural Network (CNN) model. The CNN model processes the image and produces a density map using which the head count is calculated. The count is compared with the pre-determined threshold value for a given area. If the count exceeds the threshold value, the authorities are notified. The data obtained is stored in a database for future references. The data stored can be used to organize an event better in the future.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Crowd analysis and scene understanding has drawn a lot of attention recently because it has a broad range of applications in video surveillance. Besides surveillance, crowd scenes also exist in movies, TV shows, personal video collections, and in videos shared through social media.

1.1 General Overview

Crowd scenes have a large number of people accumulated with frequent and heavy occlusions. Many existing technologies for detection, tracking, and activity recognition, are only applicable to sparse scenes, they do not work well in crowded scenes. Therefore, a lot of new research works, especially targeting crowd scenes, have been done in the past years. It covers a broad range of topics, including crowd segmentation and detection, crowd tracking, crowd counting, pedestrian travelling time estimation, crowd attribute recognition, crowd behaviour analysis, and crowd abnormality detection. Many existing works on crowd analysis are scene-specific, i.e., models trained from a particular scene can only be applied to the same scene.

When switching to a different scene, data needs to be collected from the new scene to train the model again. It limits the applications of these works. Recently, people worked toward the goal of scene-independent crowd analysis. Once a generic crowd model is trained, it can be applied to different scenes without being retrained. It is nontrivial given the inherently complex crowd behaviour observed across different scenes. As there are so many crowd scenes, comparing and characterizing their dynamics is a big challenge.

1.2 Problem Definition

Over the years it has become increasingly difficult to manage a large group of people gathered at one place. There are a number of riots taking place almost on a daily basis. And in most cases, the responsible authorities will not receive the information in time and are often helpless. In order to overcome the problem, a solution has been designed using image processing to find out the crowd density of a given area and detect any anomaly in that same area. If the crowd density increases beyond a given threshold or an anomaly is detected, the concerned authorities are immediately notified thereby giving them time to take necessary precautions and preventing any mishaps from happening.

1.3 Objectives

- Surveillance of public places and keeping track of the number of people at a place.
- Making it easier to organize large events.
- Getting notified immediately whenever the crowd density goes beyond the threshold value or if there is an abnormal variation in the crowd density.
- Preventing riots, stampede, and fights from taking place in the area under surveillance.

1.4 Scope

The proposed work gives insight into mathematics and procedure involved in crowd density computation and understanding the critical aspects of applying Convolutional Neural Network. In machine learning, convolutional neural network is a class of deep, feed forward artificial neural networks that has been successfully been applied to analyzing visual imagery. CNNs use little pre-processing compared to other image classification algorithms.

Convolutional networks were inspired by biological processes. In that the connectivity pattern between the neurons resembles the organization of animal visual cortex. Individual cortical neurons respond to stimuli only in the restricted region of visual field known as receptive field. The receptive fields of different neurons partially overlap such that they cover the entire visual field.

CHAPTER 2

LITERATURE REVIEW

A literature review is a survey of scholarly related to a specific topic or research question. It is often written as part of a thesis, dissertation, or research paper, in order to situate current work in relation to existing knowledge. The purpose of a literature review is to gain an understanding of the existing research and debates relevant to a particular topic or area of study, and to present that knowledge in the form of a written report. Literature reviews are secondary sources and do not report new or original work.

2.1 "Al-Masjid An-Nabawi Crowd Adviser Crowd Level Estimation Using Head Detection" by Raghad Jaza Alamri, Maha Suliman Alkhuriji, Malak Saed Alshamani, Omniyyah Yahya Ibrahim and Fazilah Haron - 2020

Raghad Jaza Alamri, et al proposes a method which aims at developing a system which estimates the crowd levels at the doors of Al-Masjid An-Nabawi, by adopting suitable human detection algorithms. Crowd disasters are common in places of human gathering. Thousands of people gather at Al-Masjid An-Nabawi to perform their daily prayers. The mosque is packed with people and finding the right entrance is always an issue. The method proposed will be incorporated in a mobile application, which acts as a "crowd adviser" to direct the visitors of Al-Masjid An-Nabawi to less crowded doors (if any) using indoor built map. The system detects the heads in the crowd using Aggregate Channel Features (or Viola-Jones depending on the performance), count the detected objects to estimate the crowd level and present it to the user on a mobile application.

The study consists of two main phases: Phase I and Phase II. Phase I is the evaluation of the two detectors Aggregate Channel Features (ACF) and Viola-Jones (VJ) using a dataset obtained from Al-Masjid An-Nabawi at Al-Salam door. Phase II, deals the implementation of a crowd estimating system using the detector with the best performance between the two.

Phase I: Performance Evaluation

Feature learning and detector offline training: There is no simple answer to know how many samples are needed. One of the most important factors which affect the result obtained is the nature of the input. Having a large number of training samples does not always guarantee a better result. There is a very high possibility of overfitting taking place which leads to undesired results. Five thousand to six thousand samples are ideally used to train the model to detect the number of people present. AdaBoost is used for determining the good learners in both ACF and VJ, which filters the features and narrow them down to the best ones in order to distinguish the object from the background. Cascade and soft cascade are used to speed up the process of classification.

Phase II: The Proposed System Framework

Define the Region of Interest: A region of interest (ROI) is a part of the input image on which the counting operation will be performed. A ROI is defined by drawing a line right at the floor of the entrance (3 meters wide) and a parallel line making a 3m x 3m square.

Compare with max capacity: The maximum capacity proposed is 7 people per meter square and any number beyond which is considered a ‘risky’ or ‘dangerous’ situation. The crowd density is calculated and compared with the standard values and the data is stored temporarily in a database.

User Application: The user application is composed of two modules: The first is to show the number of people and the crowd density at the entrances and the second is to guide users by providing an ideal entrance with as less people as possible. The proposed system deals with managing large crowds in a mosque and guiding them to less crowded areas. The system minimizes the probability of crowd disasters taking place and also makes the lives of visitors easy.

Raghad Jaza Alamri, et al manage to show the crowdedness level of Al-Masjid An-Nabawi doors in real-time and successfully manage to disperse the crowd to less crowded entrances with the help of their application.

2.2 "Estimation of crowd density in surveillance scenes based on deep convolutional neural network" by Shiliang Pu, Tao Song, Yuan Zhang, Di Xie - 2020

Shiliang Pu, et al proposed a method to estimate crowd densities by deep ConvNets by directly extracting image features and mapping the features to crowd density levels such as very low, low, medium, high and very high. The value of these ranges is scene specific and depends on the characteristics of a particular area under observation. The system makes use of two very popular deep ConvNets in the form of GoogleNet and VGG as they are well known in the field of crowd estimation.

A typical ConvNet consists of convolutional layers, pooling layers, and fully-connected layers. Convolutional layer's job is to combine the input image in the form of a feature map with a liner filter. The output obtained will be a feature map which represents the responses of each filter. Pooling layer operate on each feature map independently and is useful to reduce the number of network parameters and computation in the network. The objective of a fully connected layer is to take the results of the convolution/pooling process and use them to classify the image into a label.

The dataset used is that of a subway carriage which contains 31 crowd scenes and over 160k annotated images. The dataset is divided into 3 subsets called train, validation and test. Input images are re-sized to 224×224, and the number of output layer channels is set to 5 corresponding to the indexes of crowd density levels. The density levels are named as follows: Very-low, Low, Medium, High, and Very-high. The train subset is used to train the models and are labelled with one of the density levels as the ground truths.

The assessment criteria for the experiment are made up of two main types:
1. Evaluation of the five-grade classification accuracy directly, including the overall 5-class accuracy together with classification accuracy of each density level.
2. Evaluation of three levels of density classification performance, namely 3-class overall accuracy. Here, "very low" class and "low" class density levels are combined into a single class called "low" and similarly "very high" class and "high" class into a single class called "high" and thereby having only 3 classes.

In conclusion, the proposed system is a new crowd density estimation method by deep ConvNets. The dataset built is a new crowd dataset of subway carriage scenes with over 160K annotated pictures to evaluate the accuracy of cross-scene crowd density estimation methods. Experiment results provide the best accuracy of 91.73% on average and confirm that the proposed method could perform well for practical applications. The next challenge would be to make the system more accurate in performance by adding a roughpeople counting function.

2.3 “Video Analytics for Indoor Crowd Estimation” by RyanTan, Indriati Atmosukarto and Wee Han Lim - 2021

Overcrowding of trains is a major issue which needs to be addressed. The main cause of overcrowding is the fact that a large number of people boarding the train tend to board the middle section of the train. Hence, the middle section of the train gets overcrowded and the two ends of the train will have ample unused space. The main objective of the proposed system is to use security camera footages with the help of deep convolutional neural networks to provide cabin-level crowd density.

The dataset used to train the model was directly obtained from the trains in Singapore. It was simple and cost effective to do so because security cameras are already installed in every cabin of every train in Singapore. Before feeding the image into a CNN, the image had to be pre-processed. It is not possible to feed in raw image data into a CNN (or any form of artificial neural networks) for prediction because raw data cannot be logically processed and computed. The images were then converted into an image matrix representation saved as a NumPy array.

During the conversion, all images were downscaled to a height of 64 pixels, retaining the original aspect ratio. However, because of the computing limitations of the training machine, the specified image size was chosen as it was big enough to contain valuable information, yet small enough to maintain a decently low training time. Additionally, the images were also converted to greyscale in an effort to prevent any colour-bias forming in the model during the training process.

The machine learning frameworks used were TensorFlow and TFLearn. An 8-layer CNN was constructed with efficiency and simplicity in mind. A pooling layer is another building block of CNN. Its function is to progressively reduce the spatial size of the representation to reduce the number of parameters and computation in the network. A MaxPool layer was introduced as the first layer in the CNN and it significantly reduced the training duration without compromising the accuracy. The first two convolution layers had filters of size 7×7, while the remaining of size 5×5. All convolution layers use ReLU as the activation function. The first and last convolution layers had 16 filters, while the second convolution layer had 32.

All MaxPool layers had a filter size of 2×2, with stride values of 2. A dropout layer with 50% dropout rated was inserted before the last layer in order to reduce overfitting. Finally, a fully-connected layer was used to aggregate the probability of a prediction. Upon launching the web application, the model was loaded to the CNN. Then, six random images were selected from the test dataset. In the testing phase, the six images are sequentially fed into the CNN for prediction. Once the prediction is complete, the six images are displayed adjacently to one another in green, yellow, and red coloured boxes, representing empty and low, medium, and high crowd levels respectively.

The main drawback of the application is that in order to disperse the people into cabins which are less crowded, more staff has to be deployed. The job of these staff is to monitor the crowd density in each cabin of the arriving train and disperse the crowd accordingly where it is less crowded. However, the deployment of additional staff is a costly affair.

2.4 "Single-Image Crowd Counting via Multi Column Convolutional Neural Network" by Yingying Zhang, Desen Zhou, Shenghua Gao, Yi Ma – 2021

In 2015, it was observed that China experienced a massive number of stampedes which resulted in lots of lives being lost. Yingying Zhang, et al proposes a multi column convolutional network (MCNN) for crowd counting in an arbitrary still image. It is made up three parallel columns of convolutional neural network (CNN), where the input for the

MCNN is image and output is the crowd density map of the image whose integration gives the overall crowd count. The advantage of using a density map is that it preserves more information. Compared to the total number of the crowd, density map gives the spatial distribution of the crowd in the given image, and such distribution information is useful in many applications. If the density in a small region is much higher than the other regions, it may indicate the presence of some abnormality.

The dataset used to train the model is called the Shanghaitech which contains nearly 1,200 images with around 330,000 accurately labelled heads. It is one of the largest crowd counting dataset in terms of number annotated heads. The dataset is made up of two parts named as Part A and Part B respectively. Images from Part A is randomly gathered from the internet while the images in Part B is gathered from the busy streets of Shanghai.

There are two natural configurations to obtain the output in a CNN. One is a network whose input is the image and the output is the estimated head count and the other one is to output a density map of the crowd (say the number of people per square meter), and then obtain the head count by integration. The second configuration is implemented as the density map preserves more information and gives the spatial distribution of the crowd in a given image.

The images usually contain a lot of heads and are of varying sizes. The reason is because some people are closer to the camera and their heads appear larger and the people far away from the camera have smaller head. Hence filters with receptive fields of the same size are unlikely to capture characteristics of crowd density at different scales. Therefore, filters with different sizes of local receptive field were used to learn the map from raw pixels to the density maps.

There is a major difference between a conventional CNN and a MCNN. Conventional CNNs usually normalize their input images to the same size. But in MCNN, the input images are preferred to be of their original sizes because resizing images to the same size will introduce additional distortion in the density map that is difficult to estimate.

In order to test the proposed model, validation is done on four different datasets. One of them is the Shanghaitech dataset mentioned earlier and the other three are the UCF CC 50 dataset, the UCSD dataset, and the WorldExpo'10 dataset. The main drawback of the model is that the Mean Accurate Error (MAE) is slightly higher than other crowd density estimation CNN models.

2.5 "Violence Detection in Surveillance Videos with Deep Network using Transfer Learning" by Aquib Mumtaz, Allah Bux Sargano, Zulfiqar Habib – 2019

One of the major concerns when a large number of people gather at a place is the possibility of occurrence of violence breaking out. The situation can escalate very quickly leading to a large number of casualties. In order to prevent such events from taking place, Aquib Mumtaz, et al propose a system to detect any violent activities taking place and taking necessary actions. Violence recognition is a key step towards developing automated security surveillance systems, to distinguish normal human activities from abnormal/ violent actions.

The proposed method involves making use of transfer learning based deep CNN model to detect violent/fight activates in videos sequences. The pre-trained model used is GoogleNet model because of its deep network architecture with 12 times fewer parameters than AlexNet. The datasets on which it operates on is the movies and hockey datasets.

GoogleNet is selected as a pre-trained source network architecture with learned features from ImageNet dataset on 15 million annotated images for 1000 categories. GoogleNet codenamed Inception is a 22 layers deep network, composed of repeated inception modules. Although the network is 22 layers deep but it has 12 times fewer parameters than AlexNet to make it efficient deep neural network architecture for computer vision.

The Hockey dataset is a unique dataset whose sole purpose is to be used only to detect fighting activities. It has 1000 video clips of 360x288 resolution and is further sub-divided into two categories, fight and non-fight, with each category containing 500 clips.

The dataset is obtained from National Hockey League (NHL) of hockey games with real-life violent events. The Movies dataset is also made particularly for fight activity detection. It has 200 video clips for both fight and non-fight activities. Fight scenes are extracted from different actions movie clips.

Network is fine-tuned on both datasets using batch size 64, constant learning rate 0.0001, with momentum set to 0.9. Based on experiments, 5 epochs training scheme was adopted to find optimal results on both datasets. Due to significant size of target datasets, network was fine-tuned by back propagating errors throughout the network to previous layers. GoogleNet was provided with resized images pipeline of 224x224 pixels for each fold during training. Experiments were conducted on NVIDIA 1080 ti GPU.

The main drawback of the system is the fact that more staff have to be deployed to monitor the situation and have to be at the site of action in order to stop the fighting from taking place and it can prove to be a costly affair.

CHAPTER 3

PRELIMINARIES

Major topics involving artificial neural network, activation function, and various learning techniques available to train the model are covered in the chapter.

3.1 Neural Network

Neural networks are made up of many layers with each layer being connected to other layers forming the network. A Feed Forward Neural Network (FFNN) can be thought of in terms of neural activation and the strength of connections between each pair of neurons.

In FFNN, the neurons are connected in a directed way having clear start and stop place, which are the input and output layers. The layer between these two layers is called the hidden layer. Learning occurs through adjustment of weights. The aim is to try and minimize errors between the output obtained from the output layer and the input that goes to the input layer. The weights are adjusted by the process of back propagation (in which the partial derivative with respect to the last layer of weights is calculated). The process of weight adjustment is repeated in a recursive manner until weight layer connected to the input layer is updated.

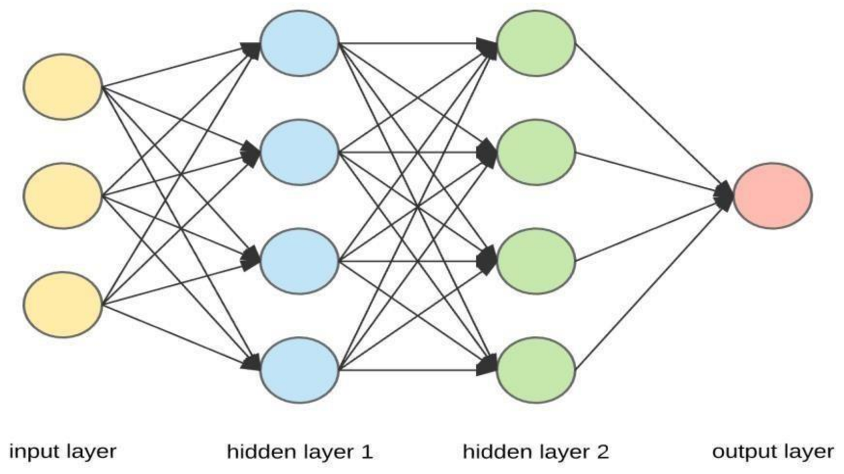


Fig 3.1: Neural Network

3.2 Convolutional Neural Network

In machine learning, convolutional neural network is a class of deep, feed forward artificial neural networks that has been successfully applied to analyze visual imagery.

Convolutional networks were inspired by biological processes in that the connectivity pattern between the neurons resembles the organization of animal visual cortex. Individual cortical neurons respond to stimuli only in the restricted region of visual field known as receptive field. The receptive fields of different neurons partially overlap such that they cover the entire visual field.

CNNs use little pre-processing compared to other image classification algorithms. It means that the network learns the filters that in traditional algorithms were hand engineered. The independence from prior knowledge and human effort in feature design is a major advantage. CNNs are widely used in image and video recognition, recommender systems and natural language processing.

3.2.1 Weights

Each communication link is associated with weights. The weight contains information about the input signal. The information is used by net to solve a problem. The weight can be represented in terms of matrix. The weight matrix is also called as the connection matrix. The weight matrix W is defined by:

$$w_1 = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix}, \; w_2 = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix}, \; w_3 = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix}$$

$$\begin{bmatrix} w_1 & w_2 & w_3 & w_4 \\ w_1 & w_2 & w_3 & w_4 \\ w_1 & w_2 & w_3 & w_4 \end{bmatrix} \text{ becomes } \begin{bmatrix} \leftarrow w_1^{\mathrm{T}} \rightarrow \\ \leftarrow w_2^{\mathrm{T}} \rightarrow \\ \leftarrow w_3^{\mathrm{T}} \rightarrow \end{bmatrix}$$

Fig 3.2: Weight Matrix Representation

3.2.2 Filter

A filter is represented by a vector of weights which convolves the input. The filter provides a measure for how close a patch of input resembles a feature. A feature may be a vertical edge or an arch. The features identified by the filter are not engineered manuallybut derived from the data through the learning algorithm.

In convolutional (filtering and encoding by transformation) neural networks (CNN) every network layer acts as a detection filter for the presence of specific features or patterns present in the original data. The first layers in a CNN detect (large) features that can be recognized and interpreted relatively easy.

Later layers detect increasingly (smaller) features that are more abstract (and are usually present in many of the larger features detected by earlier layers). The last layer of the CNN is able to make an ultra-specific classification by combining all the specific features detected by the previous layers in the input data.

3.2.3 Feature Map

The feature map on CNN is the output of one filter applied to the previous layer. A filter is drawn across the previous layer entirely and is moved one pixel at a time. Each position results in activation of the neuron and the output are collected in the feature map.

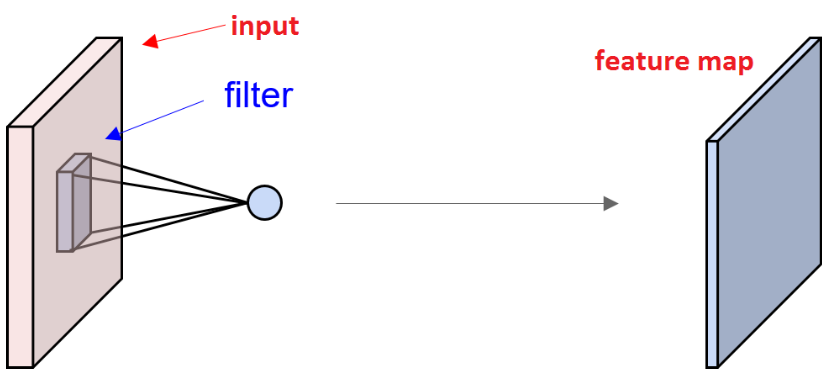


Fig 3.3: Feature Map

3.2.4 Strides

Stride controls how the filter convolves around the input volume. The amount by which the filter shifts is the stride.

3.2.5 Padding

Both the padding and stride impacts the size of data. The size of data will get reduced for the next layer in the absence of padding. Whereas adequate padding will keep the size intact. More strides imply limiting the overlap of two subsequent dot products in the convolution operation. It means that each output value in the activation will be more independent of the neighboring values.

3.2.6 Threshold

Threshold is a set of values based on which the final output of the network is calculated. The threshold value is used in the activation function. A comparison is made between the calculated net input and the threshold to obtain the network output. For each and every application there is a threshold limit. Consider a direct current (DC) motor. If its maximum speed is 1500 rpm then the threshold based on the speed is 1500rpm. If the motor is run on a speed higher than its set threshold, it may damage motor coils. Similarly, in neural networks, based on the threshold value, the activation function is defined and the output is calculated. The activation function using threshold can be can be defined as:

1, if net >= 0
-1, if net < 0

Where net is the net input which is provided to the activation function.

3.2.7 Learning Rate

The learning rate is denoted by "α". It is used to control the amount of weight adjustment at each step of training. The learning rate, ranging from 0 to 1, determines the rate of learning at each time step.

3.2.8 Working of CNN

Convolutional Neural Network is a variant of Multi-layer Perceptron (MLP) which is inspired from biology. These filters are local in input space and are thus better suited to exploit the strong spatially local correlation present in natural images. CNNs are designed to process two dimensional (2D) images. The network consists of three types of layers namely convolutional layer, pooling layer and fully connected layer. The input to the network is a 2D image. The network has input layer which takes the image as the input, the output layer which gives the trained output and the intermediate layers called the hidden layers. As stated earlier, the network has a series of convolutional and sub-sampling layers. Together the layers produce an approximation of input image data.

CNNs exploit spatial local correlation by enforcing a local connectivity pattern between neurons of adjacent layers. Neurons in layer, say ‘m’ is connected to a local subset of neurons from the previous layer of (m-1), where the neurons of the (m-1) layer have contiguous receptive fields, as shown in Figure 3.4.

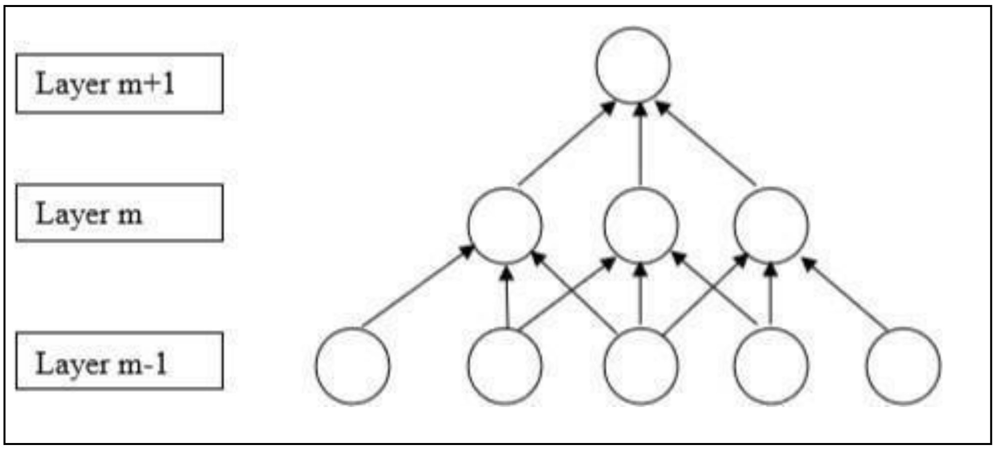


Fig 3.4: Graphical Flow of Layers-Connection

In the CNN algorithm, each sparse filter is replicated across the entire visual field. These units then form a feature map, which share weight vector and bias. Figure 3.1 represents two hidden units of same feature map. The weights of same color are shared, thus constrained to be identical. The gradient of shared weights is the sum of the gradients of the parameters being shared. Such replication in a way allows features to be detected regardless of their position in visual field. In addition, weight sharing also allows to reduce the number of free learning parameters. Due to the control, CNN tends to achieve better generalization on vision problems.

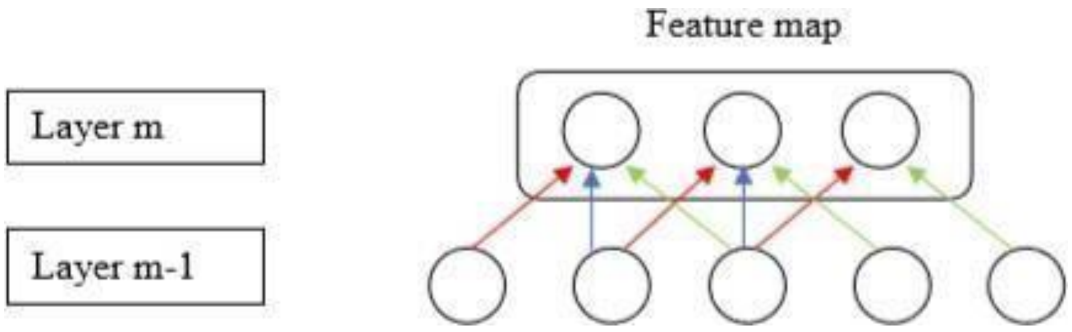


Fig 3.5: Graphical Flow of Layers-Share of Weights

CNN also make use of the concept of max-pooling, which is a form of non-linear down sampling. In the method, the input image is partitioned into non-overlapping rectangles. The output for each sub region is the maximum value.

3.2.9 Convolution Layer

The convolution layer is the first layer of the CNN network. The structure of current layer is shown in Figure 3.6. It consists of a convolution mask, bias terms and a function expression. Together, these generate output of the layer.

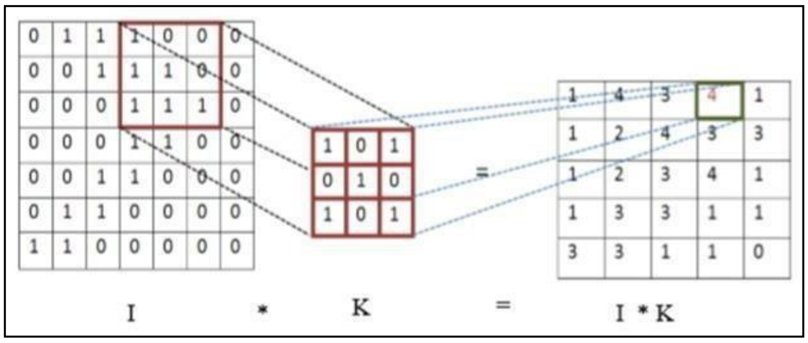


Fig 3.6: Convolution Layer

Figure 3.7 shows a 3X3 mask that perform convolution over a 32X32 input feature map. The resultant output is a 28X28 matrix. Then bias is added and sigmoid function is applied on the matrix. The convolutional layer,

- Accepts a volume of size W1×H1×D1
- Requires two hyper parameters:
  - their spatial extent F,
  - the stride S
- Produces a volume of size W2×H2×D2 where:
  - W2=(W1−F)/S+1
  - H2=(H1−F)/S+1
  - D2=D1
- Introduces zero parameters since it computes a fixed function of the input.

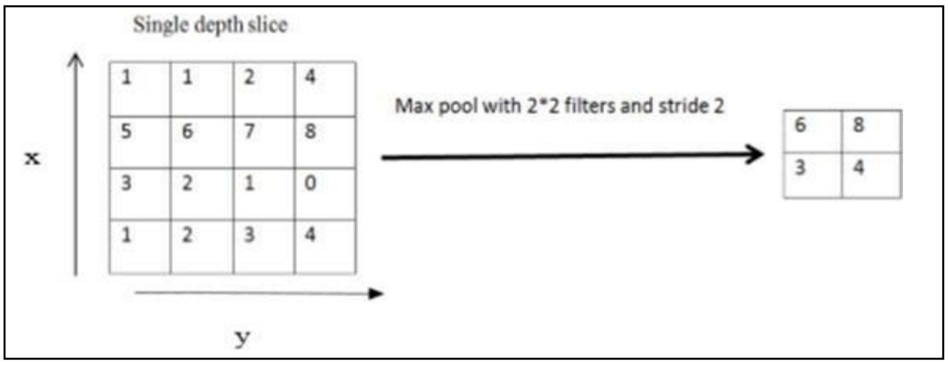


Fig 3.7: Pooling Layer

3.2.10 Fully Connected Layer

Fully connected layers connect every neuron in one layer to every other neuron in another layer. It is in principle the same as the traditional multi-layer perceptron neural network. Neurons in a fully connected layer have full connections to all activations in the previous layer, as seen in regular Neural Networks. Their activations can hence be computed with a matrix multiplication followed by a bias offset.

3.2.11 Activation Functions

Activation function is applied over the net input to calculate the output of a neural network. The information processing of a processing element can be viewed as consisting of two major parts: input and output. An integration function (say f) is associated with the input of a processing element. The function serves to combine activation, information from an external source or other processing elements into a net input to the processing element. The non-linear activation function is used to ensure that a nodes response is bounded i.e. the actual response of the node is conditioned as a result of activating stimuli and is thus controllable. Certain nonlinear functions are used to achieve the advantages of a multilayer network from a single layer network. So, nonlinear functions are widely used in multilayer networks. There are several activation functions as follows:

1. ReLU Function

The ReLU (Rectified Linear Units) function is defined as follows:

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

ReLU is linear (identity) for all positive values, and zero for all negative values. ReLU has many variants like Leaky ReLU, PReLU, and ELU. The graphical representation of ReLU is shown in Fig 3.6

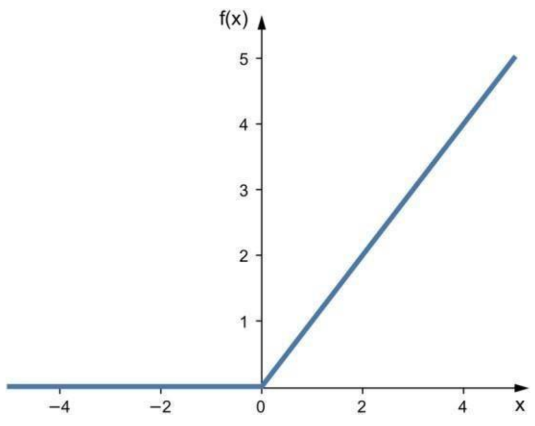


Fig 3.8: ReLU Function

2. Softmax Function

The softmax function is usually used to compute losses that can be expected when training a data set.

$$P_t\left(a\right) = \frac{e^{\frac{q_t\ (a)}{\tau}}}{\sum_{i=1}^{K} e^{\frac{q_t\ (i)}{\tau}}}$$

3.3 Backpropagation

The back-propagation learning algorithm is one of the most important developments in neural networks. The learning algorithm is applied to multilayer feed-forward networks consisting of processing elements with continuous differentiable activation functions. For a given set of training input-output pair, the algorithm provides a procedure for changing the weights in a back-propagation network to classify the given input pattern correctly. The error calculated at the output layer is propagated back to hidden unit. The aim of the neural network is to train the net to achieve a balance between the net's ability to respond and its ability to give reasonable response to the input that is not identical to the one that is used in training.

The back-propagation algorithm is different from other networks in respect to the process by which the weights are calculated during the learning period of the network. The only difficulty here is net is calculating the weights of the hidden layer in an efficient manner. When the hidden layers are increased, the network training becomes complex. The error which is the difference between the actual (calculated) and the desired (target) output, is easily measured at the output layer. The training of back-propagation networks (BPN) is done in three stages- the feed forward of the input training pattern, the calculation and back-propagation of error, and updating weights. The testing of BPN involves only the feed-forward phase. Even though the training process is slow, once the network is trained it can produce its output rapidly. Figure 3.8 shows the architecture of a BPN, showing only the direction of information flow for the feed-forward phase. During the back-propagation phase of learning, signals are sent in the reverse direction.

The terminologies used are as follows:

x = input training vector (1,……, xi,……., xn)

t = target output vector ($t1$,…….., tk,… ....., tm)

α = learning rate parameter

xi = input unit i

zj = hidden unit j

yk = output unit

kδk = error correction weight adjustment for wjk that is due to an error at output unit yk which is back propagated to the hidden unit that feed into yk

δj = error correction weight adjustment for vij that is due to the back propagation of error to the hidden unit.

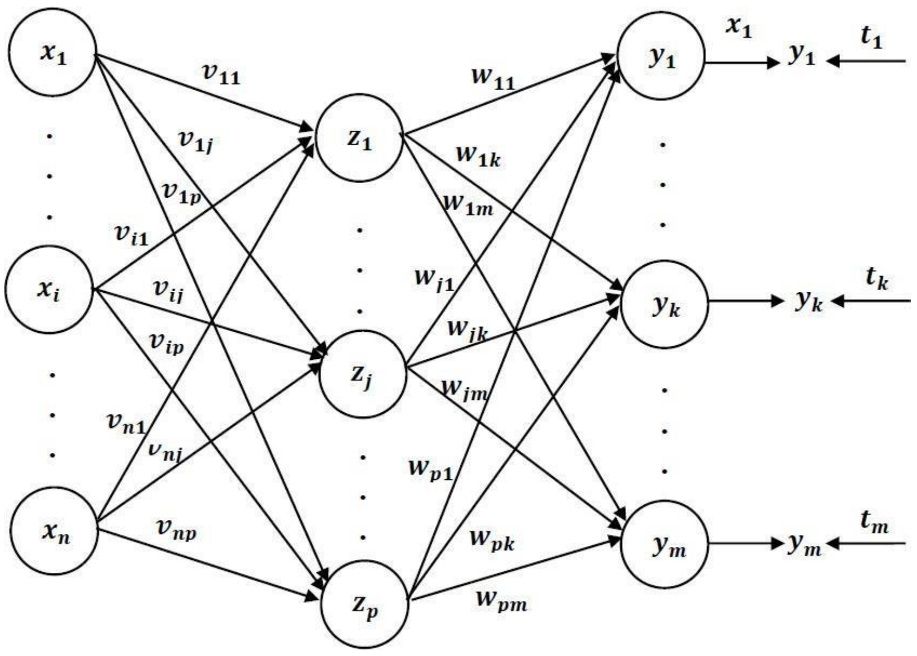


Fig 3.9: Architecture of BP

3.4 Dataset

A data set or dataset is a collection of related, discrete items of related data that may be accessed individually or in combination or managed as a whole entity.

3.4.1 Crowd Estimation

There are many datasets that are open sourced online for the purpose of developing deep learning models for crowd count estimation. From Mall dataset, caltech pedestrian dataset to world expo datasets, but all of these datasets have the annotated bodies of people and the drawbacks of these are in detecting the relative distance between any two people from a live visual. Therefore, a better option is to find out the density. The only dataset that has the above option is the shanghaitech dataset.

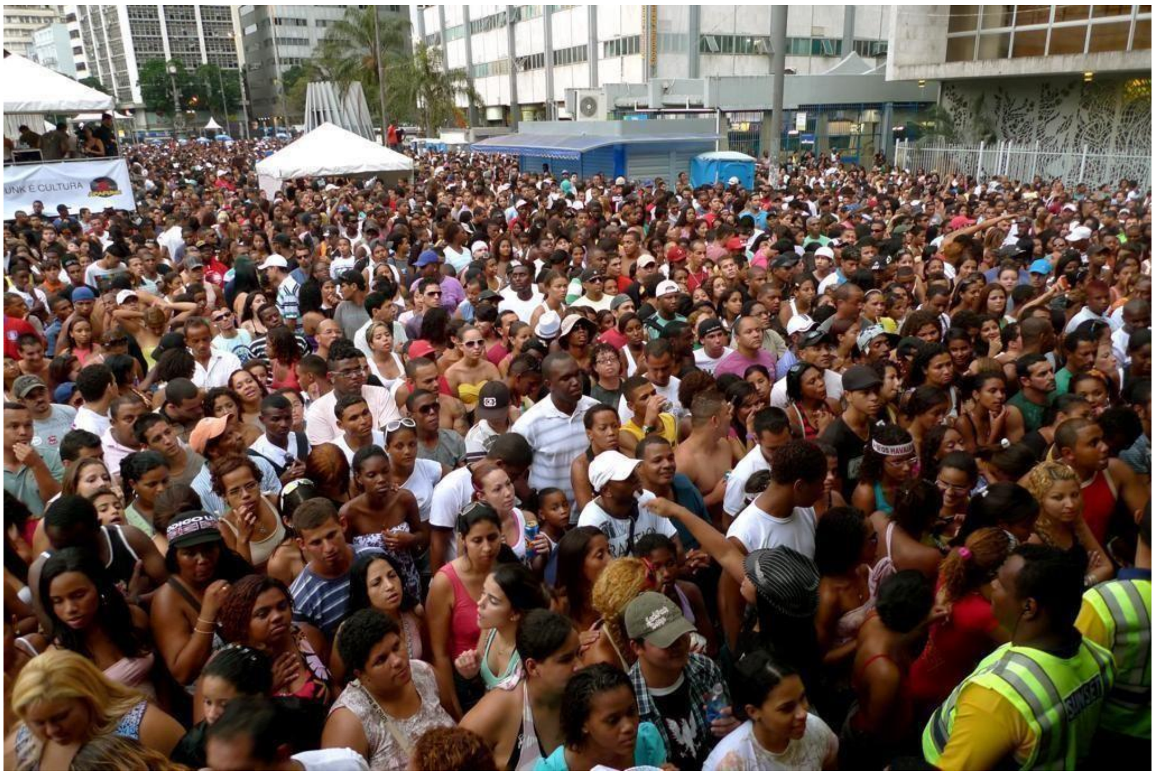


Fig 3.10: Part A of the Shanghai Tech Dataset

The Shanghai Tech dataset is made up of two parts. Part A consists of 500 images collected from the internet and made up of all ranges of resolution. One of the images is shown in Figure 3.11 and the corresponding heat map is shown in figure 3.12.

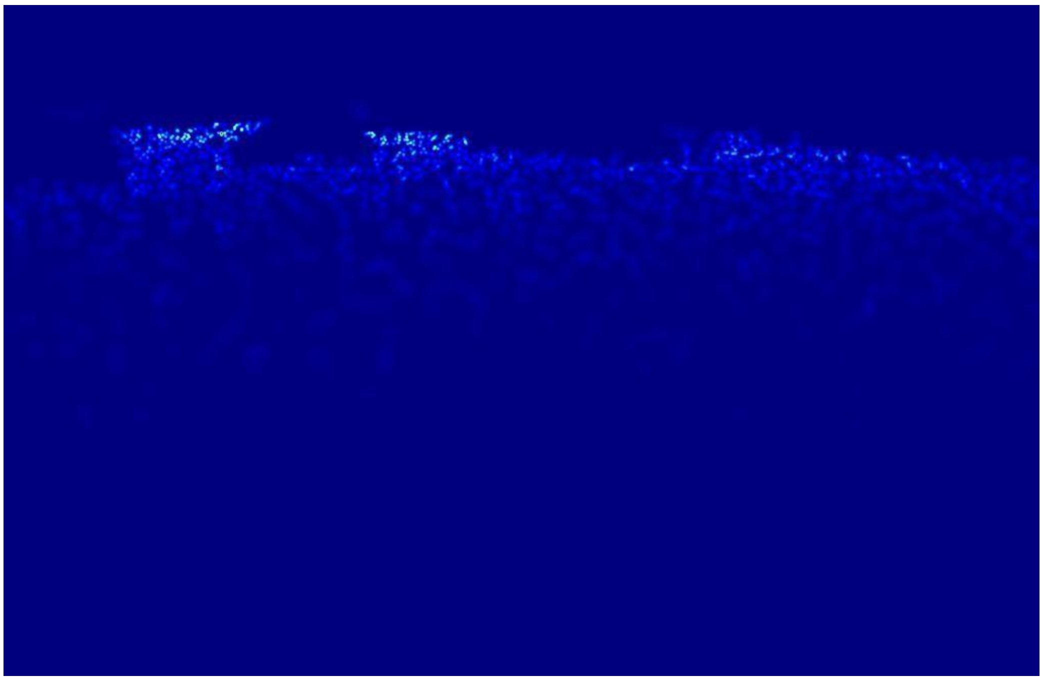


Fig 3.11: Heat Map for Part A of the Shanghai Tech Dataset

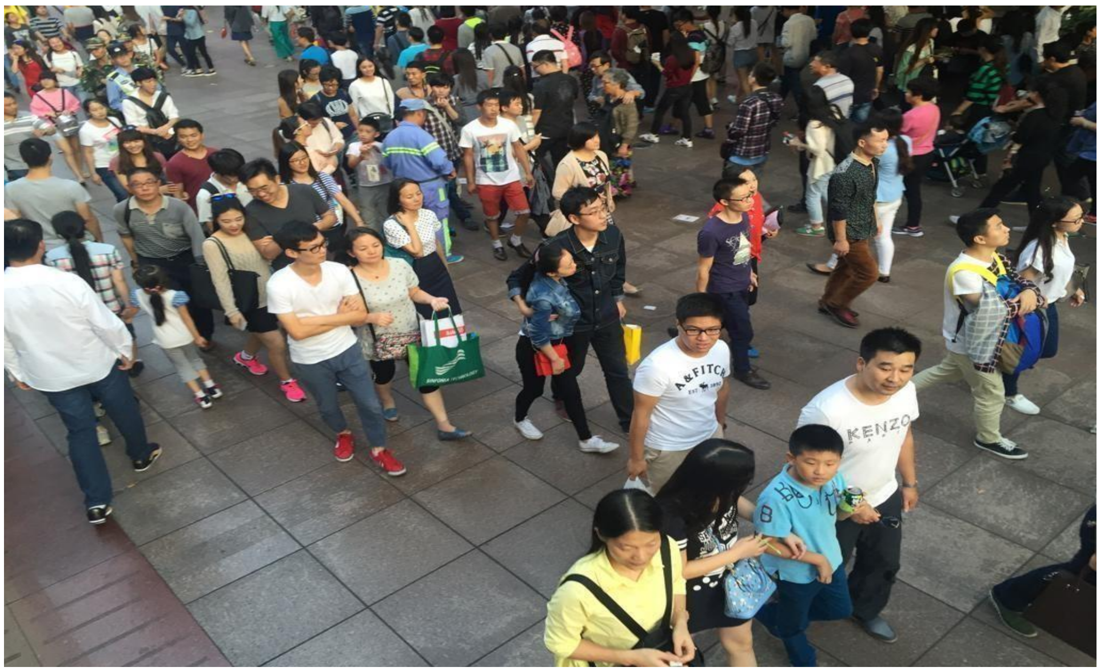


Fig 3.12: Part B of the Shanghai Tech Dataset

Part B is made up of 700 images collected directly from the streets of Shanghai and is of a standard resolution of 1280x720 pixels. One of the images is shown in Figure 3.13 and the corresponding heat map is shown in figure 3.14

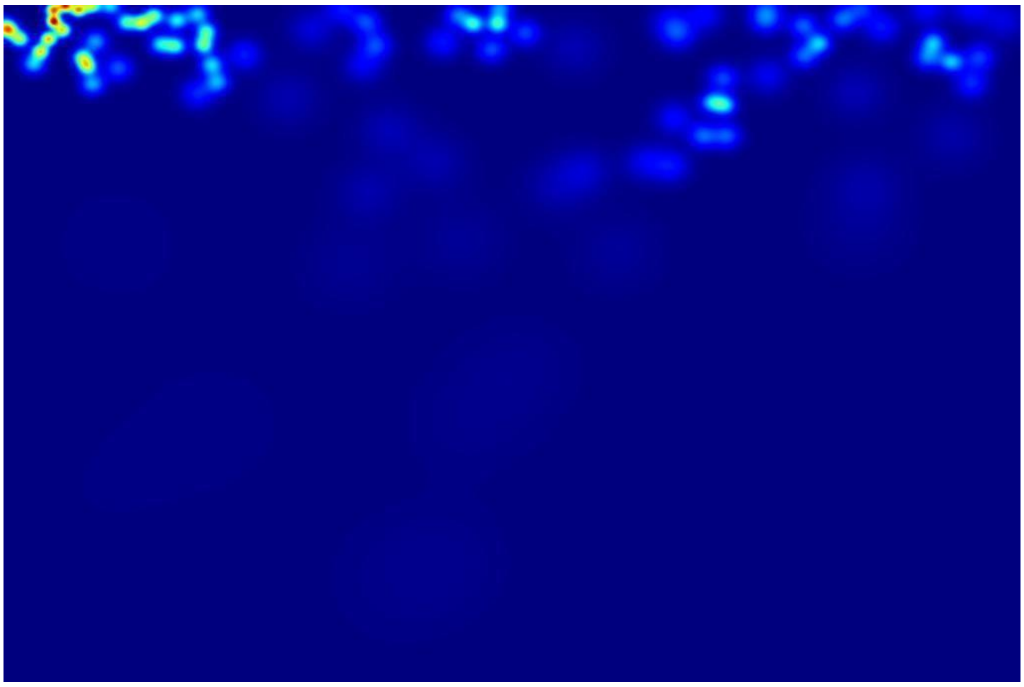


Fig 3.13: Heat Map for Part A of the Shanghai Tech Dataset

Both parts have unique images. And each part has 3 folders, images, ground truth and groundtruth-h5. Images folder has the jpeg files, the ground truth folder has matlab files contain annotated head (coordinate x, y) for that image.

And the groundtruth-h5 folder is having the density map of that image. The density map of the image is calculated by using both matlab file and the jpeg file by using gaussian filter on the places where heads are annotated, pixels where the head is present add up to one and the other pixels are all equated to zero.

3.4.2 Anomaly Detection

In the anomaly detection part, violence detection has been considered. Similar to crowd estimation datasets, anomaly detection also has a number of datasets like movie fight and hockey fight dataset. But these datasets have a very low quality of datasets. Hence real-life violence situation dataset is used it has 1000 videos of violence and non-violence of length varying from 5 seconds to 2 minutes and it is made up of mixture of movie clips and hockey fight datasets along with creator made videos for the anomaly detection the videos are cut into frames for training the model.

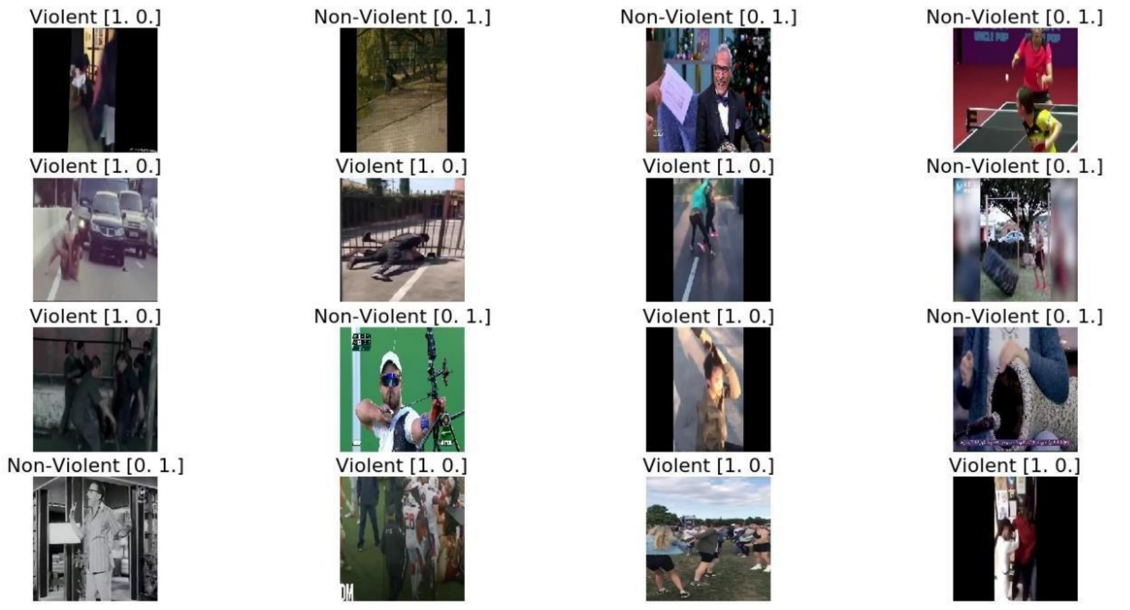


Fig 3.14: Images Representing Violent and Non-violent Data

A real-life violence situation dataset is shown in figure 3.15. It contains images with labels based on classification as to whether a given image is violent or non-violent. Also, there is a 1x2 matrix representing [1. 0.] for violent images and [0. 1.] for non-violent images.

CHAPTER 4
REQUIREMENT SPECIFICATIONS

A Software Requirements Specification (SRS) is a brief description of a software system to be developed with its hardware and software requirements. The software requirements specification document consists of all necessary requirements required for development.

4.1 Introduction

The main purpose is to build a CNN model that can count the number of people in a crowd. Back propagation algorithm is implemented in python using spyder.

4.2 General Assumptions

The following assumptions are made throughout the model development phase:

- Images of all resolutions can be used.
- The image should be visible under any kind of conditions.
- Learning rate of 1 x 10^{-6} is considered in the classification.

4.3 Hardware Requirements

- RAM: 8GB
- Hard Disk: 2GB
- Processor: Intel i5 / AMD Ryzen 5 or above
- GPU: 2GB NVIDEA GeForce chipset

4.4 Software Requirements

- Operating System: Windows 10
- Programming Language: Python version 3.7
- Framework: Anaconda with all the required libraries like PyTorch, Tensorflow, Matplotlib, OpenCV, numpy, pillow.
- Web browser: Google chrome

CHAPTER 5

SYSTEM DESIGN

Systems design is the process of defining elements of a system like modules, architecture, components and their interfaces and data for a system based on the specified requirements. It is the process of defining, developing and designing systems which satisfies the specific needs and requirements of a business or organization.

5.1 General Framework

In order to detect the crowd count in an area, the following process is followed which is showed in figure 5.1.

5.1.1 Video/Image Input

The first phase of the proposed work is to get the video or image as an input from the user.

5.1.2 Video to Frames

If the input is a video then, the given video is split into frames (two frames in equal intervals of the video). But if the input is an image, then the step is not done.

5.1.3 Pre-processing

Before the frame is fed into the CNN model the image is resized to (224x224x3) size, where 3 represents the RGB value.

5.1.4 Normalization

Before the image /frame is fed into the CNN model, the image is converted into a tensor and the tensor is normalized.

5.1.5 CNN Model

The CNN model is previously trained by several images. So, the output can be estimated. The read output tensor from the previous stage is used as an input for the model.

5.1.6 CNN Model Output

In case of crowd density estimation, the CNN model gives out the estimated density map in the form of a tensor which in turn is converted into a numpy array. In case of anomaly detection, the CNN model classifies the image based on anomaly detection. The output describes whether anomaly is detected or not.

5.1.7 Crowd Count

The numpy array can be displayed by using matplotlib. Summation of the array gives the total crowd count in the given input

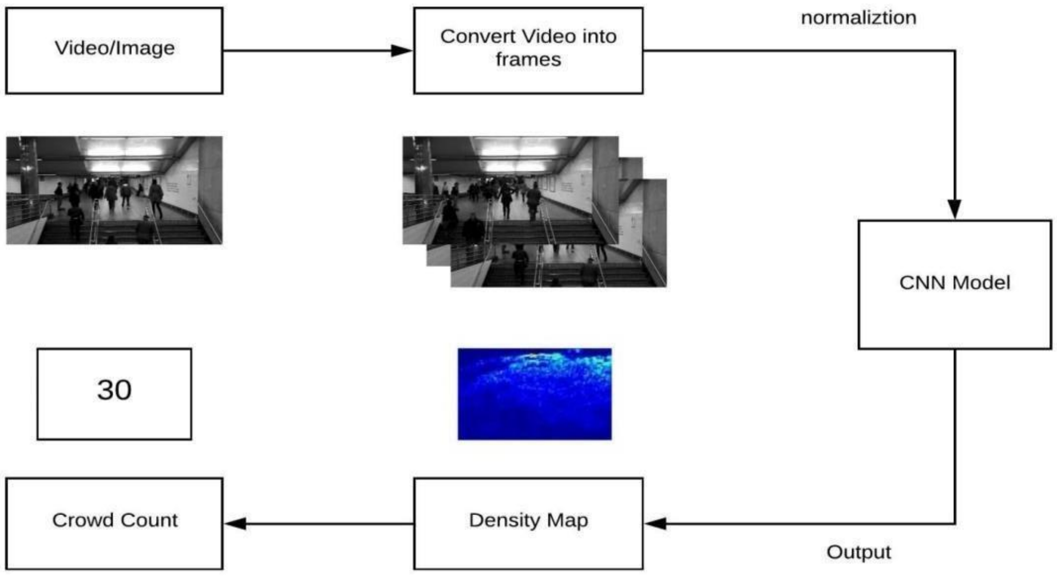


Fig 5.1: General Framework of the Model

5.1.8 Performance Evaluation

Performance evaluation is the last stage of the process. Here, the output density map from the model is checked with the original density map of the image. It is done because the accuracy of the image cannot be calculated. MAE (mean absolute error) is used to estimate the differences between the two maps. The process starts with a collection of images as training data for the model. These images are fed into the CNN model which gets better with every iteration of data. The output of the system is the density which is the expected output, then a performance evaluation is done in order to determine the proficiency of the algorithm at that given time.

5.2 System Architecture

The proposed system architecture involves two main modules, crowd estimation and anomaly detection.

5.2.1 Crowd Estimation

A convolutional neural network consists of an input and an output layer, along with multiple hidden layers. The hidden layers of a CNN typically consist of a series of convolutional layers that convolve with multiplication or other dot product. Theactivation function used is RELU layer, and is subsequently followed by additional convolutions such as pooling layers, fully connected layers and normalization layers. They are referred to as hidden layers because their inputs and outputs are masked by the activation function and final convolution. The proposed system architecture is made up of6 convolution layers and 2 max pooling layers. Convolutional layers convolve the input and pass its result to the next layer. It is similar to the response of a neuron in the visual cortex to a specific stimulus. Each convolutional neuron processes data only for its receptive field.

Pooling layers reduces the dimension of the data by combining the outputs of neuron clusters at one layer into a single neuron in the next layer. Max pooling uses the maximum value from each of a cluster of neurons at the prior layer. Each neuron in a neural network computes an output value by applying a specific function to the input value

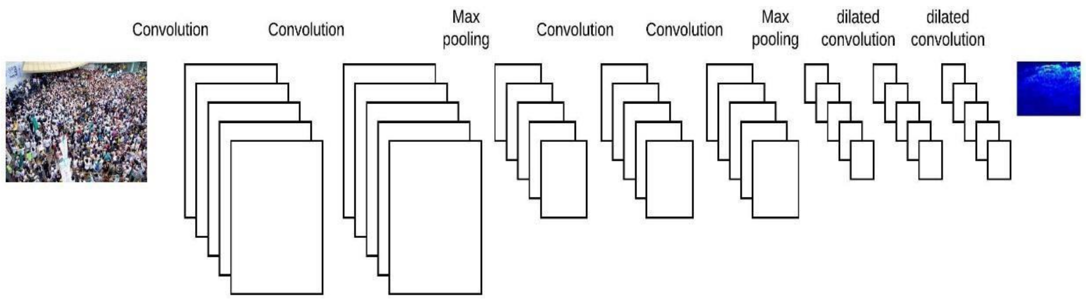


Fig 5.2: System Architecture

coming from the receptive field in the previous layer. The function that is applied to the input values is determined by a vector of weights and a bias (typically real numbers).

5.2.2 Anomaly detection

The model is built using tensorflow 2.1.0. Last 4 layers of the VGG16 pre-trained network are fine-tuned, along with the fully connected layers. Video is given as input, which is split into frames and pre-processing is done on the frames. In pre-processing, the frame/image size is reduced to 224 * 224 * 3 (RGB image) and fed into the VGG16 model. The VGG16 model classifies and detects anomaly. It then gives the output frame with text printed on the top right corner if anomaly is detected. The frames which have been given as the output are combined together to form a video of 5 frames per second and a .avi extension. Figure 5.2 shows the VGG16 architecture.

The model achieves 92.7% test accuracy on ImageNet dataset which contains 14 million images belonging to 1000 classes. The ImageNet dataset contains images of fixed size of 224*224 and have RGB channels. It contains a tensor of (224, 224, 3) as input. The model processes the input image and outputs a vector of 1000 values. The vector represents the classification probability for the corresponding class.

The input to the network is image of dimension (224, 224, 3). The first two layers have 64 channels of 3*3 filter size and same padding. Then after, a max pool layer of stride (2, 2), two layers which have convolution layers of 256 filter size and filter size (3, 3). It is followed by a max pooling layer of stride (2, 2) which is same as previous layer. Then there are 2 convolution layers of filter size (3, 3) and 256 filter.

After that there are 2 sets of 3 convolution layer and a max pool layer. Each have 512 filters of (3, 3) size with same padding. The image is then passed to the stack of two convolution layers. In these convolution and max pooling layers, the filters being used is of the size 3*3 instead of 11*11 in AlexNet and 7*7 in ZF-Net.

In some of the layers, it also uses 1*1 pixel which is used to manipulate the number of input channels. There is a padding of 1-pixel (same padding) done after each convolution layer to prevent the spatial feature of the image.

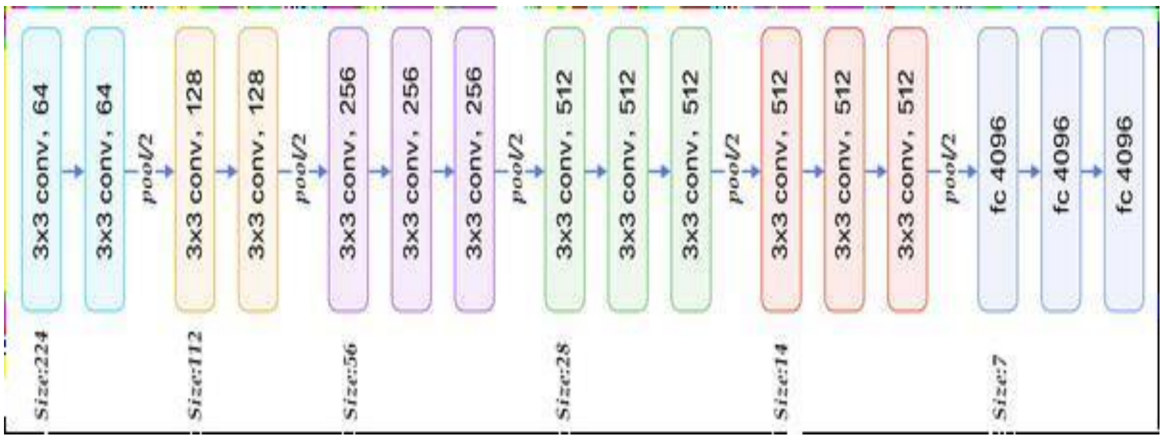


Fig 5.3: Complete VGG16 Architecture

After the stack of convolution and max-pooling layer, the output is a (7, 7, 512) feature map. The output is flattened to make it a (1, 25088) feature vector. There are 3 fully connected layer, the first layer takes input from the last feature vector and outputs a (1, 4096) vector, second layer also outputs a vector of size (1, 4096) but the third layer output a 1000 channels for 1000 classes of ILSVRC challenge, then after the output of 3rd fully connected layer is passed to softmax layer in order to normalize the classification vector. After the output of classification vector top-5 categories for evaluation. All the hidden layers use ReLU as its activation function. ReLU is more computationally efficient because it results in faster learning and it also decreases the likelihood of vanishing gradient problem.

CHAPTER 6

IMPLEMENTATION

The implementation phase represents the work done to meet the requirements of the scope of work and fulfil the charter. During the implementation phase, the work defined in the plan is accomplished and adjustments are made when some factors changed during development.

6.1 Crowd Estimation

Implementation of crowd estimation are covered in the following topics. It contains all the details starting from how an image or a video processed in order to give the head count as output with necessary pre-processing.

6.1.1 Image/Video Input

If the input is an image no conversion is needed. It is directly given as input to the model. If the input is a video it is converted into frames with the help of OpenCV. OpenCV is one of easiest tool which is used to manipulate videos and images. Python has OpenCV in-built library, that provides many built-in functions for image processing. One of the important in-built function used was VideoCapture().

The VideoCapture() function is responsible for converting the video into image sequences. Every second of the video is made up of 30 image frames. It may vary with the type and quality of camera used to capture the video. Only three frames are extracted from equal interval of the input video.

6.1.2 Pre-Processing

Data Preprocessing is a technique that is used to convert the raw data into a clean data set. Pre-processing refers to the transformations applied to the data before feeding it to the model. Normalization is the only preprocessing done on the image. The input image or frame is first taken as an RGB image and converted into a tensor for easier processing and then the tensor is normalized. The mean and standard deviation values are [0.485, 0.456, 0.406], [0.229, 0.224, 0.225] respectively.

The three values correspond to each RGB value and the dataset is shuffled. After which all the images are cropped into 9 patches, and the size of each patch is ¼th of the original size of the image. The first 4 patches are divided into 4 quarters and the other 5 patches are randomly cropped. Finally, the mirror of each patch is taken to double the training set.

6.1.3 Convolutional Neural Network

The image which is in the form of a tensor is fed into the CNN model. It is a pure convolution neural network, meaning no fully connected layer is used in the model. It would be in the form of a tensor. The CNN model is made up majorly of two parts. They are the frontend and the backend. The frontend consists of 4 convolution layers the all the layers have a kernel size of 3x3 but the number of kernels in each layer vary.

The first two layers have 64 kernels followed by a maxpooling layer of 2x2 filter size. The other two convolution layers have 128 kernels again followed by a maxpooling layer of the same filter size. The backend is made up of two dilated convolution layer of kernel size 3x3 and number of kernels are 128 and 64 respectively. The dilated convolution layers are used because it preserves the dimensions of the image.

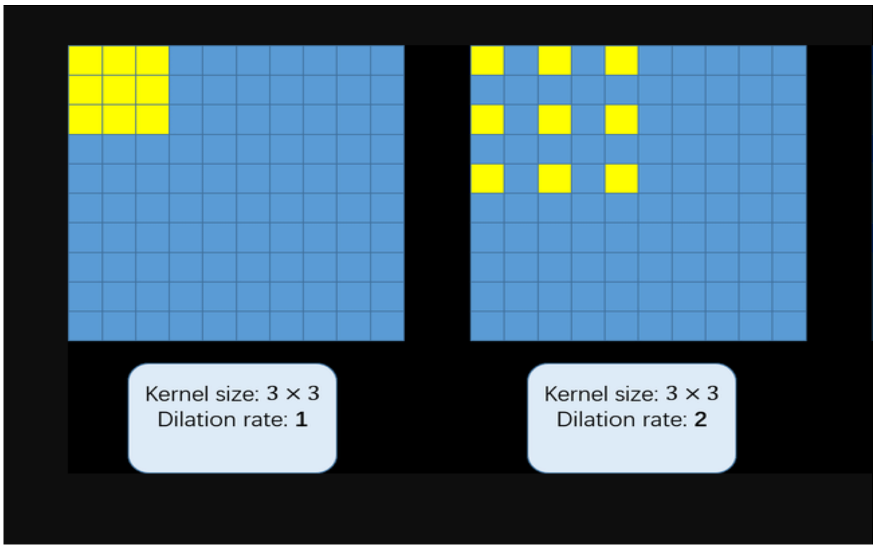


Fig 6.1: Dilation

By using two maxpooling layers the output image is 1/4th the size of the input image. The activation function used is ReLU function, which is the simplest non-linear activation function. The Rectified Linear Unit (ReLU) is the most commonly used activation function in deep learning models.

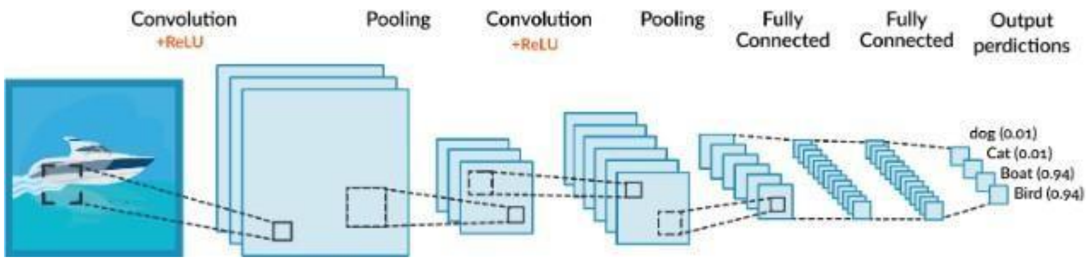


Fig 6.2: Convolutional Neural Network

6.1.4 Stochastic Gradient Descent with Momentum

Stochastic Gradient Descent (SGD) with momentum is method which helps accelerate gradients vectors in the right directions, thus leading to faster converging.

It is one of the most popular optimization algorithms and many state-of-the-art models are trained using it. In contrast it performs a parameter update for each training example.

A certain value is of momentum is defined which is a moving average of the gradients. It is then used to update the weight of the network. Which can be represented as follows.

$$V_t = \beta V_{t-1} + \alpha \nabla_w L(W, X, y)$$
$$W = W - V_t$$

Where *L* is loss function, *delta* is the gradient with respect to weight and *alpha* is the learning rate. Loss function is a function that gives an idea on how good the neural network is performing for a certain task. An intuitive way is to take each training example and pass through the network to get the number, subtract it from the actual number and square it. Squaring is done to avoid getting negative values. The loss function is written as follows.

$$L(y,\hat{y})=\frac{1}{m}\sum_{i=1}^{m}(y_i-\hat{y}_i)^2$$

Where *y* stands for the number actual value expected as output from the network, *y*ˆ is the number received after passing an example through the network, *i* is the index of a training example.

SGD has trouble navigating ravines. That is, areas where the surface curves much more steeply in one dimension than in another. It is common around the local optima. In these scenarios, SGD oscillates across the slopes of the ravine while only making hesitant progress along the bottom towards the local optimum.

The momentum term increases for dimensions whose gradients point in the same directions and reduces updates for dimensions whose gradients change directions. As a result, faster convergence is gained with reduced oscillation.

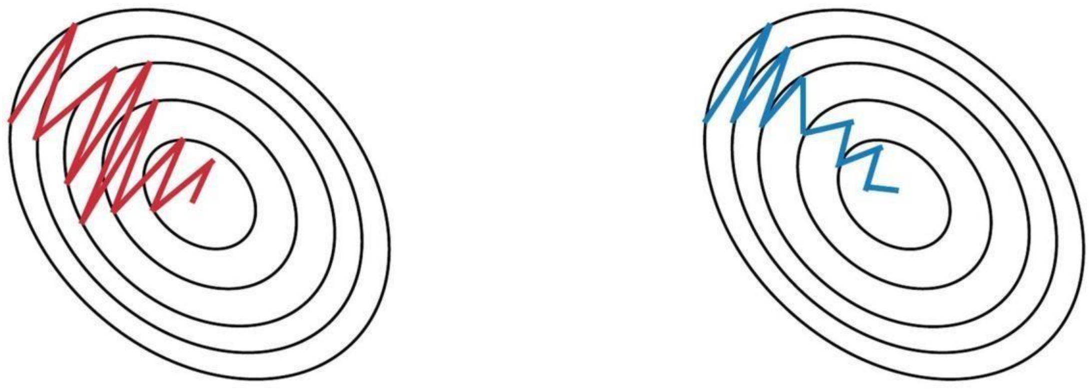


Fig 6.3: Difference between SGD without Momentum(left) and SGD with Momentum(right)

6.1.5 Training the CNN model

CNN model is trained using the back propagation supervised learning algorithm. The model is trained a number of times with the same training set. Each time the model encounters the entire training dataset in the training process is called as an Epoch. One epoch is equal to one forward pass and one backward pass of all the training examples.

The model is trained using Stochastic Gradient Descent (SGD) with momentum. A single data point is taken from the dataset for which the weights of the model are updated because SGD is used and the batch size is 1. Mean square loss is taken as the loss function with the learning rate fixed at 10^{-5} and momentum at 0.95 which is a standard value.

The two main algorithms which are being used are data processing algorithm and the training and testing algorithm are Data Processing Algorithm and the Training and Testing Algorithm.

6.1.5.1 Data Processing Algorithm

Step 1: Load the input image and ground truth.
Step 2: Perform Image normalization.
Step 3: Create train, test batch, test label for training and testing dataset.

6.1.5.2 Training and Testing Algorithm

Step 1: Build CNN model.
Step 2: Provide the training set as the input to the CNN model and train the model.
Step 3: Test the model using test set.
Step 4: Define loss function.

6.1.6 Building the CNN Model

For building the CNN model Pytorch library is necessary. Pytorch is an open source software library for high performance numerical computation. Its flexible architecture allows easy deployment of computation across a variety of platforms (CPUs, GPUs), and from desktops to clusters of servers to mobile and edge devices.

Originally developed by Facebook's AI Research lab (FAIR). it comes with strong support for machine learning and deep learning and the flexible numerical computation core is used across many other scientific domains.

A number of pieces of Deep Learning software are built on top of PyTorch, including Tesla, Uber's Pyro, HuggingFace's Transformers, and Catalyst.

6.2 Anomaly Detection

With the rapid growth of surveillance cameras to monitor the human activity demands such system which recognize the anomalies and suspicious events automatically. Abnormal and anomaly detection has become an active research area of computer vision and image processing.

6.2.1 Input Video

The input is a video which is converted into frames with the help of OpenCV. Only mp4 format is taken as the input and based on the number of frames per seconds. Only one frame is extracted from each second of the input video.

6.2.2 Pre-processing

Image resize is the only preprocessing done on the image. The input image or frame is first taken as an RGB image and converted into an image of size 224x224 for easier processing.

6.2.3 Convolutional Neural Network Model

The image which is now resized is fed into the CNN model. The CNN model is a customized VGG16 model. The first 12 layers of the model are taken as it is. But the last 4 layers are modified. The last maxpooling layer is changed to an average pooling layer. The next two layers are dense layers with 64 nodes or neurons with an activation function as ReLu. But the last layer or the output layer has only two neurons with the softmax activation function. The first 12 layers in the model are pre-trained.

6.2.4 Adam Optimization Algorithm

Stochastic gradient-based optimization is of core practical importance in many fields of science and engineering. Many problems in these fields can be cast as the optimization of some scalar parameterized objective function requiring maximization or minimization with respect to its parameters. If the function is differentiable with respect to its parameters, gradient descent is a relatively efficient optimization method, since the computation of first-order partial derivatives with respect to all the parameters is of the same computational complexity as just evaluating the function.

The proposed work uses Adam optimization algorithm which is an extension to stochastic gradient descent. Stochastic gradient descent maintains a single learning rate (termed alpha) for all weight updates and the learning rate does not change during training. The Adam optimization algorithm leverages the power of adaptive learning rates methods to find individual learning rates for each parameter.

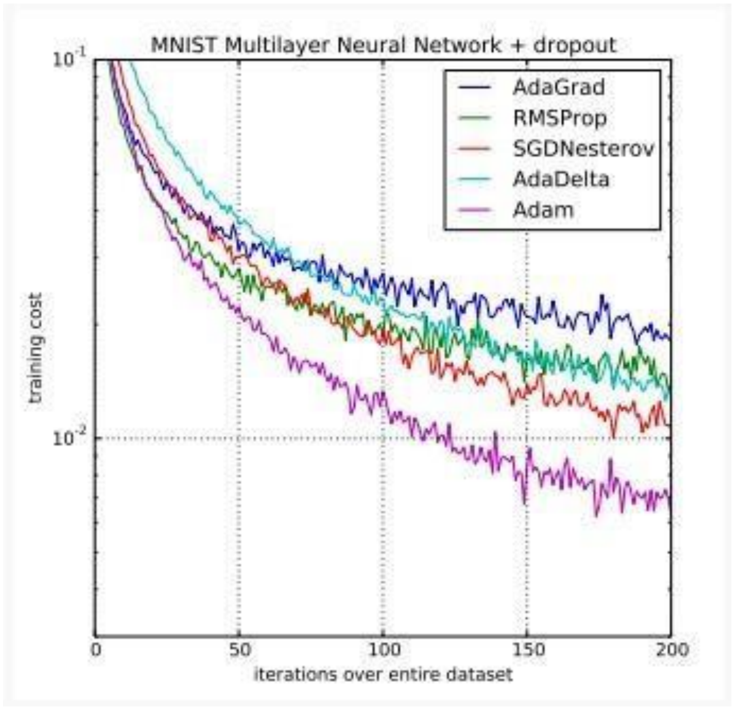


Fig 6.4: Comparison of Adam to Other Optimization Algorithms

The algorithm also combines RMSprop which maintains per-parameter learning rates that are adapted based on the average of recent magnitudes of the gradients for the weight. It can handle sparse gradients on noisy problems and uses estimations of first and second moments of gradient to adapt the learning rate for each weight of the neural network.

6.2.5 Training a CNN model

CNN model is trained using the back-propagation supervised learning algorithm. The model is trained using ADAM optimization and the batch size is taken as 16. Categorical cross entropy is taken as the loss function with the learning rate fixed at 10^{-5}. The algorithms used are Data Pre-processing algorithm and Training and Testing algorithm.

6.2.5.1 Data Pre-processing Algorithm

Step 1: Load the input image and ground truth.
Step 2: Perform Image resize to (224x224x3)
Step 3: Create train, test batch, test label for training and testing dataset.

6.2.5.2 Training and Testing Algorithm

Step 1: Build the CNN model.
Step 2: Provide the training set as the input to the CNN model and train the model.
Step 3: test the model using test set.
Step 4: Define loss function.

6.2.6 Building the CNN Model

For building the CNN model TensorFlow library is necessary. TensorFlow is an open source software library for high performance numerical computation. Its flexible architecture allows easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices. Originally developed by researchers and engineers from the Google Brain team within Google’s AI organization, it comes with strong support for machine learning and deep learning and the flexible numerical computation core is used across many other scientific domains.

CHAPTER 7

TESTING

The document describes the overall test phases of the image classification process. It describes the methodologies and techniques used in the various test phases. Black box testing is carried out to check the software's confirmation to the specification. White Box testing is done to check the software's confirmation to the design.

7.1 Unit Testing

Unit testing is used to verify a single program or small section of program. It can be considered white-box testing since the details of the implementation is known, and the uses that knowledge to come with test cases. Unit testing, checks for the successful execution of all the internal execution paths. Unit testing was conducted during the coding phase for both the package to test internal logic of each module. All modules are tested against specifications produced during design of each module.

Table 7.1: Unit Testing

TEST CASE ID	**TEST CASE NAME**	**TEST CASE DESCRIPTION**	**TEST STEPS**				**TEST STATUS**
			STEPS	**INPUT GIVEN**	**EXPECTED OUTPUT**	**ACTUAL OUTPUT**	
01	Giving input	Providing an input as video and check if it is accepted	Click the button and browse to choose a video to be input	Ce.mp4	File format unsupported	File format unsupported	Pass
02	Giving input	Providing an input as image and check if it is accepted	Click the button and browse to choose a video to be input	Ce.jpg	Accepted and further processed	Accepted and further processed	Pass

03	Giving input symbols for threshold	Alphabets, Symbols Non-Symbols	Type the input symbol	Symbols	Not accepted	Not accepted	Pass
04	Giving input symbols for threshold	Numbers	Type the input symbol	Symbols	Accepted	Accepted	Pass
05	Input video for Anomaly detection	Providing an input as video and check if it is accepted	Click the button and browse to choose a video to be input	Ad.mp4	Accepted and further processed	Accepted and further processed	Pass
06	Input image for Anomaly detection	Providing an input as video and check if it is accepted	Click the button and browse to choose a video to be input	Ad.jpg	Rejected and further processed	Rejected and further processed	Pass
07	Input image for crowd estimation	Providing an input as image and check if it is accepted	Click the button and browse to click an image to be input	Ce.jpg	Accepted and further processed	Accepted and further processed	Pass
08	Input video for crowd estimation	Providing an input as video and check if it is accepted	Click the button and browse to choose	Ce.mp4	Rejected and further processed	Rejected and further processed	Pass

			a video to be input				
09	Preview Uploaded Image for Anomaly detection	Providing an input as image and check if it is previewed	Click the button and browse to click an image to be input	Ad.jpg	Expected to previewed	Previewed	Pass
10	Preview Uploaded Image for Crowd estimation	Providing an input as image and check if it is previewed	Click the button and browse to click an image to be input	Ce.jpg	Expected to previewed	Not previewed	Fail

CHAPTER 8

RESULT AND ANALYSIS

Analysis is a specialist discipline involving systematic observations to enhance performance and improve decision making, primarily delivered through the provision of objective statistical (Data Analysis) and visual feedback (Video Analysis).

8.1 Crowd Estimation

Mean Absolute Error (MAE) is equivalent to accuracy of the model. Since the exact accuracy for a given image in crowd estimation cannot be calculated. Figure 8.1.

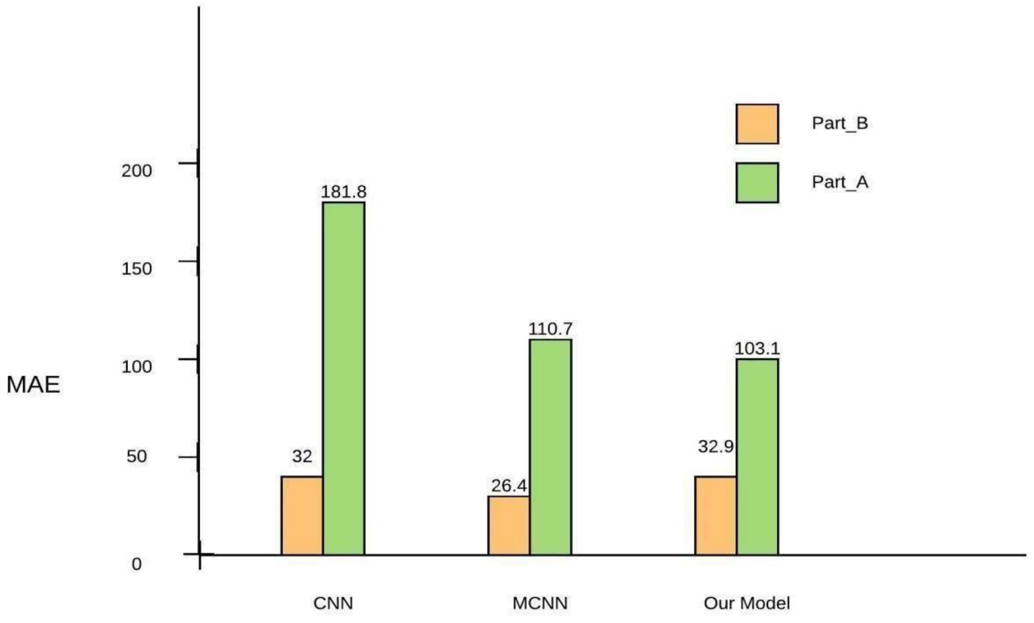


Fig 8.1: MAE for different models

The first bar graph (CNN) is plotted when a CNN model gave the estimated count directly without a density map. The MCNN is the Multicolumn CNN from the reference papers comparing MAE of the model developed with others. It shows that the MAE for part A is much better than the other two models but MAE for part B is slightly more than the other two models. When these models are run on CPU the computation time for giving the output is compared in the graph the time by 6, 8, 12, and 16-layer models considerably more for single image. Except the sixteen-layer model all the other layer models take above or around 10.

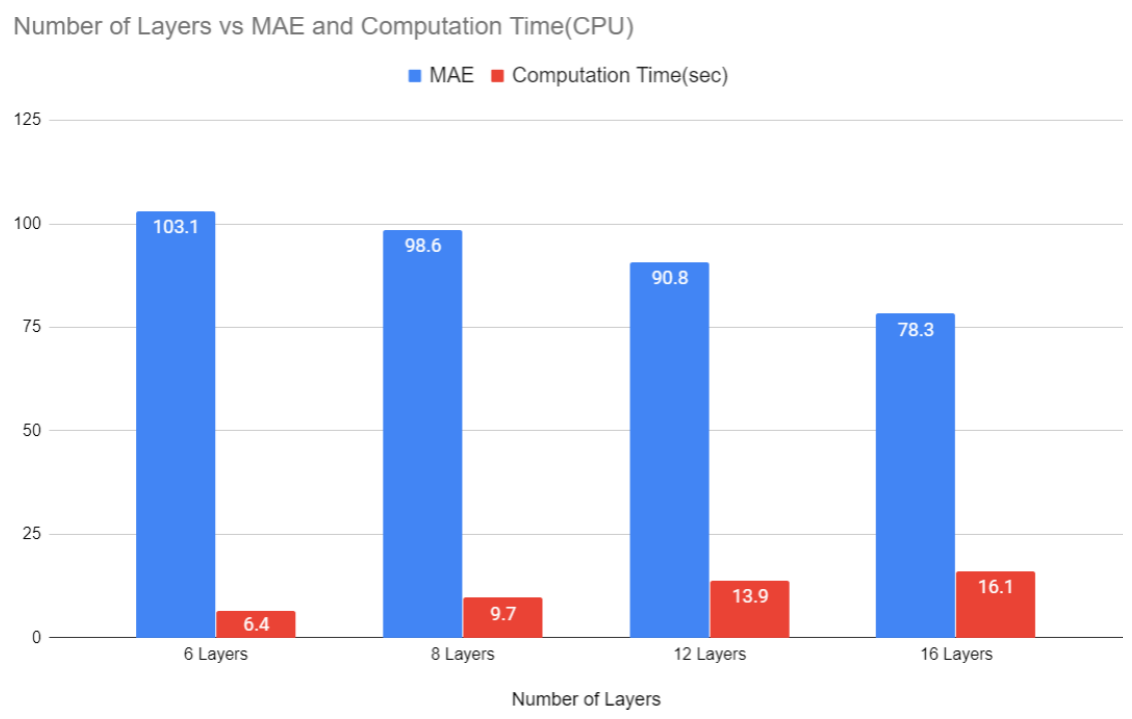


Fig 8.2: MAE and Computation Time for Different Layers

When GPU is used, the difference in the computation time between all the models reduced to a few seconds. But if a powerful GPU is used then the difference in computation time would be in the matter of milliseconds.

8.2 Anomaly Detection

Figure 8.3 represents the accuracy that was observed during training and testing phase for increase in number of epochs.

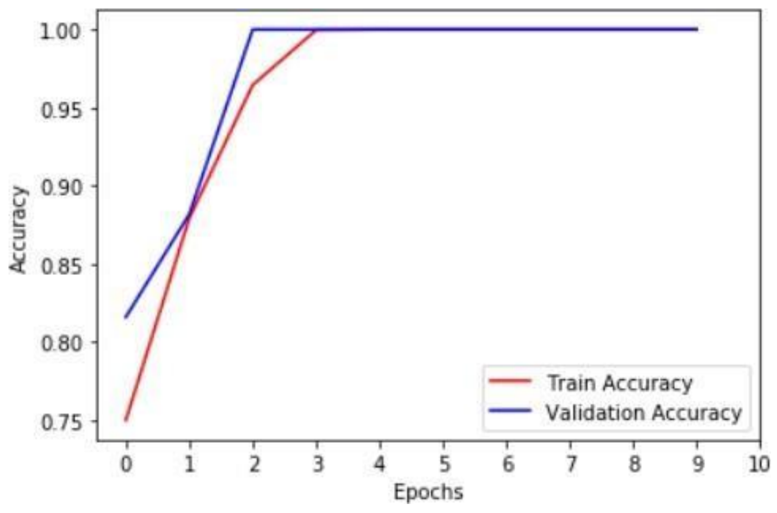


Fig 8.3: Accuracy for Training and Validation Phase

The x-axis represents the accuracy and the y-axis represents number of epochs. There is sudden increase in accuracy when the number of epochs is ranging from one to three hundred, after that it becomes constant.

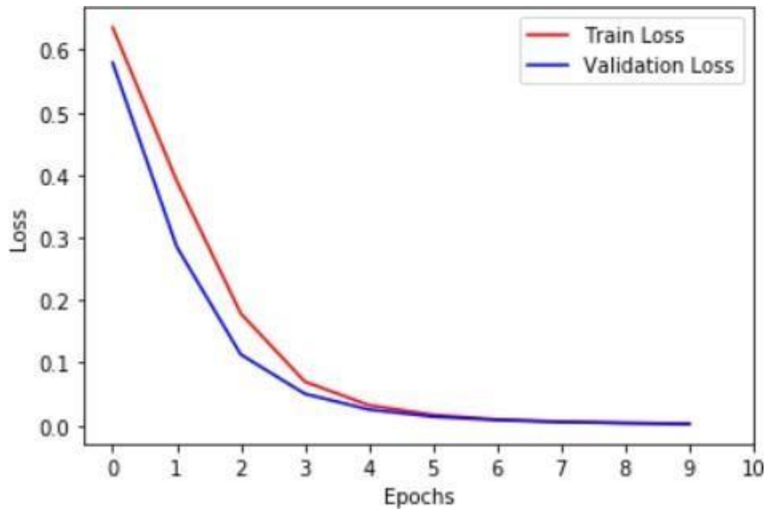
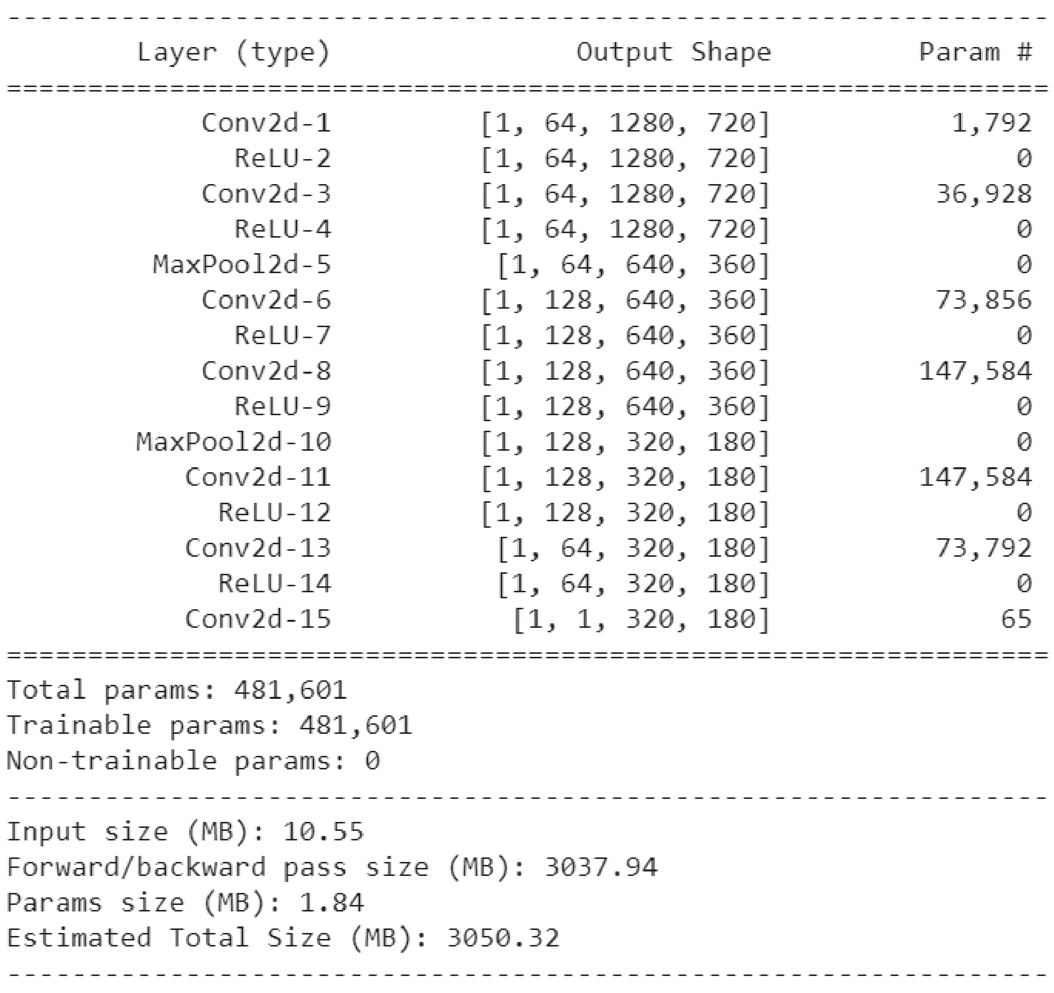


Fig 8.4: Loss for Training and Validation Phase

As the number of epochs increases, loss is minimized. Figure 8.4 represents how loss gradually decreases as the number of epochs increases. In the figure, x-axis represents the number of epochs and y-axis represents loss during training and testing phase.

8.3 Model Summary

Torch-summary provides information complementary to what is provided by print in PyTorch, similar to Tensorflow's model.summary() API to view the visualization of the model, which is helpful while debugging the neural network. In the proposed work, a similar functionality is implemented in PyTorch to create a clean and simple usable interface. The model.summary method returns ModelStatistics object to access summary data method. PyTorch is used for crowd estimation and Keras is used for anomaly detection. Keras provides a sequential model where layers in the CNN are added layer wise. The model does not use any non-trainable parameters. The input to the CNN is of the size 1280x720. Accuracy for the model is 92% with a loss of 0.0025 for training and validation. Summary for the model is shown in figure 8.5.



```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1         [1, 64, 1280, 720]           1,792
              ReLU-2         [1, 64, 1280, 720]               0
            Conv2d-3         [1, 64, 1280, 720]          36,928
              ReLU-4         [1, 64, 1280, 720]               0
         MaxPool2d-5          [1, 64, 640, 360]               0
            Conv2d-6         [1, 128, 640, 360]          73,856
              ReLU-7         [1, 128, 640, 360]               0
            Conv2d-8         [1, 128, 640, 360]         147,584
              ReLU-9         [1, 128, 640, 360]               0
        MaxPool2d-10         [1, 128, 320, 180]               0
           Conv2d-11         [1, 128, 320, 180]         147,584
             ReLU-12         [1, 128, 320, 180]               0
           Conv2d-13          [1, 64, 320, 180]          73,792
             ReLU-14          [1, 64, 320, 180]               0
           Conv2d-15           [1, 1, 320, 180]              65
================================================================
Total params: 481,601
Trainable params: 481,601
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 10.55
Forward/backward pass size (MB): 3037.94
Params size (MB): 1.84
Estimated Total Size (MB): 3050.32
----------------------------------------------------------------
```

Fig 8.5: Model Summary

CHAPTER 9

CONCLUSION AND FUTURE WORK

The chapter summarizes and concludes the investigation of crowd density estimation and anomaly detection process. It emphasizes the main points of the theoretic and practical work performed during development and presents conclusions that can be drawn from the results obtained.

9.1 Conclusion

The proposed work presents a CNN model for estimating the crowd count and performs anomaly detection in a given area. The model takes the input in the form of an image or a video. If the input is a video, then three frames are captured in equal intervals and the crowd count is calculated for those 3 frames. After training the model for several values of epoch, the accuracy was found to be greatest for an epoch value of 400. With a MAE of 103.1 and an execution time of 6.4 seconds on CPU, it is performing better than many of the existing models when run on CPU. The challenges faced include identifying or recognizing people who are very close to the camera. The accuracy of the system depends on the angle at which the image is captured. The accuracy is maximum when the image is captured directly from above but it is difficult to install the cameras at such an angle. Training the model takes a long time and hence care should be taken to not interrupt the training process.

9.2 Future Work

Future work includes implementing the given system to take a live video stream as input and display the real time crowd count. The system can be improved by integrating other anomaly detections as well. A new dataset can be built using images taken from the local surroundings which enhance the performance of the model. By applying transfer learning, other ideas can be developed and implemented to tackle different issues such as estimating the number of vehicles in a traffic signal. Accuracy and precision can be improved by changing hyper parameters such as:

- Using different Loss functions.
- Using different values of Batch size.
- Changing the learning rate.

REFERENCES

[1] Raghad Jaza Alamri, Maha Suliman Alkhuriji, Malak Saed Alshamani, Omniyyah Yahya Ibrahim and Fazilah Haron, "Al-Masjid An-Nabawi Crowd Adviser Crowd Level Estimation Using Head Detection", 2020

[2] Shiliang Pu, Tao Song, Yuan Zhang, Di Xie, "Estimation of crowd density in surveillance scenes based on deep convolutional neural network", 2020

[3] Ryan Tan, Indriati Atmosukarto and Wee Han Lim, "Video Analytics for Indoor Crowd Estimation", 2021

[4] Yingying Zhang, Desen Zhou, Siqin Chen, Shenghua Gao, Yi Ma, "Single-Image Crowd Counting via Multi Column Convolutional Neural Network", 2021.

[5] Li Biaping, Han Xinyi, Wu Dongmei, "Real-time crowd density estimation based on convolutional neural networks", 2019

[6] PyTorch https://pytorch.org/docs/stable/index.html

[7] TensorFlow https://www.tensorflow.org/guide

[8] Artificial neural network https://playground.tensorflow.org/

[9] Convlution neural networks https://poloclub.github.io/cnn-explainer/

[10] Data pre-processing https://neptune.ai/blog/data-preprocessing-guide