Deep Crowd: Estimating Crowd Density with Deep learning

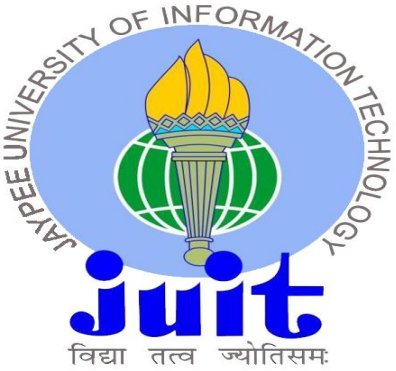
A major project report submitted in partial fulfillment of the requirement for the award of degree of

Bachelor of Technology
in
Computer Science & Engineering / Information Technology

Submitted by
Rohan Rana (201116)

Karan Hansraj (201339)

Under the guidance & supervision of
Dr. Anita



Department of Computer Science & Engineering and Information Technology
Jaypee University of Information Technology,
Waknaghat, Solan - 173234 (India)

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

PLAGIARISM VERIFICATION REPORT

Date: .....

Type of Document (Tick): PhD Thesis | M.Tech Dissertation/ Report | B.Tech Project Report | Paper

Name: _____ Department: _____ Enrolment No _____

Contact No. _____ E-mail. _____

Name of the Supervisor: _____

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): _____

_____

_____

UNDERTAKING

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

Complete Thesis/Report Pages Detail:

- Total No. of Pages =
- Total No. of Preliminary pages =
- Total No. of pages accommodate bibliography/references =

(Signature of Student)

FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found Similarity Index at..... (%). Therefore, we
are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

(Signature of Guide/Supervisor) Signature of HOD

FOR LRC USE

The above document was scanned for plagiarism check. The outcome of the same is reported below:

<table>
<tr><td>Copy Received on</td><td rowspan="4">• All Preliminary Pages
• Bibliography/Images/Quotes
• 14 Words String</td><td rowspan="2"></td><td>Word Counts</td><td></td></tr>
<tr><td rowspan="2">Report Generated on</td><td>Character Counts</td><td></td></tr>
<tr><td rowspan="2"></td><td>Submission ID</td><td>Total Pages Scanned</td><td></td></tr>
<tr><td></td><td>File Size</td><td></td></tr>
</table>

Checked by
Name & Signature Librarian

Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File) through the supervisor at plagcheck.juit@gmail.com

CERTIFICATE

This is to certify that the work which is being presented in the project report titled **"Deep Crowd: Estimating Crowd Density with Deep learning"** in partial fulfillment of the requirements for the award of the degree of B.Tech in Computer Science And Engineering and submitted to the Department of Computer Science And Engineering, Jaypee University of Information Technology, Waknaghat is an authentic record of work carried out by Rohan Rana (201116) & Karan Hansraj (201339)" during the period from July 2023 to May 2024 under the supervision of Dr. Anita, Department of Computer Science & Engineering and Information Technology, Jaypee University of Information Technology, Waknaghat.

Rohan Rana
201116

Karan Hansraj
201339

The above statement made is correct to the best of my knowledge.

Dr. Anita
Assistant Professor (SG)
Computer Science & Engineering and Information Technology

CANDIDATE'S DECLARATION

We hereby declare that the work presented in this report entitled **'Deep Crowd: Estimating Crowd Density with Deep learning'** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science & Engineering** submitted in the Department of Computer Science & Engineering and Information Technology**,** Jaypee University of Information Technology, Waknaghat is an authentic record of my own work carried out over a period from August 2023 to May 2024 under the supervision of **Dr. Anita** (Assistant Professor (SG) Computer Science & Engineering and Information Technology).

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

Rohan Rana
201116

Karan Hansraj
201339

This is to certify that the above statement made by the candidate is true to the best of our knowledge.

Dr. Anita
Assistant Professor (SG)
Computer Science & Engineering and Information Technology

Dated:

ACKNOWLEDGEMENT

Firstly, we express my heartiest thanks and gratefulness to almighty God for His divine blessing makes it possible for us to complete the project work successfully.

We are grateful and wish my profound indebtedness to my supervisor Dr. Anita, Department of Computer Science Engineering and Information Technology, Jaypee University of Information Technology, Waknaghat. Deep Knowledge & keen interest of our supervision the field of "Deep Crowd: Estimating Crowd Density with Deep learning" to carry out this project. His endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, reading many inferior drafts and correcting them at all stages have made it possible to complete this project.

We are grateful to each and every individual who directly or indirectly helped me in making this project a success.

Finally, we must acknowledge with due respect the constant support and patience of my parents and grandparents.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF GRAPHS

LIST OF ABBREVIATIONS

CNN	Convolutional Neural Network
M-CNN	Multi-Column Convolutional Neural Network
VGG	Visual Geometry Group
ML	Machine Learning
DL	Deep Learning
KNN	K-Nearest Neighbour
SVM	Support Vector Machine
ReLU	Rectified Linear Unit
IDE	Integrated Development Environment
RNN	Recurrent Neural Network
OpenCV	Open Computer Vision
OS	Operating System
CP-CNN	Contextual Pyramid Convolutional Neural Networks
Sa-CNN	Self-Attention Convolutional Neural Networks
CSRNet	Congested Scene Recognition Networks
CMTL	Cascaded Multi-Task Learning
CC-NN	Counting Convolutional Neural Networks
R-CNN	Region_based Convolutional Neural Networks
HMM	Hidden Markov Models

ABSTRACT

MCNN a multi- column convolutional neural network is the model used to study crowd density estimation in a huge project. This project aims at comprehending and forecasting diverse crowd density at both mass events like sports, and on roads like urban intersections. An MCNN model that is capable of capturing complex patterns in a hectic scene was trained and tested on a specially chosen dataset to make adjustments.

Key precision measures include Mean Absolute Error (MAE) and Mean squared error (MSE) whereby they indicate the success of the project. The aforementioned metrics play a critical role in gauging the model's accuracy in forecasting crowd density. The obtained result has shown a high accuracy with small MAE denoting very close predicted and actual value and low MSE implying good precision.

It was faced with a host of challenges such as dataset diversity, hyperparameter tuning, and constraints of implementing it in real-time. Although facing some challenges, the project has confirmed the efficiency of the MCNN model as relevant for the crowd density estimation. Further, it provides some crucial information for other deep learning and crowd analytics projects.

Therefore, this major project signifies a pivotal step towards crowd density estimation through MCNN model. The reported accuracy metrics prove the strength of this model, giving hope for applications such as crowd management, surveillance, and others. Knowledge gained from overcoming obstacles strengthens understanding in dynamic settings, preparing for further gains in inter-relationship between artificial intelligence and crowd analysis.

CHAPTER 1: INTRODUCTION

1.1 Introduction

People counting is a key topic in applications using video surveillance. Given the urgent need for crowd monitoring, determining behaviour and crowd density can improve safety and quality life. To properly comprehend situation, counting persons in a crowd by hand will be challenging and might not be enough, particularly in situations when crowd conduct could be crucial. For instance, it's critical do more than just gauge the attendance at events like stadium athletic events or sizable protests in order to identify possible security risks and enhance crowd control techniques.

Recent research has concentrated on creating methods for assessing crowd density in order to address this problem. Using these methods, one can estimate the density of a crowd by removing information from video feeds, such as activity levels, movement patterns, and crowd density.

Although crowd behavioural analysis presents numerous hurdles, such as problems with perspective distortion, occlusion, and the intricacy of crowd behaviour, it is now feasible because to recent developments in machine learning and computer vision. It has been probable to efficiently extract and evaluate crowd density data from recorded video streams and photos by using methods like trajectory clustering, optical flow analysis, and deep learning.

This study will consider the use of such methods for a crowd density estimations analysis in creating a CCN model that will provide crowd counts, classification.

An M-CNN Architecture that has been implemented can precisely assess crowd densities. Furthermore, We will investigate the effect of various characteristics on the performance of the model, such as cameras location, lighting circumstances, and crowd density, as well as will assess its efficacy using standard assessment metrics and real-world scenarios. The purpose of this research is to develop an effective system of crowd behaviour analysis that could help to enhance effective crowd control strategies in several applications contributing to ensuring public safety.

1.1 Problem Statement

1.2.1 Definition

Crowd locations are frequent around the world and in India, raising severe challenges for emergency management. Any incident that is detected in time can save countless lives and infrastructure. To efficiently estimate crowd density, we need to design a system that makes use of current infrastructure and technology. This uses two different deep learning models that may be used to track crowd densities using a camera in real time.

1.2.2Analysis of Problem Statement

To calibrate a digital camera and generate a set of projections parameters, the Tsai technique is employed. The Ground plane is the map plane formed by the blob projection. When an object is far away, and the camera's angle becomes narrower, the projected size grows and visibility improves. To some extent, this can be reduced by establishing the ground plane at the same height with the subject and using parallel plane to calculate the point when the projections cross each other. The subsequent plane is known as the Head aircraft. Blobs are formed by morphological dilatation in the vacant areas. In actuality, the two following characteristics should cause the head plane to be adjusted. 1) Anything below the Head Plane is eliminated by the twofold projection.2) HPH[2] only partially controls the initial projected area challenges at different distances, which is quite possible.

It's a simpler method for managing the quantity in big gatherings. During processing, background removal is mostly utilised to find foreground blobs. The approach is predicated on a multitude of Gaussian distributions. To construct homogeneous blobs, morphological blob dilatation and algorithm tuning are used to fill in the constant gaps left by the intrinsic statistical restrictions of algorithms.Deep networks capture the high level semantics necessary for crowd counting, which adopts an architectural style similar to the VGG-16[4] network architecture. The filters have particularly good applications in object segmentation, saliency prediction, and generic captioning. By deleting the completely linked layers utilises in VGG design[4,] pixel level foresights are achieved, allowing picture categorization difficulties to be avoided, as just one discontinuous label is supplied for the whole image.

1.3 Objectives

- Deep learning is being used to research tools as well as strategies for crowd estimation assessment.
- For various DL approaches to assess crowd density behaviour.
- Crowd density analysis measures the level of the crowd in witness footage and analyses crowd behaviour.
- Develop a deep learning based crowd prediction and modelling system.
- Significance and Motivation of the Project Work

1.4 Steps for Design a Model

1.4.1Collection of Data

- The parameters of datasets can impact our project's results.
- Next step of the project will provide the sample data for the training part.
- Utilise datasets from Kaggle, pre-collected data, UCF_QNRF, and so forth.

1.4.1.2Preparation of Data

- Collect and then organise training part of data.
- Work on cleaning up every detail that may benefit from it (removing duplicates, correcting errors, managing missing numbers, normalising, converting data types, and so on).
- Make data visualisations to help in the identification of significant connections between variables.
- Create different classes for evaluation & training.

1.4.1.3Model Selection

- Select the appropriate algorithm or model for different work which give you better results with maximum accuracy.

1.4.1.4 Model Training

- Providing an accurate results or projection as often as possible is the aim of training.
- Every step of the procedure is a training part of the phase.

1.4.1.5Model Analyzation

- Evaluate the model's objective performance using a metric or set of measurements.
- This raw data is intended to provide a reasonable representation of the model's performance in the actual world, even if it is still being improved.

1.4.1.6Parameter Tuning

- Changing the quantity and size of data, the model's accuracy, MAE, MSE and precision may be alter.
- Among the simple model hyperparameters that may be employed are initialization settings, exploring rate, training phase count, and distribution.

1.4.1.7Make Predictions

- Adding additional data which was previously excluded from the model but which has class labels allows a better understanding of the model performance as it is expected to perform in real-time conditions.

1.4.2Methodology Proposed for Deep Crowd Estimation

To improve performance, we have proposed a M-CNN and deep learning based crowd density estimate approach.

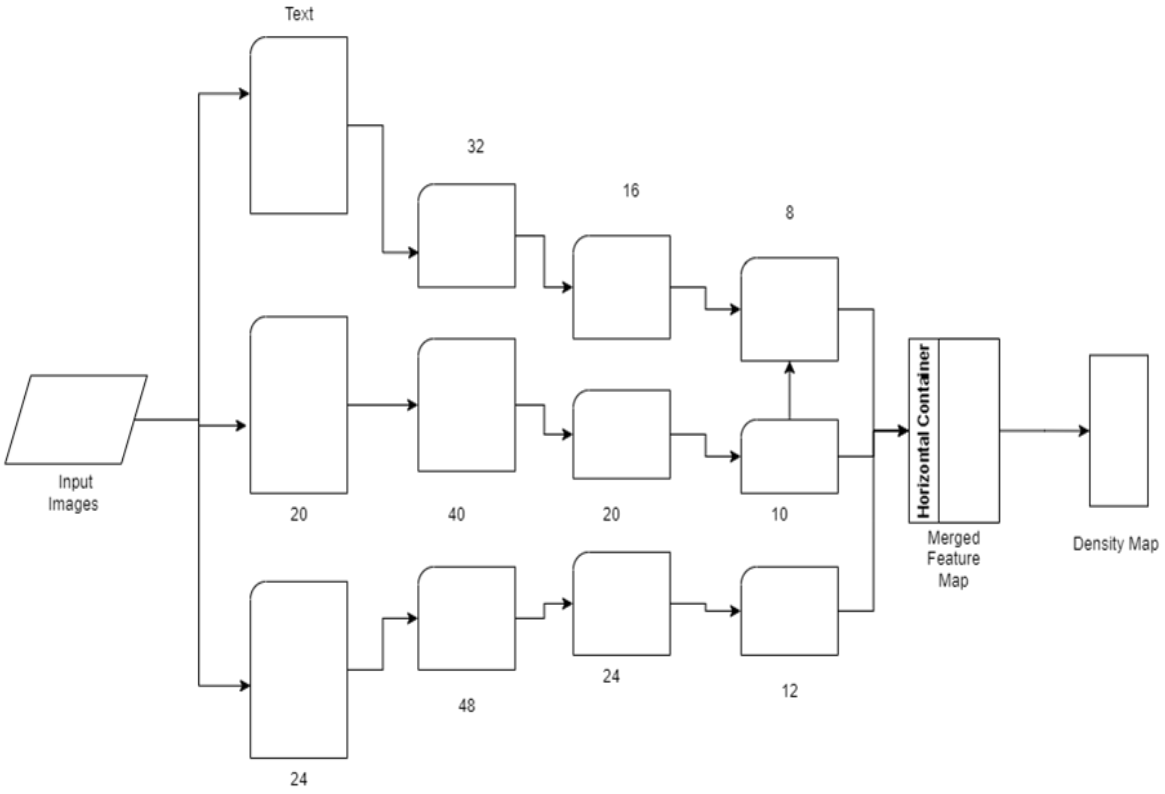


Figure 1: M-CNN

1.5 Organization

Chapter 1:

Introduction chapter provides a quick review of the project. This chapter includes a brief study overview and an explanation of Crowd Density Estimation via Deep Learning. Deep Crowd Estimation goals & the problem statement for the whole undertaking are also covered in this chapter. In addition to providing an overview regarding the deep Crowd Estimation methodology, this Introduction contains a thorough explanation of the model the project uses to categorise the crowd density with Keras & transfer learning.

Chapter 2:

Information about previous studies on crowd density estimate techniques could be found in this chapter. This chapter also covers neural networks, deep learning & machine learning.

Numerous journals and relevant publications that describe earlier work have been published. The chapter discussed the many models that have been attempted to be used by researchers in an attempt to develop an effective deep learning model for crowd density estimate. The strategies and outcomes presented in this chapter dictate the approaches we should take in order to train/create the model .

Chapter 3:

The procedures we will follow to complete the project were covered in great detail in this chapter. There is talk of both system and model development. The data set we'll be utilising is described in the chapter. All this information about the libraries that we will use is contained in chapter 3. Moreover, it hints at the type of CNN to employ. All aspects of the neural network theory are presented, thereby explaining each part supporting the entire CNN. It gives the different stages of neural networks. Talk about the instruments of measurement and validity. It also includes information about the process of designing and planning.

Chapter 4:

We cover the overall process that we took place under this project and how we tracked each step throughout. The report gives information about activities performed in different levels and the results obtained at various stages. It outlines our different modules of the model as well as the model itself. Below are the results of our performance indices that we used during this project. This includes the predictions that emerge from the created model as well as how accurate the model is. This whole part of the chapter discusses the efficiency of the entire model or the entire project.

Chapter 5:

It comprises all of the work which is the subject of the project overview. The potential scope of the project is given along with specifics about each phase. In order to advance automation in that industry, it also offers information about the project's intended use and possible application areas. It provides information about the project's enhancements as well as future directions for this project's advancement.

CHAPTER 2: LITERATURE SURVEY

Using a MCNN for Single-Image Crowd Counting[8]. The goal of this research is to create a method that, given a single photograph and a random crowd density and viewpoint, can reliably count the people in a crowd. To this end, we have put up a straightforward yet efficient MCNN architecture to maps the picture to its Crowd density map. The input image can be any size as well as resolution when using the suggested MCNN.Each column can be made to respond differently based on the size difference for reasons like perspective effect or picture resolution, by using different filters with varying receptive fields.
Furthermore, geometry-adaptive kernels—which do not require knowledge of the input image's perspective map—correctly construct the true density map. We utilised the data of 1463 pictures of about 330,000 heads labelled to sufficiently capture all the scenarios considered in our work, as existing crowd measuring datasets are not sufficient in this respect. To confirm the efficacy of the suggested model and procedure, we thoroughly test it on this difficult new dataset and all the existing datasets. Specifically, their strategy performs better than any other method when utilising the suggested basic MCNN model.

Due to the resultant distortions of perspective, the heads in the images look at distinctly different scales or densities. The reason behind this is that filters with same receptive fields cannot effectively spot such details as density in crowds of people.This makes learning the map from raw images to density maps easier natural by utilising filters with varied local receptive field widths. A series of scaled head-related filter banks are applied on each column in the proposed MCNN model thus generating relevant density maps. Larger receptive fields in filters will be useful, for instance, while carrying out density maps of sizable head areas.

Convolutional neural networks for fast crowd density estimation[9]
Using an improved convolutional neural network, the proposed approach evaluates crowd density (ConvNet). The contributions are divided into two parts. CNN is originally built to estimate crowd density. Because duplicate feature maps exist, certain network connections are removed to considerably speed up estimate. Second, to boost accuracy and speed, a

cascade of 2 ConvNet classifiers was developed. A cascade-optimized CNN is suggested in this paper as a real world solution for deep crowd estimatation.
The first method involves Multi-ConvNets technology. In the second stage, one realizes that some of those connections are not necessary and uses them to delete the duplicated similarities and their corresponding links.

The network has a light computation system that enables easy connectivity and lesser power. Because there is no standardised set of data for crowd density estimate, the approach is evaluated on 3 data sets: Such films are Pets 2009 (Ferryman & Evans, 2010), a metro clip (Ma et al. 2008; 2010) and a video on Chunxi Street in Chengdu (Zhou).

Several methodologies have been devised for Crowd behaviour analysis, which may be generally classed as follows:

- Monitoring crowd movement and flow
- Making trajectories
- Spatiotemporal Gradients

S. No.	Paper Title	Journal/Conference (Year)	Tools/Techniques/Dataset	Results	Limitations
1.	An Adaptive Multi-Scale Network Based on Depth Information for Crowd Counting	International Conference on Computer and Communication Systems (ICCCS 2023)	NF-Net, Datasets: Shanghai Tech Part A and B, UCF_CC_50	MAE (ShanghaiTech) – 56.26 MSE (ShanghaiTech) – 93.24 MSE(UCF_CC_50) – 112.7 MAE(UCF_CC_50) – 214.33	The paper does not provide detailed information on how the loss function is calculated.
2.	Single Convolutional Neural Network with Three Layers Model for Crowd Density Estimation	IEEE Access (2022)	Single CNN(S-CNN3), MATLAB Dataset: ShanghaiTech dataset	99.88% of average test accuracy and 0.02 of average validation loss.	The model's performance has only been evaluated on this particular dataset.

3.	Crowd Density Estimation by Using Attention Based Capsule Network and Multi-column CNN	IEEE Access (2021)	Multi-Column CNN, CapsNet Dataset: UCSD, ShangaiTech Part A, B and WorldExpo'10	ShangaiTech Part A – 96.1% UCSD – 88.5% WorldExpo'10 – 95.2%	It is computationally expensive to train.
4.	Crowd Density Estimation Using Fusion of Multi-layer Features	IEEE Transactions on Intelligent Transportation Systems (2021)	Novel Encoder-Decoder CNN Dataset: ShanghaiTech, WorldExpo'10, Mall Dataset, UCSD	MAE (ShanghaiTech Part A) – 69.8 MSE(ShanghaiTech Part A) – 114.7	Limited empirical research, exploration of outcomes, impact of factors
5.	Encoder-Decoder Based Convolutional Neural Networks with Multi-Scale-Aware Modules for Crowd Counting	International Conference on Pattern Recognition (ICPR) (2021)	SFANet, SegNet Dataset: -ShanghaiTech dataset, UCF_CC_50 dataset	MAE (ShanghaiTech) – 57.5 MSE (ShanghaiTech) – 94.48	Scale aware module sampling rates are fixed and not adjustable during training, potentially limiting performance in unfamiliar

6.	A Real-time Deep Network for Crowd Counting	IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (2020)	Compact CNN Dataset: ShanghaiTech dataset	MAE (ShanghaiTech Part A) – 88.1 MSE(ShanghaiTech Part A) – 141.7 MAE (ShanghaiTech Part B) – 14.9 MSE (ShanghaiTech Part B) – 22.1	The parameter size of this model is smaller, which can result in reduced performance on tasks that require deep and complex understanding.
7.	Crowd Counting Method Based on Convolutional Neural Network with Global Density Feature	IEEE Access (2019)	M-CNN Dataset: ShanghaiTech, UCF CC 50	MAE (ShanghaiTech) – 86.6 MSE (ShanghaiTech) – 129.7, MSE(UCF_CC_50) – 306.7 MAE(UCF_CC_50) – 396.3	Some backgrounds may be incorrectly counted as people, leading to noise in the estimated density map.

8.	Crowd counting via scale-adaptive CNN	IEEE Winter Conference on Applications of Computer Vision (WACV) (2018)	Sa-CNN Dataset: ShanghaiTech, UCF CC 50	MAE (ShanghaiTech) – 86.6 MSE (ShanghaiTech) – 129.7, MSE(UCF_CC_50) – 306.7 MAE(UCF_CC_50) – 396.3	It may still face challenges in extremely diverse scenarios with a wide range of scale variations.

Table 1: Literature Review

CHAPTER 3: SYSTEM DEVELOPMENT

3.1 MODEL DEVELOPMENT

3.1.1NUMPY

To work with arrays, utilize the NumPy Python module. Functions, matrix operations, and the Fourier transform are also provided for use in the field of linear algebra. Numpy, which stands for "Numerical Python," is a module that includes many ways for managing multidimensional arrays as well as multidimensional array objects. NumPy is frequently used for both mathematical and logical operations on arrays. It also covers different indexing schemes, array ways, etc. Because it's an open source project, you are free to use it for whatever. The Python utilised in math is called NumPy. Programmes may access and alter NumPy arrays more easily than lists since they are kept in a single, unified region of memory. This attribute is called location of reference in computer science. This is the main reason, and one that matters much to us, that NumPy is quicker than lists. It has also been improved to handle the CPU architectures of today.

3.1.2OS

It is considered the link among hardware and software components. For our case, we used Windows 10 and Windows 11, it’s very easy to work with them, with a comfortable command prompt, and the development is fast and safe. We also could not choose a different OS because we had one, and it would have been a Linux system or Mac if a few were available. Nevertheless, by importing a system library into Python, one can seamlessly incorporate system files, read and write instructions, read system date and time among others which will be necessary in designing and developing of these prototypes for our major project.We can make use of the os.path command to manipulate the direct of a file while open() function is simple enough to open any file saved in our computer’s file directory. Also, we may create temporary files which to be used in between until we attain the objective result.

3.1.3Random

To create random numbers, utilise Python's Random module. Please take note that these numbers are pseudo-random, meaning they were created using a non-random parameter rather than being truly random.

3.1.4Keras

Keras is an open-source, easy-to-use tool for building and evaluating deep learning models that are effective and intuitive for users. We present two frameworks for designing and training neural network models: Theano and TensorFlow. Both are designed for rapid numerical computation. It makes use of C#, Python, and C++ code libraries in addition to standalone machine learning toolkits.

While TensorFlow and Theano are very strong tools for configuring neural networks, they are also hard to understand. Keras facilitates the rapid definition of deep learning models. For applications requiring deep learning, Keras is undoubtedly the best option.

3.1.5TensorFlow

TensorFlow is a popular Python framework that Google developed and made available for fast computations and numerical applications. The TensorFlow training is beneficial for both beginners and professionals. Sentiment analysis, deep neural networks, image processing, and other advanced and foundational concepts in machine learning and deep learning are all covered in this session. A popular deep learning framework, TensorFlow. The tutorial aims at making a deep learning project utilizing TensorFlow – free, open-source software which is written in Python’s programming language.

3.1.6Matplotlib

A static visualisation may be created using the same library that is used to visualise the data.The Matplotlib Python visualisation library is excellent for 2D array displays. NumPy arrays serve as the foundation for Matplotlib, a cross-platform data visualisation tool designed to handle the bigger SciPy stack. It was initially introduced by John Hunter in 2002.

Our ability to visually access enormous amounts of data in easily comprehensible ways is one of the main benefits of visualisation. Matplotlib has a large number of plots.

3.1.7Pytorch

The Torch library serves as the foundation for PyTorch, a well-known open-source machine learning package. It is developed and maintained by the Facebook AI research group and made in IDE, making it easy to work and interface to several other libraries. PyTorch is a continuous computational graph for research and experimentation that lets programmers build and modify neural networks instantly. It also supports full GPU acceleration, which significantly speeds up deep neural network training and inference. All things considered, PyTorch is a powerful library for building ML models, with both academics and business experts are using it more frequently.

3.1.8Scikit-learn

The Scikit-learn machine learning toolbox for Python is quite popular, providing many means of pre-processing data, object recognizing, result forecasting, ordering data and choosing a model. This means that one can implement machine learning approaches based on computational science libraries such as SciPy, NumPy, and Matplotlib since it has an intuitive user interface. These include unsupervised and supervised learning algorithms such as principal component analysis (PCA), k-means clustering, random forests, decision trees, and Support Vector Machines. This also includes methods for model evaluation like cross validation and performance measures. Indeed, if we consider all this, then scikit-learn is a very functional and multi dimensional python tool of machine learning which is popular with machine learning specialists and data scientists.

3.1.9YOLOV5

The (YOLOv5) DL object identification Algorithm has gained widespread recognition for its exceptional precision and rapid inference speed. Its single-stage detector is able to forecast item bounding boxes as well as classification probabilities from input photographs only, without the use of area proposals networks or anchor boxes. YOLOv5 is easy to train and implement on low-resource devices because to its lightweight architecture and few parameters. Because it can do so quickly and accurately, it is particularly useful for identifying people in crowds.

All things considered, YOLOv5 is a powerful and versatile object identification method that has grown in favour for real-world computer vision applications, such as the detection of crowd density estimation.

3.2DATASETS USED

3.2.1ShanghaiTech Dataset

The ShanghaiTech dataset, a substantial crowd counting dataset, was released in 2016. Part-A and Part-B contain the 1198 annotated crowd photographs in total. Part-A has 482 photographs, while Part-B contains 716 pictures. The images in Part-A were downloaded from the Internet, whilst the images in Part-B were shot on the congested streets of Shanghai. In a crowd photo, each person has a label placed in the middle of their head. There are 330,165 annotated people in the collection overall.

Scientists studying crowd counting techniques can benefit from the ShanghaiTech dataset. Because it includes a wide range of crowd circumstances and is among the most challenging crowd counting datasets available, it is one of the most often utilised datasets. Using the dataset, a number of state-of-the-art crowd counting methods had been trained and assessed.

The ShanghaiTech dataset offers several benefits, including the following:

- It's a big and challenging dataset.
- Various crowd situations are discussed.
- It's well-structured and easy to use..
- It has been used to train and evaluate a large number of current crowd counting algorithms.

3.2.2Examining Dataset

When any data item does not meet established data quality criteria, its origin, quantity, and impact are verified through a separate phase of the data gathering process called a Data Quality Assessment. lifetime quality of the data. In an ongoing project, the Data Quality Assessment may be carried out only once or often to ensure the accuracy of the data.

Even if you follow strict data collecting methods and sanitise the data as it into your database, the accuracy of your data may nevertheless drastically deteriorate over time.

A data quality review may help identify records which have become inaccurate and can also help identify the origins of the data or any potential consequences that an inaccurate

record may have had. This review might assist to find any new problems and make the necessary corrections.

3.2.3Preprocessing and Outlier deletion

Data purification is one integral part machine learning. It has to do with model making. Though not sophisticated in the realm of machine learning, it's neither a conundrum nor a puzzle. There are also no loose ends and unanswered questions or twists.

However, good data cleansing can either make or mar your projects. This usually takes up a considerable part of a professional data scientist's time. Moreover, as one can ascertain from "better data beats a smart algorithm".

Often times even a simple method can provide usable results if it is applied for analysis on a good data set. Obviously, different kinds of data types must be cleaned differently. However, in this case, one should commence from systemic approach.

3.2.4Normalisation and Data Transformation

All of the data has already been smoothed and cleaned. Instead, data transformation describes the steps to transform the data in a way it can be used for machine learning. Data transformation involves combining data from their original blurry/segmented/normal state, formatting it dimension-wise, denormalizing it, and then permitting analysis.

It can be costly, time-consuming, and difficult to change data if the incorrect technological stack is in place. But conversion will guarantee the highest quality of data, which is required for precise analysis and the creation of insightful knowledge that will ultimately enable data-driven decision-making.

It's a great idea to create and train models for data analysis, and more businesses are using machine learning or intend to use it for a variety of real-world applications. But data has to be arranged such that examination of the data produces insightful information if models are to learn from it and make useful predictions.

3.2.5 COCO

In computer vision, the COCO (Common Objects in Context) dataset is a commonly used benchmark for tasks involving object detection, segmentation, and captioning. Images of intricate, ordinary scenes with segmented masks and bounding boxes labelling items are included in the COCO dataset. It includes more than 330,000 photos with over 1.5 million

occurrences of objects in 80 distinct categories. It is an invaluable tool for training and assessing computer vision models because of its diversity and size. The state of the art in a number of computer vision tasks has advanced thanks in large part to the COCO dataset. The COCO dataset has been used to train and assess numerous industry-leading models and algorithms in the fields of object detection, instance segmentation, and image captioning. Research in fields including multi-object tracking, visual relationship recognition, and image production has been sparked by its availability.

3.3 ALGORITHMS

- **Supervised Learning**

  The supervisory signal, often referred to as the input item and often in the form of a vector, and the intended output value are the two components of supervised learning. A supervised learning approach generates a function inferred from the training data, which may be applied to map fresh samples. In an ideal scenario, the algorithm could be capable to accurately determine the labels of the classes for data that are not yet apparent. In supervised learning, the main challenges come from dealing with two types of problems: classification and regression. The hardships come in when it is necessary to enable machines to classify information (classification) or to provide prediction of numerical values (regression) via supplied information data. These are the challenges faced in making sure that the model understands and predicts various kinds of outcomes properly.

- **Unsupervised Learning**

  This uses X input variables and unlabelled data as input. The learning algorithm makes predictions about the underlying structure of the information by using data that is unlabelled from such X variables. Concerns with association and clustering arise in unsupervised learning.

- **Reinforcement learning**

  Teaching a computer how to decide well can be likened to reinforcing learning by rewarding correct decisions in this case. For example, trying to maneuver through a

packed venue, or predicting how dense a place may be. In reinforcement learning, the computer learns from a certain state (being in a crowd) and then makes decisions about their density. If it guesses correctly on the crowd density and is lucky, then its reward would be “good one” otherwise it will learn from mistake.

Imagine now that the computer is learning with experimentation and finding out which technique gives it the highest rewards for predicting crowd density. It learns over-time how to make the accurate prediction adjusting itself to various conditions that are typical of crowded space. Reinforcement learning makes computers get the hang of things as it is similar to learning through experience in the aspect of crowd density estimation just as we humans learn it.

3.4 MODELS UTILIZED

- **CNN (Convolution neural Networks)**

  Crowd density estimation is somewhat similar to training a computer in numbering people in a crowd. This is tricky because crowds can be as large as that of a football game or as small as that at a picnic with friends, where people are either closely packed or spaced apart. Instead of relying on a conventional approach, one can imagine employing what is known as CNN.

  Imagine CNN as an intelligent image cop. The CNN will break the image on the second page into constituent units or elements that may be viewed as pieces or segments of a puzzle similar to how we focus on various elements of an image when shown a photograph of a crowd. It allows the computer to pick out patterns such as how people have been placed side by side.

  CNN performs very well in feature recognition such as the shape and size of people gathered at a public area. This is simply similar to the detective marking critical leads in a photograph. The computer then becomes better at noticing these patterns and clues as it analyzes more pictures to help it predict better crowd densities.

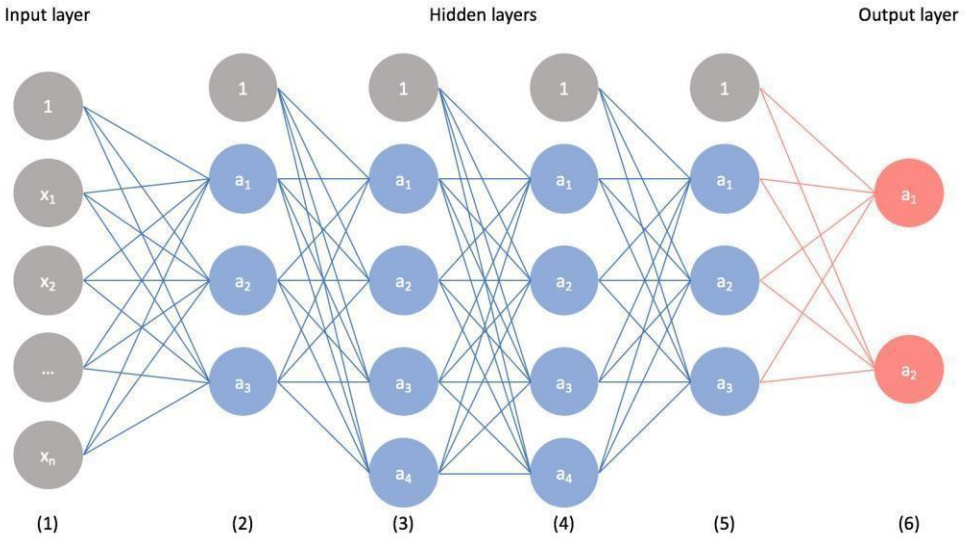


Figure 2 CNN Design

- **Functioning of CNN**

  Consider CNN as computer eyes that can work as detectives. Just as we observe details in a photograph when we break a picture into small parts, so does the case with CNN when it is provided a picture./ The small pieces are looked at for patterns such as shape and size that help the computer interpret what is going on in the image.

  Therefore, when we employ CNNs, it is like having a smart friend who has an eye for critical features in pictures. There are various ways in which CNNs can be applied successfully. For example, in distinguishing between cats and dogs or even determining the number of people in a particular place.

- **VGG 16**

  VGG16 functions as the best artist for images, specifically in estimating the number of individuals gathered in a group. In such respects, think of having a carefully described coloring book page with VGG16 as the professional going through each small area carefully.

In the area involving crowd density estimation, VGG16 dissects crowd pictures into small parts, looking closely to elements like the shapes and layouts of individuals. It's just like having an excellent crowdsourcing detective who can readily recognize the trends in the crowd.

This is known as VGG16 and acts like highlighting features within an image such as vital clues which are crucial towards recognizing object within the images. It also recognizes various forms and sizes whenever it perceives a crowd in any given picture, improving through every other scene that contains crowds.

Imagine that you have hired a visual assistant called VGG16. It learns from numerous crowd pictures and ends up becoming an expert in crowd estimation. Therefore, when discussing VGG16 in crowd density estimation, consider it your right-hand man who transforms difficult crowd images into the straightforward determination of density.

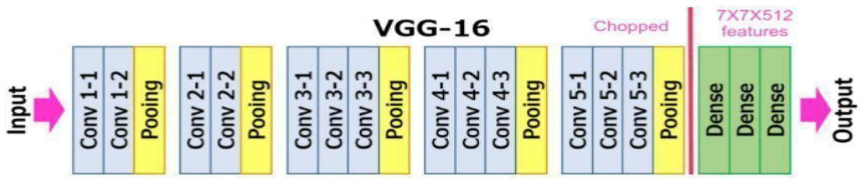


Figure 3 VGG 16

Source: Adapted from "Very Deep Convolutional Networks for Large-Scale Image Recognition," by K. Simonyan and A. Zisserman, 2015, arXiv:1409.1556.

- **VGG 19**

  VGG19 works just like a very clever artist who has been properly taught to interpret images and determine the number of people present at a particular location. It's a type of artificial intelligence which studies pictures.

  VGG19 can be likened to a talented artist that only concentrates on identification of features. It knows where to find what in the picture and almost marks essential details. It does that by studying photos of crowds thereby improving its ability to comprehend and estimate the population distribution within a site.

  Therefore, talking about the use of CrowdVGG19 for crowd density estimation is like having an intelligent art assistant that looks at images, gets educated from the pictures, and assists in estimating how congested the area is. It is like having artistic mind which becomes better in guessing the crowd numbers, with each image it interprets.

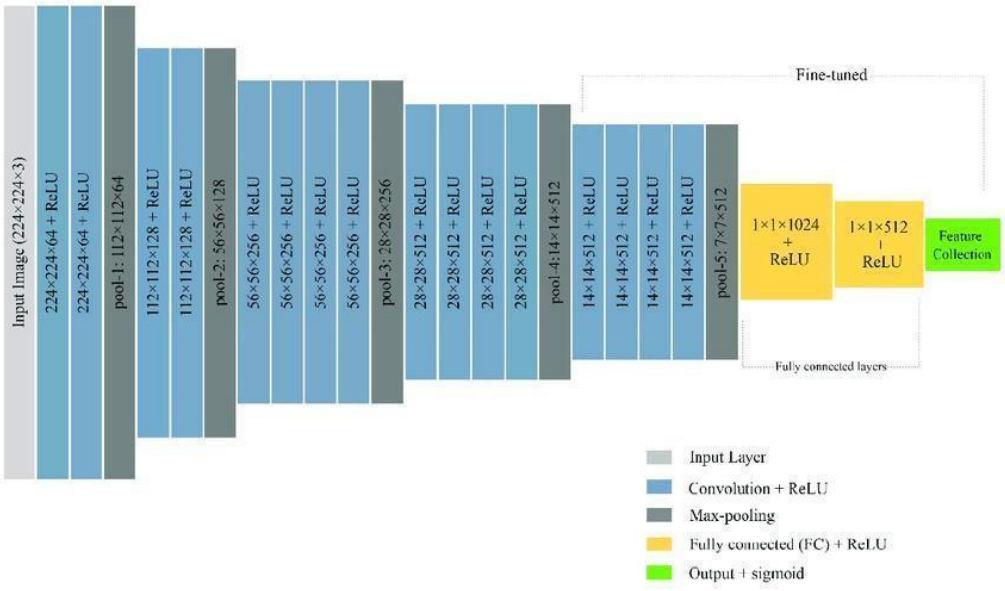


Figure 4. VGG 19

Source: Adapted from "Very Deep Convolutional Networks for Large-Scale Image Recognition," by K. Simonyan and A. Zisserman, 2015, arXiv:1409.1556

- **YOLO with Ultralytics**

  YOLOv5 is a popular object detection algorithm that has gained attention for its speed and accuracy. It is part of the You Only Look Once (YOLO) family of models, known for their real-time object detection capabilities. YOLOv5 builds upon the previous versions with improvements in speed and performance, making it suitable for a wide range of applications. UltraLytics, on the other hand, is a tool or framework that extends the functionality of YOLOv5. It likely adds features related to analytics, insights, or visualizations that enhance the utility of the object detection results. For example, UltraLytics might provide tools for tracking objects across frames, analyzing object movement patterns, or generating reports based on the detected objects.

3.4.1Project Design and Architecture

- **CNN architecture**

  One popular use for CNNs, a subclass of Deep Neural Networks, is the processing of photographs. CNNs have the ability to identify and classify certain components found in pictures. They are used in computer vision, video and image recognition, image categorization, and therapeutic image analysis.

  Convolution is the process of multiplying the two functions in mathematics to create a third function that represents the way that one function's form is changed by the other. CNN refers to this mathematical operation as a convolution. Two pictures that may be presented as matrices are multiplied to produce an output in order to extract features from a picture.

  Technically speaking, for objects that have probabilistic values ranging from 0 to 1, each input image should undergo convolutional layers along with Kernels, pooling, FC as well as SoftMax functions. This is how CNN deep learning models are

crafted and evaluated. The illustration below depicts a CNN analysis of an input image and assignment of values to its components.

3.4.1.2 Basic Architecture Used

- There are two main parts to a convolutional neural network's architecture.
- Using the feature extraction process, a convolution tool differentiates the picture's distinctive attributes for analysis.
- A layer with a fully connection, which makes conclusion about class for the picture using the previously extracted data and results of previous phase of convolution.

3.4.2 CNN Layers

Deep neural networks are structures of this sort because these layers recur frequently, indicating that our neural networks are deep.

- Raw pixel values are provided as input.
- Convolutional Layers: The CNN works to understand these visual features by detecting edges and shapes found among the layered images.
- Pooling Layers: In turn, pooling layers reduce dimensionality thereby capturing the more informative contents and increasing efficiency.
- Fully Connected Layers: The linking of each neuron from one layer to another enables the CNN to predict using the features that have been learnt.
- Activation Layers: Non-linearity is introduced via activation layers which add flexibility to CNN's decision making process as they capture complexity within the input data.

There is a noticeable rise in complexity as we progress through the stages. Though accuracy might increase, time consumption unfortunately advances as well, which makes the journey valuable.

FC layer: By concentrating on the scoring class, the highest score generated by the input digits may be ascertained.

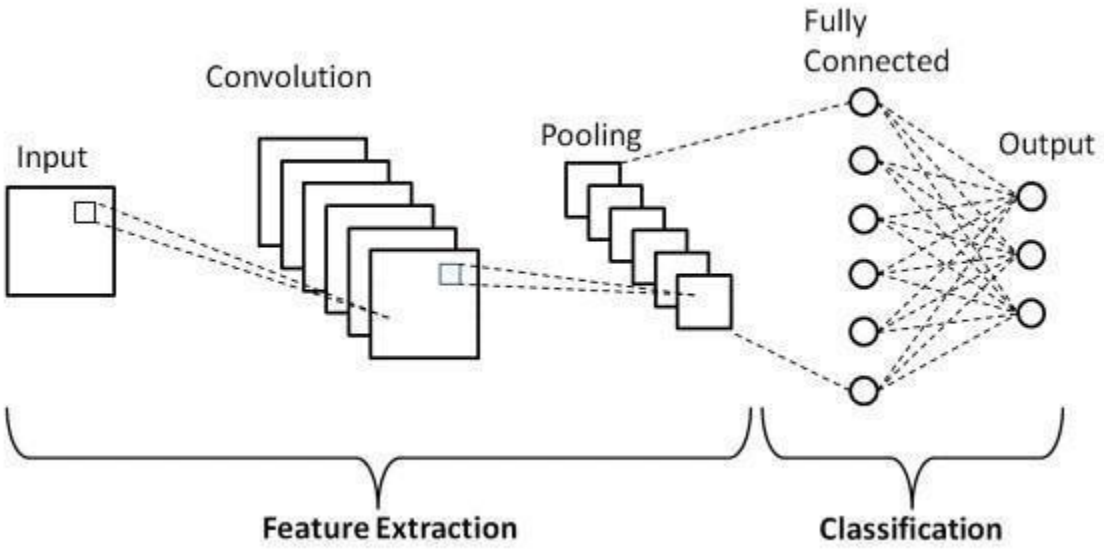


Figure 5 CNN layers

1. Convolutional Layer

First, the different properties were extracted from the incoming photographs using this layer. At this layer, the input is a MxM filter, and between the two, mathematical convolution is performed. By applying the filter on the input picture according to its size, the dot product (MxM) between the filter and the image's elements is created.

The image's borders and corners are described in great detail in the feature map. Other layers will eventually have access to this feature map, allowing them to add more features from the original picture.

2. Pooling Layer

A pooling layer is usually employed after a convolutional layer. The objective of this layer is to minimise the resulting feature map's size in order to reduce computing costs.
To do this, there are fewer connections between layers and distinct modifications made to every feature map.

Depending on the technology being used, there are several distinct pooling techniques. Reducing the size of the feature map is the most crucial aspect of max pooling. By applying average pooling, segments of an input picture segment of a specific size are averaged out.
Sum pooling is used to acquire cumulative sums of the raw data contained in the selected picture segment. It is commonly associated with the pooling layer which connects the FC layer and the convolutional layer.

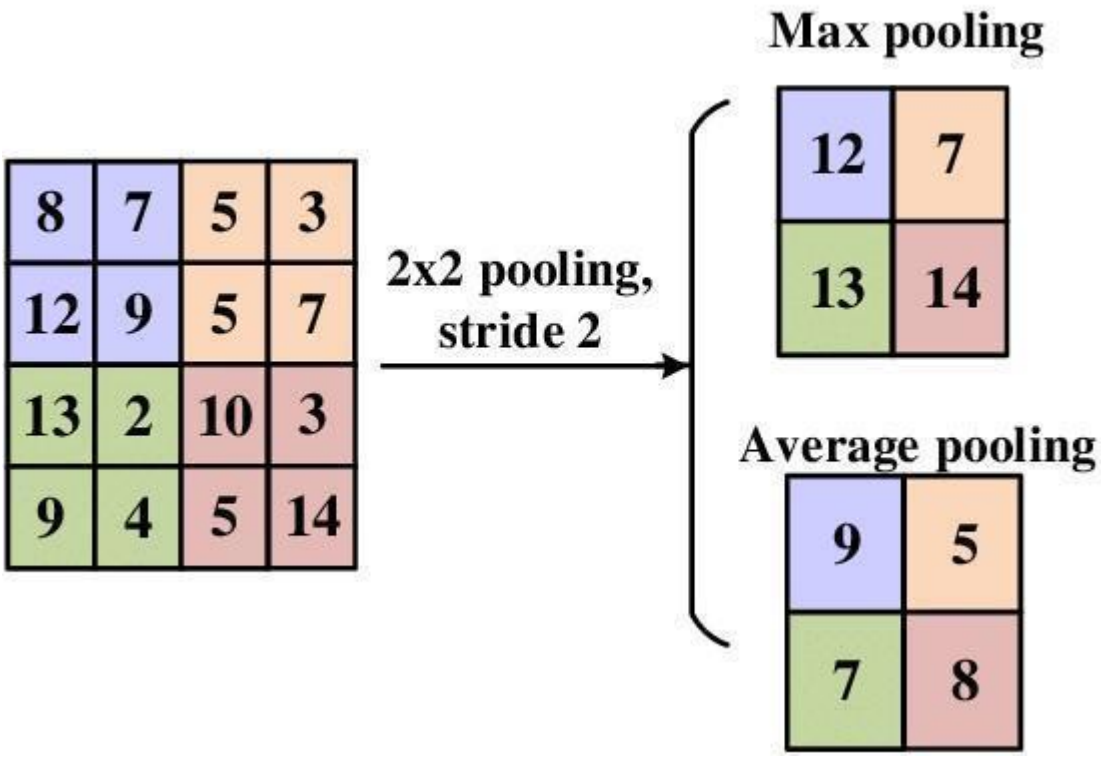


Figure 6. Pooling Layer

Source: Adapted from "Deep Learning," by I. Goodfellow, Y. Bengio, and A. Courville, 2016, MIT Press.

3. Fully Connected Layers

There is a fully connected layer that connects the neurons across the two layers with weights and biases. They generally precede output layer of a typical CNN setup.
This provides a flattened shape that consists of the input image from the layers underneath to the FC layer. The final vector is flattened and specific mathematical functions such as fine-tuned processes are then calculated prior to going through several FC steps of processing. This stage of ordering has become necessary.

4. Dropout

The presence of any traits linked to the FC layer may frequently result in fitting of the training set. This happens when a model works well using training data but poorly uses other information.
To solve this problem, the model's size is reduced by using a dropout layer that is eliminated with some neurons when the neural network is trained. Whenever 30 percent of nodes of a neural networks reach a drop in threshold of 0.3, they will be randomly removed.

5. Activation Functions

Activation functions are important for introducing non-linearity into deep neural networks particularly CNNs. The roles of these functions lie in ensuring that a neuron takes action or not. They therefore control the signal pathways through the network.

Common activation functions include:

- ReLU (Rectified Linear Unit): One of the popular approaches is known as ReLu. It involves setting any negative value to zero and preserving any positive value intact. It enables the network to identify and understand complex patterns efficiently.

- Sigmoid: These squash or compress input values which lie in the range of 0 to 1, and are usually employed for binary classification problems because they approximate probabilities.

- TanH (Hyperbolic Tangent): Like sigmoid but squashes input values between negative one and positive one with a zero centred output distribution.

- Softmax: Softmax is often used out for multi-class, typically in the output layer for converting raw scores into probabilities so as to make it easy to interpret final predictions that come with any network.

Neural networks use activation functions to model complex data relations and extract useful features during learning. Integrating dropout layers into CNNs make them capable of learning complex nature of real world data thus assisting in making correct predictions.

Selected Applications:

Today, we have powerful CNN models for object detection such as Fast R-CNN and Faster R-CNN. R-CNN is more or less a standard pipe of features used in 34 self-driving vehicles and more, facial recognition, among others.

Semantic Segmentation: Deep Parsing Network is one of the networks that was established in 2015 by a group of Hong Kong academics so as to improve on the poor performance the other network were exhibiting when it came to using segmented photos with rich information for instance the text in the images and words In addition, UC Berkeley scientists invented full convolutional networks which made semantic segmentation state-of-the-art.

3.5 IMPLEMENTATION

Generate a Density Map

```
def gen_density_map(img, anno_points):

    density_map = np.zeros_like(img, dtype=np.float64)
    h, w = density_map.shape
    kernel_size = 15
    sigma = 4.0

    for point in anno_points:

        x, y = min(w-1, abs(math.floor(point[0]))), min(h-1, abs(math.floor(point[1])))

        x1, y1 = x-kernel_size // 2, y-kernel_size // 2
        x2, y2 = x + kernel_size // 2 + 1, y + kernel_size // 2 + 1

        out_of_bounds = False
        dx1, dy1, dx2, dy2 = 0, 0, 0, 0
        if x1 <0:
            dx1 = abs(x1)
            x1 = 0
            out_of_bounds = True
        if y1 <0:
            dy1 = abs(y1)
            y1 = 0
            out_of_bounds = True
        if x2> w:
            dx2 = x2-w
            x2 = w
            out of bounds = True
    if y2> h:
        dy2 = y2-h
        y2 = h
        out_of_bounds = True

    if out_of_bounds:

        kernel_h = kernel_size-dy1-dy2
        kernel_w = kernel_size-dx1-dx2

        H = np.multiply(cv2.getGaussianKernel(kernel_h, sigma), (cv2.getGaussianKernel(kernel_w, sigma)).
        T)
    else:

        H = np.multiply(cv2.getGaussianKernel(kernel_size, sigma), (cv2.getGaussianKernel(kernel_size,
        sigma)).T)

    density_map[y1:y2, x1:x2] += H
return density_map
```

Figure 7: Code Gen Density Map

```
class DataLoader(object):
    def __init__(self, data_path, gt_path, shuffle=False, gt_downsample=False):

        self.data_path = data_path
        self.gt_path = gt_path
        self.shuffle = shuffle
        self.gt_downsample = gt_downsample
        self.data_files = [filename for filename in os.listdir(data_path)]
        self.num_samples = len(self.data_files)
        self.blob_list = []

        for fname in self.data_files:
            img = cv2.imread(os.path.join(self.data_path, fname), 0)
            img = img.astype(np.float32, copy=False)
            ht = img.shape[0]
            wd = img.shape[1]
            ht_1 = int((ht / 4) * 4)
            wd_1 = int((wd / 4) * 4)
            img = cv2.resize(img, (wd_1, ht_1))
            img = img.reshape((img.shape[0], img.shape[1], 1))
            den = pd.read_csv(os.path.join(self.gt_path, os.path.splitext(fname)[0] +'.csv'),
                              header=None).values
            den = den.astype(np.float32, copy=False)
            if self.gt_downsample:
                wd_1 = int(wd_1 / 4)
                ht_1 = int(ht_1 / 4)
            den = cv2.resize(den, (wd_1, ht_1))
            den = den * ((wd * ht) / (wd_1 * ht_1))
            den = den.reshape((den.shape[0], den.shape[1], 1))

            blob = dict()
            blob['data'] = img
            blob['gt'] = den
            blob['fname'] = fname
            self.blob_list.append(blob)

        if self.shuffle:
            np.random.shuffle(self.blob_list)

    def flow(self, batch_size=32):
        loop_count = self.num_samples // batch_size
        while True:
            np.random.shuffle(self.blob_list)
            for i in range(loop_count):
                blobs = self.blob_list[i*batch_size: (i+1)*batch_size]
                X_batch = np.array([blob['data'] for blob in blobs])
                Y_batch = np.array([blob['gt'] for blob in blobs])
                yield X_batch, Y_batch

    def get_all(self):
        X = np.array([blob['data'] for blob in self.blob_list])
        Y = np.array([blob['gt'] for blob in self.blob_list])
        return X, Y

    def __iter__(self):
        for blob in self.blob_list:
            yield blob

def mae(y_true, y_pred):
    return K.abs(K.sum(y_true) - K.sum(y_pred))

def mse(y_true, y_pred):
    return (K.sum(y_true) - K.sum(y_pred)) * (K.sum(y_true) - K.sum(y_pred))
```

Figure 8: Code Dataloader

CREATE A MODEL

```
from keras.models import Model
from keras.layers import Conv2D, MaxPooling2D, Input, Concatenate

def MCNN(input_shape=None):
    inputs = Input(shape=input_shape)

    column_1 = Conv2D(16, (9, 9), padding='same', activation='relu')(inputs)
    column_1 = MaxPooling2D(2)(column_1)
    column_1 = (column_1)
    column_1 = Conv2D(32, (7, 7), padding='same', activation='relu')(column_1)
    column_1 = MaxPooling2D(2)(column_1)
    column_1 = Conv2D(16, (7, 7), padding='same', activation='relu')(column_1)
    column_1 = Conv2D(8, (7, 7), padding='same', activation='relu')(column_1)

    column_2 = Conv2D(20, (7, 7), padding='same', activation='relu')(inputs)
    column_2 = MaxPooling2D(2)(column_2)
    column_2 = (column_2)
    column_2 = Conv2D(40, (5, 5), padding='same', activation='relu')(column_2)
    column_2 = MaxPooling2D(2)(column_2)
    column_2 = Conv2D(20, (5, 5), padding='same', activation='relu')(column_2)
    column_2 = Conv2D(10, (5, 5), padding='same', activation='relu')(column_2)

    column_3 = Conv2D(24, (5, 5), padding='same', activation='relu')(inputs)
    column_3 = MaxPooling2D(2)(column_3)
    column_3 = (column_3)
    column_3 = Conv2D(48, (3, 3), padding='same', activation='relu')(column_3)
    column_3 = MaxPooling2D(2)(column_3)
    column_3 = Conv2D(24, (3, 3), padding='same', activation='relu')(column_3)
    column_3 = Conv2D(12, (3, 3), padding='same', activation='relu')(column_3)

merges = Concatenate(axis=-1)([column_1, column_2, column_3])

density_map = Conv2D(1, (1, 1), padding='same')(merges)

model = Model(inputs=inputs, outputs=density_map)
return model
```

Figure 9: Create Model

Tracker Class for Object Tracking

tracker.py

```python
import math
class Tracker:
    def __init__(self):
        # Store the center positions of the objects
        self.center_points = {}
        # Keep the count of the IDs
        # each time a new object id detected, the count will increase by one
        self.id_count = 0

    def update(self, objects_rect):
        # Objects boxes and ids
        objects_bbs_ids = []

        # Get center point of new object
        for rect in objects_rect:
            x, y, w, h = rect
            cx = (x + x + w) // 2
            cy = (y + y + h) // 2

            # Find out if that object was detected already
            same_object_detected = False
            for id, pt in self.center_points.items():
                dist = math.hypot(cx - pt[0], cy - pt[1])

                if dist < 35:
                    self.center_points[id] = (cx, cy)
#                    print(self.center_points)
                    objects_bbs_ids.append([x, y, w, h, id])
                    same_object_detected = True
                    break

            # New object is detected we assign the ID to that object
            if same_object_detected is False:
                self.center_points[self.id_count] = (cx, cy)
                objects_bbs_ids.append([x, y, w, h, self.id_count])
                self.id_count += 1

        # Clean the dictionary by center points to remove IDS not used anymore
        new_center_points = {}
        for obj_bb_id in objects_bbs_ids:
            _, _, _, _, object_id = obj_bb_id
            center = self.center_points[object_id]
            new_center_points[object_id] = center

        # Update dictionary with IDs not used removed
        self.center_points = new_center_points.copy()
        return objects_bbs_ids
```

Figure 10: Tracker Class

3.6 KEY CHALLENGES

The crowd density estimation foray with the MCNN model has been quite educational. Nevertheless, this journey has its own set of challenges. In this chapter, we present some of the challenges that arose as we tackled this project and how these were dealt with.

- **Diverse and Representative Dataset:**
  One major problem was to assemble a dataset accurately reflecting actual crowd configurations which would manifest in reality. For the model to be able to generalise it required a wide ranging dataset including all kinds of events, locations and densities of crowds. Collecting, annotation and balancing of this data set took a considerable amount of hard work and time.
- **Hyperparameter Tuning:**
  Yet, another great challenge was that of fine-tuning the myriad of hyperparameters inside the MCNN model. These involved iteratively re-balancing between accuracy and computational effectiveness. It was an intricate process that required one to understand in detail, the complexity of model architecture as well as parameter intricacies.

- **Computational Intensity:**
  Training the MCNN model had high computational requirements. Computing power was needed because of complexity in the design and because it entailed learning through different kinds of the dataset. However, it was a matter of extreme care in managing these computational resources for prompt results.

- **Generalization to Unseen Environments:**
  Generalisation of the MCNN model to unseen environment continued to be a challenge.

CHAPTER 4: TESTING

4.1Testing Strategy

4.1.1MCNN network

Filters with the same-sized receptive fields are unsuccessful at detecting crowd density data at different scales even though perspective distortion frequently results in pictures with heads of varying proportions.

As a result, it is preferable to construct the map from raw pixels to density maps employing filters with varying local receptive field widths.

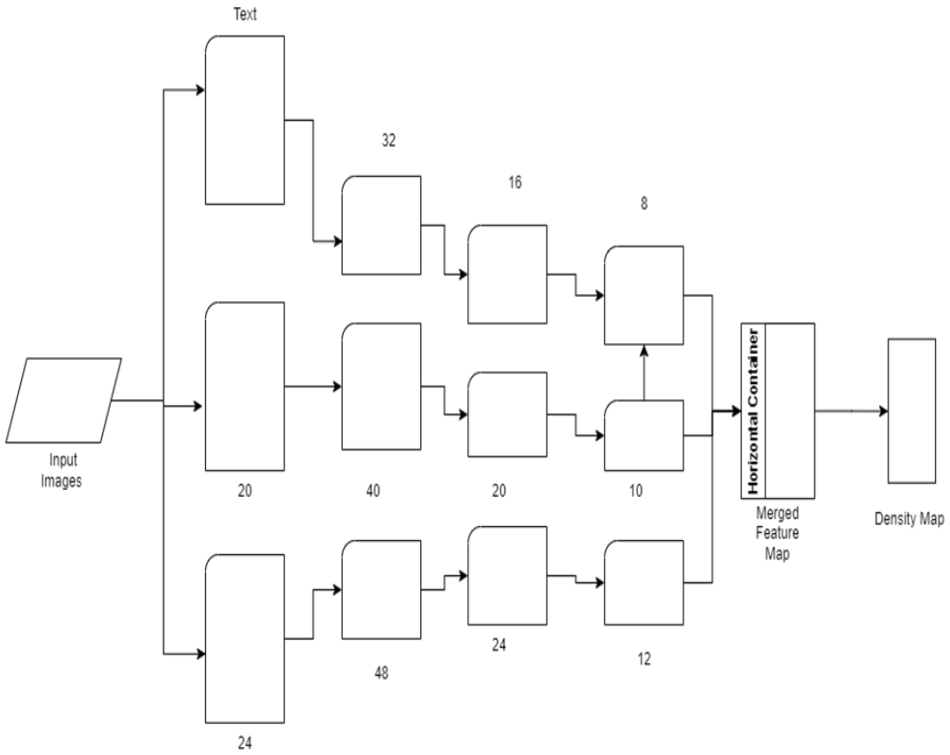


Figure 11: M-CNN Architecture

4.1.2BENEFITS OF MCNN:

MCNN is considered one of the best performing algorithms in many tasks including classification and segmentation compared to other algorithms. Think of it as the greatest algorithm out there and let me tell you why its so bright.

For instance, MCNN excels in dealing with cumbersome tasks such as measuring crowds on images or number of stuffs in pictures. This is like having a superhero who has a unique ability to see and understand all things in just an instant.

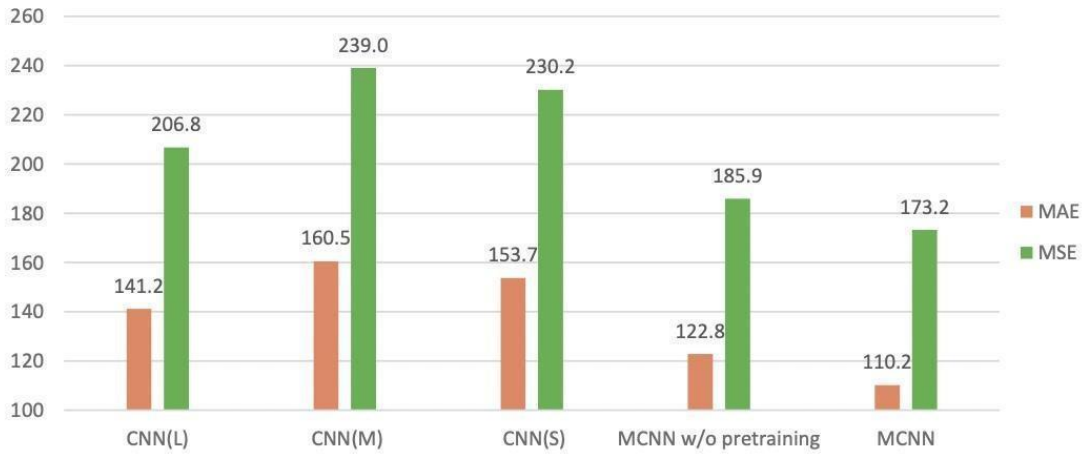
MCNN's strength, among other strengths, lies in its capacity to detect various sizes of images' features. It's as if you have a Superman with myriad pairs of eyeballs and each one of those peering into every facet of a scene. This provides MCNN to consider large but small details thereby making it efficient for tasks having varied features.

Additionally, MCNN is like a clever student – it has an ability to be adaptive and changeable depending upon the difficulty of the information delivered. This superhero algorithm seems to be ever ready for any challenges and improve on its performance making sure that it is as accurate as possible in different circumstances.

The world of algorithms likes MCNN because it is flexible, can be modified and it does not treat visual input like others do. The company feels like it has a superhero in their team showing why MCNN is mostly known for tackling complicated jobs better than other solutions.

4.1.3Benefits of MCNN over CNN:

From demonstration, MCNNs outperform single column CNNs in terms of MAE and MSE. This displays the precision of the MCNN architecture.



Graph 1. Benefits of different Design

4.1.4Performance analysis using loss functions:

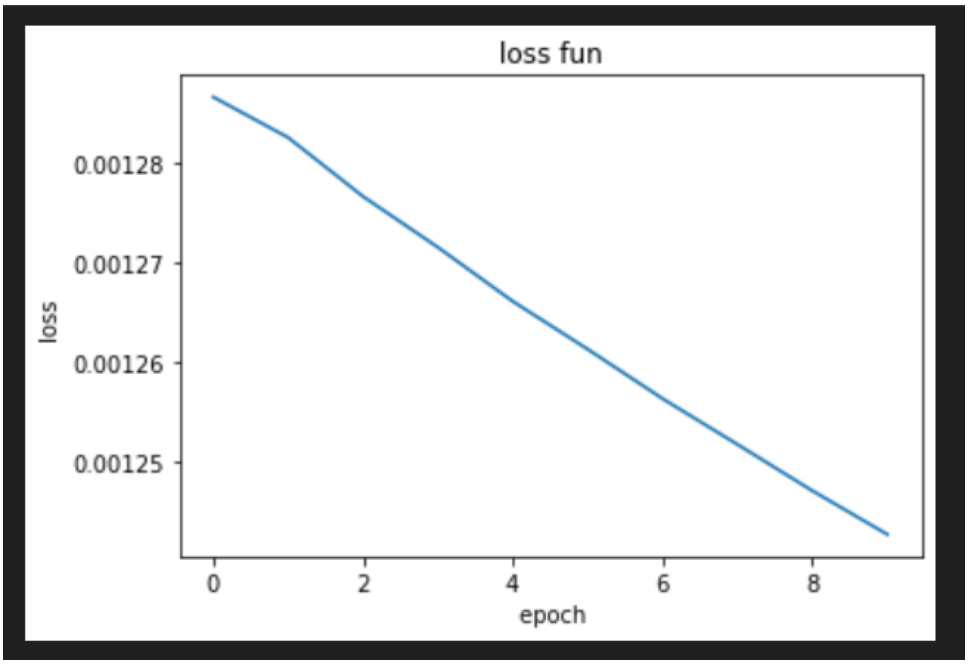
We put our method to the test on several loss functions. In additional to mapping the images to their density maps, we may directly map the photos to the overall head countings in the image.

Theta denotes the overall count of heads per image in the input set {Xi} where i = 1,…,N, while F(Xi) stands for MCNN parameters and the predicted density map. This includes the objective function follows.

$$L(\Theta) = \frac{1}{2N} \sum_{i=1}^{N} \|F(X_i; \Theta) - F_i\|_2^2,$$

The predicted density map's area, S, replaces here the ground-truth density map. Moreover, these CNNs are further pretrained in each column separately in order to compensate for this loss. Such a model can be generally referred to as crowd count regression using MCNN (MCNN-CCR).

It is evident that the crowd count regression's results are not appropriate . More of the picture's finer characteristics may be preserved by the learning density map, which enhances count precision.



Graph 2: Performance analysis using loss function

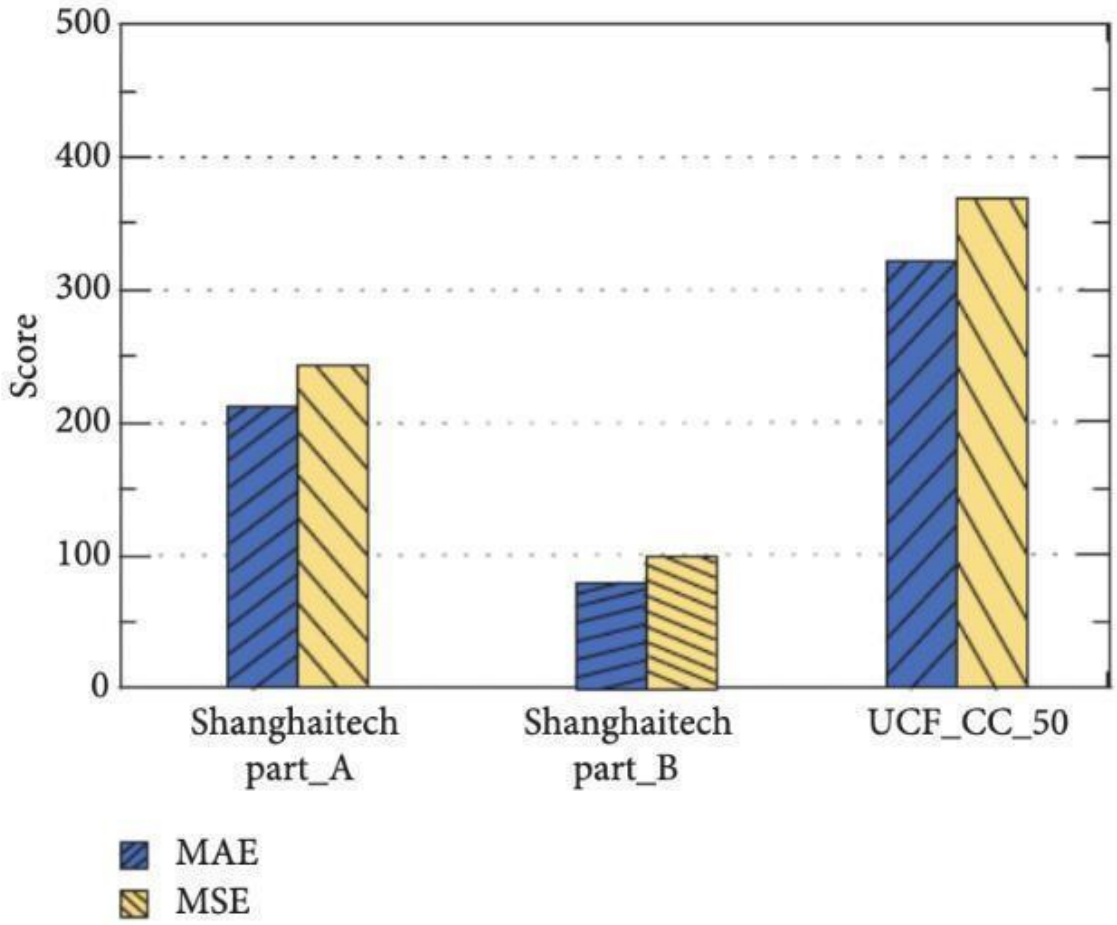
4.1.5Evaluation matrix:

We have employed the mean squared error (MSE) as well as the mean absolute error (MAE) to assess the suggested model.

$$MAE = \frac{1}{N}\sum_{1}^{N}\left|Z_i - \hat{Z}_i\right|,\ MSE = \sqrt{\frac{1}{N}\sum_{1}^{N}(Z_i - \hat{Z}_i)^2}.$$

4.1.6Testing model on different datasets:

CNN counting model was tested using the UCF CC 50 and ShanghaiTech datasets; the findings are shown:



Graph 3. Performance on different datasets

4.1.7The Implementation Of C3d Network For Analyzing Crowd Behavior

To classify whether the crowd's behaviour is aggresive or peaceful, they use a model known as C3D Model that utilises 3D convolutional neural network. Each video frame is analyzed in character to give inputs for the model. C3D is similar to 2D convolutional neural network and works with multiple frames at once.

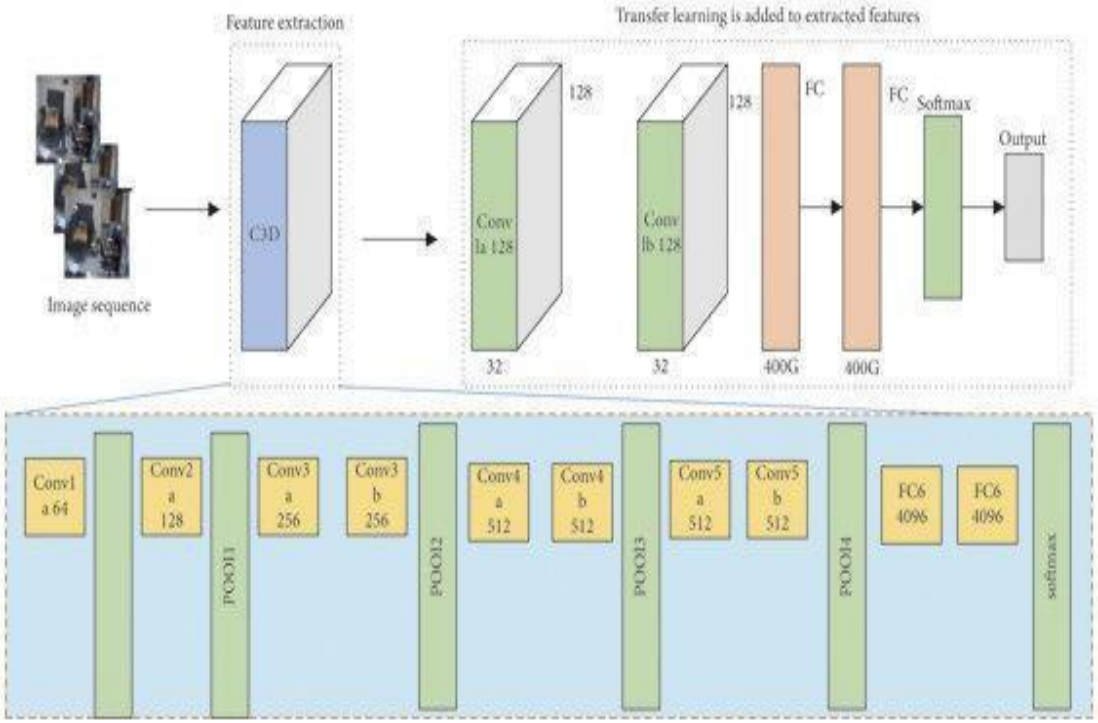


Fig 12: C3D Model architecture

4.1.8 The Implementation of YOLO Using ultralytics

The implementation and testing of an object tracking and counting system using YOLO v8 with Ultralytics, OpenCV, and custom tracking logic. The system is designed to detect and track people in a video stream, count the number of people moving up and down, and display the counts in real-time.

The object tracking and counting system successfully detects and tracks people in a video stream, providing real-time counts of people moving up and down. The implementation demonstrates the effectiveness of YOLO v8 for object detection and the custom tracking logic for object tracking. The system can be further optimized and integrated into larger projects for applications such as crowd monitoring, traffic analysis, and surveillance.

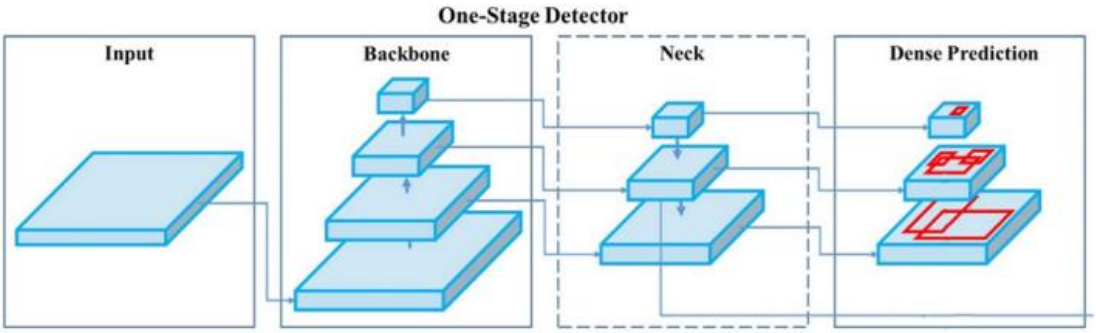


Figure 13: YOLO Architecture

Chapter 5: RESULTS AND EVALUATION

5.1 Results

Training and Testing Dynamics:

Our project's cornerstone was careful partitioning of our surroundings into train and test subsets. Such segmentation helped to make sure that the MCNN model was not only memorizing scenarios but appreciating changes in the densities of the crowds and different environments. The goal was to arm the model with flexibility in line with practical environment and enable reliable projections.

```
img_paths = ['../input/shanghaitech/ShanghaiTech/part_A/test_data/images/IMG_1.jpg',
             '../input/shanghaitech/ShanghaiTech/part_A/test_data/images/IMG_101.jpg']
for img_path in img_paths:
    img_ori = cv2.cvtColor(cv2.imread(img_path), cv2.COLOR_BGR2RGB)
    pts = loadmat(img_path.replace('.jpg', '.mat').replace('images', 'ground-truth').replace('IMG_',
    'GT_IMG_'))
    img = cv2.imread(img_path)

    sigma = 4  if 'part_A' in img_path else 15
    k = np.zeros((img.shape[0], img.shape[1]))
    gt = pts["image_info"][0, 0][0, 0][0]
    for i in range(len(gt)):
        if int(gt[i][1]) < img.shape[0] and int(gt[i][0]) < img.shape[1]:
            k[int(gt[i][1]), int(gt[i][0])] = 1

    DM = gen_density_map_gaussian(k, gt, sigma=sigma)
    fg, (ax0, ax1) = plt.subplots(1, 2, figsize=(20, 4))
    ax0.imshow(img_ori)
    ax0.set_title(str(gt.shape[0]))
    ax1.imshow(np.squeeze(DM), cmap=plt.cm.jet)
    ax1.set_title('DM -- '+str(np.sum(DM)))
    plt.show()
```

Figure 14 :Testing

Accuracy Metrics: MAE and MSE:

To gauge the performance of our MCNN model, we employed two key metrics: The MAE and the MSE. MAE made itself known as an indicator of the mean magnitude of errors whereas MSE sought to explain the difference in each instance based on the predicted and actual values that were obtained. Taken together, these metrics offered an overall test of the model's accuracy, helping us shape it into a more predictive instrument.

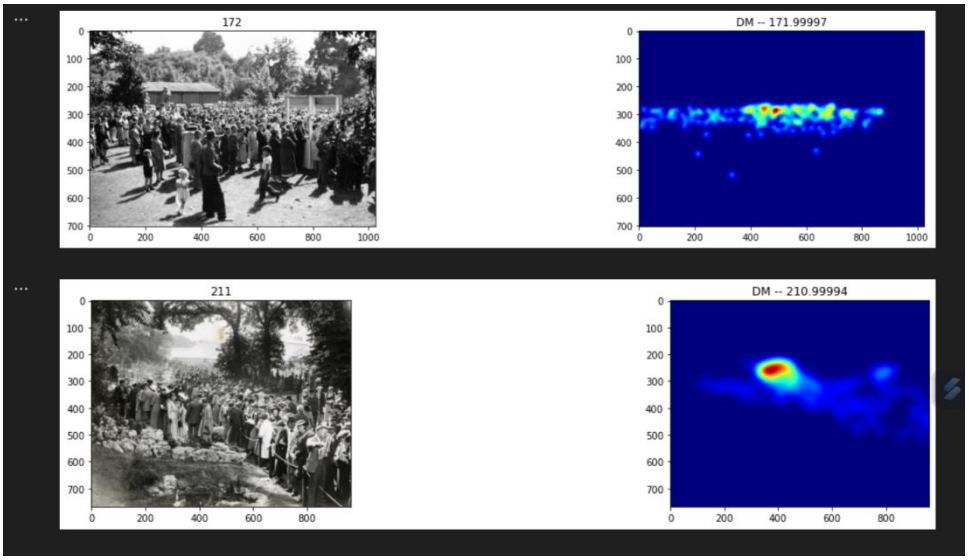


Figure 15: Density Map

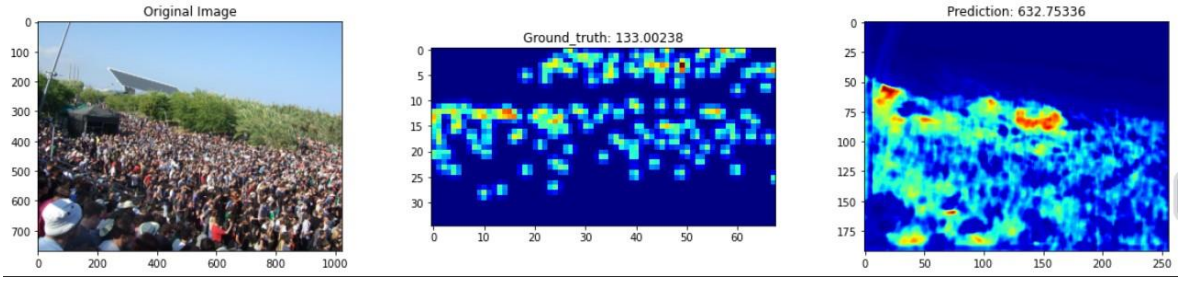


Figure 16: Predicted Density Map

Tracker Class For Object Tracking

The class named Tracker that implements a simple object tracking algorithm. This class is designed to track objects in a video stream by their bounding box coordinates. The main purpose of this class is to maintain a dictionary of object IDs and their corresponding center points.

Initialization:

• The __init__ method initializes the Tracker object.

• It initializes an empty dictionary (center_points) to store the center points of the objects being tracked.

• It also initializes a counter (id_count) to keep track of the number of unique object IDs.

tracker.py

```python
import math
class Tracker:
    def __init__(self):
        # Store the center positions of the objects
        self.center_points = {}
        # Keep the count of the IDs
        # each time a new object id detected, the count will increase by one
        self.id_count = 0

    def update(self, objects_rect):
        # Objects boxes and ids
        objects_bbs_ids = []

        # Get center point of new object
        for rect in objects_rect:
            x, y, w, h = rect
            cx = (x + x + w) // 2
            cy = (y + y + h) // 2

            # Find out if that object was detected already
            same_object_detected = False
            for id, pt in self.center_points.items():
                dist = math.hypot(cx - pt[0], cy - pt[1])

                if dist < 35:
                    self.center_points[id] = (cx, cy)
#                    print(self.center_points)
                    objects_bbs_ids.append([x, y, w, h, id])
                    same_object_detected = True
                    break

            # New object is detected we assign the ID to that object
            if same_object_detected is False:
                self.center_points[self.id_count] = (cx, cy)
                objects_bbs_ids.append([x, y, w, h, self.id_count])
                self.id_count += 1

        # Clean the dictionary by center points to remove IDS not used anymore
        new_center_points = {}
        for obj_bb_id in objects_bbs_ids:
            _, _, _, _, object_id = obj_bb_id
            center = self.center_points[object_id]
            new_center_points[object_id] = center

        # Update dictionary with IDs not used removed
        self.center_points = new_center_points.copy()
        return objects_bbs_ids
```

Figure 17: Tracker Class For Object Tracking

Figure 18: Output

Update Method:

• The update method takes a list of object bounding boxes (objects_rect) as input.

• For each bounding box in the input list, it calculates the center point of the object.

• It then iterates through the existing object IDs and their center points stored in the center_points dictionary.

• If the distance between the center point of the current bounding box and the center point of an existing object is less than a threshold (35 in this case), it considers them as the same object and updates the center point of the existing object.

• If the distance is greater than the threshold, it considers the current bounding box as a new object and assigns a new object ID (id_count) to it.

• After processing all bounding boxes, it creates a list (objects_bbs_ids) containing the bounding box coordinates, object ID, and appends it to the list.

• It then cleans up the center_points dictionary by removing IDs of objects that are no longer being tracked.

• Finally, it updates the center_points dictionary with the latest object IDs and their center points and returns the list of bounding boxes and object IDs.

```
    bbox_id=tracker.update(list)
    for bbox in bbox_id:
        x3,y3,x4,y4,id=bbox
        cx=int(x3+x4)//2
        cy=int(y3+y4)//2
        cv2.circle(frame,(cx,cy),4,(255,0,255),-1)
        #For Down
        if cy1<(cy+offset) and cy1>(cy-offset):
            cv2.rectangle(frame,(x3,y3),(x4,y4),(0,0,255),2)
            cvzone.putTextRect(frame,f'{id}',(x3,y3),1,2)
            persondown[id]=(cx,cy)

        if id in persondown:
            if cy2<(cy+offset) and cy2>(cy-offset):
                cv2.rectangle(frame,(x3,y3),(x4,y4),(0,255,255),2)
                cvzone.putTextRect(frame,f'{id}',(x3,y3),1,2)
                if counter1.count(id)==0:
                    counter1.append(id)

        #For upside#
        if cy2<(cy+offset) and cy2>(cy-offset):
            cv2.rectangle(frame,(x3,y3),(x4,y4),(0,0,255),2)
            cvzone.putTextRect(frame,f'{id}',(x3,y3),1,2)
            personup[id]=(cx,cy)

        if id in personup:
            if cy1<(cy+offset) and cy1>(cy-offset):
                cv2.rectangle(frame,(x3,y3),(x4,y4),(0,255,255),2)
                cvzone.putTextRect(frame,f'{id}',(x3,y3),1,2)
                if counter2.count(id)==0:
                    counter2.append(id)

    cv2.line(frame,(3,cy1),(1018,cy1),(0,255,0),2)
    cv2.line(frame,(5,cy2),(1019,cy2),(0,255,255),2)

    down = (len(counter1))
    cvzone.putTextRect(frame,f'Down{down}',(50,60),2,2)

    up = (len(counter2))
    cvzone.putTextRect(frame,f'Up{up}',(50,160),2,2)

    cv2.imshow("RGB", frame)
    if cv2.waitKey(1)&0xFF==27:
        break
cap.release()
cv2.destroyAllWindows()
```

Figure 19: Up-Down With_ID

Figure 20: Count Up

Figure 21: Count Down

5.2 Comparison with Existing Solutions

5.2.1Foreground segmentation

Most of the contemporary painting is built around foreground segmentation. Foreground isolation in itself is quite an involved endeavor that may result in mis segmentation which leads to a long-term negative effect on the summary of votes.
The position from where a photograph can be taken while on an assignment also varies. It is almost impossible to identify with accuracy the crowd from its background without knowing the geometrical structure and movement of the photograph. Therefore, we can only guess at how many people were in the gathering beforehand.

5.2.2Variation of scale

Because the size of the individuals in the pictures varies widely, we must mix features at many scales to reliably estimate crowd sizes for distinct images. We are compelled to employ hand-crafted features due to a shortage of tracked features, and hand-crafting features for all sizes is difficult.

5.2.3Our solution to the problem

In this study, we provide an unique convolutional neural network-based approach for crowd counting in an arbitrary still era (CNN). More precisely, we put into practise a multi-column convolutional neural network (MCNN) model for image categorization, which was motivated by the research of [8]. Their methodology enables any number of columns to be trained on inputs that have passed different pre-processing stages. The average of the various predictions made by each deep neural network is then used to determine the final predictions. Our MCNN consists of three rows with different sized filters.

Medium, small, and big filter or better said wide receptive fields are referred as different types of filters. In such a situation, it is advisable to employ a single column design, with

each element understanding its unique elements. CNN is highly tolerant of a change in head or person size due to projection effects and at different image resolutions.

For the proposed MCNN model, we replace the fully connected layer with a convolutional layer of 1 x 1 filter size. To avoid distortion, we could use a different size as an input image for our model. First, the network estimates the crowd density which gives us an eventual total.

CHAPTER 6: CONCLUSION AND FUTURE SCOPE

6.1 CONCLUSION

I employed one-shot MCNN that estimates crowd occupancy in a single image under diverse viewpoints and settings. In order to cross-check my findings against other researches, I adopted the use of ShanghaiTech dataset which consists of 330, 165 individuals that have been categorized into two main classifications. It is among the most prominent heaads for density-based spatial clustering. The result of our method is better than state-of-the-art on all test sets.

Moreover, the robustness of the proposed model is demonstrated since this model, with minor adjustments in the parameters and final layers, can be applied to various purposes across different sectors.

6.2 FUTURE SCOPE

The future of this crowd density estimation project using MCNN has many things to come for improvement. The further work on MCNN includes the enhancement of the coding and behavioral analysis towards a holistic comprehension on crowd dynamics.Model

- **Refinement and Optimization:**
  Keep refining MCNN model in order to improve its accuracy and effectiveness. Experiment further architectural improvements or sophisticated approach to enhance model efficiency.Explore the inclusion of attention mechanisms to focus on relevant features for improved crowd density predictions.

- **Dataset Expansion and Diversity:**
  Extend the dataset using other crowd scenarios that will include different behaviors, crowding densities, and environmental conditions. Identify and address the biases in the current dataset to cover and account for the real-world scenarios.

- **Behavioral Analysis Integration:**
  Integrate some behavioral analysis components into the current MCNN framework. Develop new ways to capture crowd interactions, movement and response patterns. Develop algorithms or models to interpret/translate behavioral cues in crowds.

- **Real-time Implementation and Deployment:**
  Ensure that the MCNN model optimizes real-time performance, making fast predictions in dynamic settings.
  Investigate deployment methods in actual scenarios, like public events or urban surveillance, to verify the applicability of the model in real-life applications.

- **User Interface and Visualization:**
  Create an intuitive user interface for an easy interaction with the MCNN model. Use visualizations to show crowd density predictions so that users can understand and use the model's results properly.

REFERENCES

[1] P. Zhang, W. Lei, X. Zhao, L. Dong, and Z. Lin, “An Adaptive Multi-Scale Network Based on Depth Information for Crowd Counting,” Sensors, vol. 23, no. 18, p. 7805, Sep. 2023.

[2] A.Alashban, A. Alsadan, N. F. Alhussainan and R. Ouni, "Single Convolutional Neural Network With Three Layers Model for Crowd Density Estimation," in *IEEE Access*, vol. 10, pp. 63823-63833, 2022.

[3] A. Kizrak and B. Bolat, "Crowd Density Estimation by Using Attention Based Capsule Network and Multi-Column CNN," in *IEEE Access*, vol. 9, pp. 75435-75445, 2021.

[4] X. Ding, F. He, Z. Lin, Y. Wang, H. Guo and Y. Huang, "Crowd Density Estimation Using Fusion of Multi-Layer Features," in *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 8, pp. 4776-4787, Aug. 2021.

[5] P. Thanasutives, K. -i. Fukui, M. Numao and B. Kijsirikul, "Encoder-Decoder Based Convolutional Neural Networks with Multi-Scale-Aware Modules for Crowd Counting," 2020 25th International Conference on Pattern Recognition (ICPR), Milan, Italy, 2021.

[6] X. Shi, X. Li, C. Wu, S. Kong, J. Yang and L. He, "A Real-Time Deep Network for Crowd Counting," *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Barcelona, Spain, 2020, pp. 2328-2332.

[7] Z. Liu, Y. Chen, B. Chen, L. Zhu, D. Wu and G. Shen, "Crowd Counting Method Based on Convolutional Neural Network With Global Density Feature," in *IEEE Access*, vol. 7, pp. 88789-88798, 2019.

[8] L. Zhang, M. Shi and Q. Chen, "Crowd Counting via Scale-Adaptive Convolutional Neural Network," 2018 IEEE Winter Conference on Applications of Computer Vision (WACV), Lake Tahoe, NV, USA, 2018, pp. 1113-1121.

[9] K. Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biological cybernetics, 36(4):193–202, 1980.

[10] W. Ge and R. T. Collins. Marked point processes for crowdcounting. In CVPR, pages 2913–2920. IEEE, 2009.

[11] G. Hinton, S. Osindero, and Y. Teh. A fast learning algorithm for deep belief nets. NEURAL COMPUT, 18(7):1527–1554, 2006.

[12] H. Idrees, I. Saleemi, C. Seibert, and M. Shah. Multi-source multi-scale counting in extremely dense crowd images. In CVPR, pages 2547–2554. IEEE, 2013.

[13] H. Idrees, K. Soomro, and M. Shah. Detecting humans in dense crowds using locally-consistent scale prior and global occlusion reasoning. Pattern Analysis and Machine Intelli-gence, 2005.

[14] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Gir-shick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. arXiv preprint arX-iv:1408.5093, 2014.

[15] D. Kong, D. Gray, and H. Tao. Counting pedestrians in crowds using viewpoint invariant training. In BMVC. Cite-seer, 2005.

[16] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. Proceed-ings of the IEEE, 86(11):2278–2324, 1998.

[17] V. Lempitsky and A. Zisserman. Learning to count objects in images. In Advances in Neural Information Processing Systems, pages 1324–1332, 2010.

[18] M. Li, Z. Zhang, K. Huang, and T. Tan. Estimating the number of people in crowded scenes by mid based foreground segmentation and head-shoulder detection. In ICPR, pages 1–4. IEEE, 2008.

[19] Z. Lin and L. S. Davis. Shape-based human detection and segmentation via hierarchical part-template matching.Pattern Analysis and Machine Intelligence, 32(4):604–618,2010.

[20] B. Liu and N. Vasconcelos. Bayesian model adaptation for crowd counts. In ICCV, 2015.

[21] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. arXiv preprint arX-iv:1411.4038, 2014.

[22] A. Marana, L. d. F. Costa, R. Lotufo, and S. Velastin. On the efficacy of texture analysis for crowd monitoring. In International Symposium on Computer Graphics, Image Processing,and Vision, pages 354–361. IEEE, 1998.

[23] N. Paragios and V. Ramesh. A mrf-based approach for real-time subway monitoring. In CVPR, volume 1, pages I–1034.IEEE, 2001.

[24] V. Rabaud and S. Belongie. Counting crowded moving objects. In CVPR, volume 1, pages 705–711. IEEE, 2006.

[25] C. S. Regazzoni and A. Tesei. Distributed data fusion for real-time crowding estimation. Signal Processing, 53(1):4763, 1996.

DrillBit

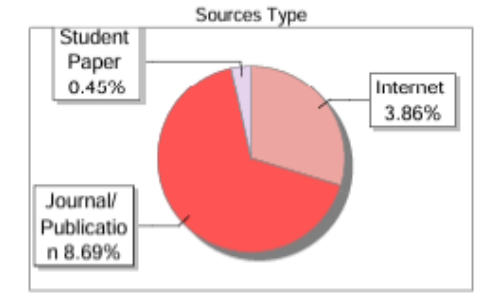
The Report is Generated by DrillBit Plagiarism Detection Software

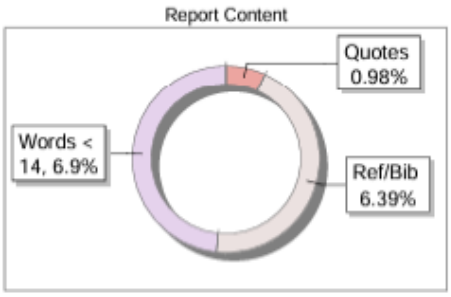
Submission Information

Author Name	ROHAN
Title	CROWD
Paper/Submission ID	1794393
Submitted by	anita@juit.ac.in
Submission Date	2024-05-13 11:06:36
Total Pages, Total Words	64, 10136
Document type	Research Paper

Result Information

Similarity **13 %**





Exclude Information

Quotes	Not Excluded
References/Bibliography	Excluded
Source: Excluded < 14 Words	Not Excluded
Excluded Source	**0 %**
Excluded Phrases	Not Excluded

Database Selection

Language	English
Student Papers	Yes
Journals & publishers	Yes
Internet or Web	Yes
Institution Repository	Yes

A Unique QR Code use to View/Download/Share Pdf File