

OBJECT DETECTION ON DENSITY MAPS BASED ON SPECTRAL ANALYSIS

Submitted in partial fulfillment of the requirements for the
award of
Bachelor of Engineering degree in Computer Science and
Engineering

by

Manukonda Saikrishna(37110435)
Kancharla madhav chowdary(37110311)



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING**

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with Grade “A” by NAAC

**JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI
- 600 119**

MARCH - 2021



SATHYABAMA



INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with “A” grade by NAAC

Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai – 600 119

www.sathyabama.ac.in

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
BONAFIDE CERTIFICATE**

This is to certify that this Project Report is the bonafide work of **MANUKONDA SAI KRISHNA (REG NO. 37110435)** and **KANCHARLA MADHAV CHOWDARY (REG NO.37110311)** who have done the Project entitled “**OBJECT DETECTION ON DENSITY MAPS BASED ON SPECTRAL ANALYSIS**” under my supervision from **August 2020 to march 2021**

Internal Guide

Dr.R..SUBHASHINI,M.E.,PH.d.,

Head of the Department

Submitted for Viva voce Examination held on_____

Internal Examiner External Examiner

DECLARATION

I **MANUKONDA SAI KRISHNA** hereby declare that the Project Report entitled
“ **OBJECT DETECTION ON DESNITY MAPS BASED ON SPECTRAL ANALYSIS**”
done by me under the guidance of **Dr. R.Subhashini,M.E.,Ph.d.**, is submitted in
partial fulfillment of the requirements for the award of Bachelor of engineering degree
in Computer Science And Engineering

DATE:

PLACESIGNATURE OF CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T.Sasikala M.E., Ph.D, Dean, School of Computing ,Dr. S. Vigneswari, M.E., Ph.D., and Dr. L. Lakshmanan, M.E., Ph.D., Head of the Department** of Information Technology for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr.R.Shubhashini.M.E,Ph.D.**, for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer science and engineering** who were helpful in many ways for the completion of the project.

Abstract

Object detection (OD) in crowded scenes is a challenging task since objects are densely distributed and partially overlapped. In this paper, we propose a novel OD method by fully exploring the information provided by the image and its estimated density map. Our proposed OD method consists of two main stages. Initial object locations are firstly computed based on object spatial distribution information obtained from the estimated density maps. Inspired by the human visual attention mechanism, a saliency map which offers object boundaries is then employed to accurately estimate the bounding boxes with the support of the estimated initial object locations.

We propose the task to estimate the density map of objects from single image with unknown perspective map. We follow the recent progress in object counting through density map estimation. Object density map estimation is usually suffered from scale variance of objects caused by unknown perspective. In addition, the background and irrelevant objects in the image lead to artifacts in the resulting density maps, which build up the error when heatmaps are needed by aggregating density maps.

TABLE OF CONTENT

S.NO.	TITLE	PAGE.NO
	ABSTRACT	
	LIST OF FIGURES	
1	INTRODUCTION	
2	LITERATURE REVIEW	
3	SYSTEM DESIGN AND METHODOLGY	
	3.1 Existing system	
	3.2 Proposed system	
	3.3 System Requirement	
	3.4 software description	
4	SOFTWARE DEVELOPMENT METHODOLOGY	
	4.1 system design	
	4.2 UML diagram	
	4.3 Architecture	
	4.4 Module Descriptions	
5	RESULT & DISCUSSION	
6	CONCLUSION & FUTURE	

	ENHANCEMENT	
	6.1 Conclusion	
	6.2 Future Enhancement	
	REFERENCE	
	APPENDIX	
	Source Code	
	Snap shots	
	Journal Paper	
	Plagiarism report	

LIST OF FIGURES

S.NO.	TITLE	PAGE NO.
3.4	PYTHON	8
3.4	PYTHON	9
4.3	ARCHITECTURE	29
	SNAP SHOTS	
	PREDICTED	53
	DENSITY MAP	53
	DATASET	54
	DENSITY ARRAY VALUES	54
	TRAINING EPOCHS	55
	CALCULATING VALIDATED MAE & MSE	55
	FINAL MAE & MSE	56

Chapter 1: Introduction

Object detection is a fundamental problem in computer vision, which is critical for surveillance applications, e.g., face detection and pedestrian detection. Deep learning-based architectures have now become the standard pipelines for general object detection. One of the traditional methods is to detect each individual in the image, which does not work well in highly congested scenes. Density estimation. Density generation. Input image. Density map. Dot map. Figure 1. Object detection on density maps: the ground-truth crowd density map is generated from the ground-truth dot annotations of people. Given an input image, a model is trained to predict the density map, which is summed to obtain the predicted count. Current approaches treat the density map generation as a fixed intermediate representation, which is based on hand-crafted designs. In this paper, we propose to jointly learn the density estimator and the density generator. Another method is to directly estimate the final count based on hand-crafted features, which can only be applied to simple scenarios. Current state-of-the-art methods use row density maps to achieve superior counting performance. Density maps are an intermediate representation, where the sum over any region in the density map indicates the number of people in the region. First, the density maps are generated from the dot annotations, where each dot indicates a person's location. Second, given the input image, algorithms are designed to predict the density map (see Figure 1), which is then summed to obtain the count. In this paper, we call these two steps density map generation and density map estimation, respectively. Most works focus on density map estimation and ignore density map generation. Many different deep

networks have been proposed to improve density map estimation, e.g., using different kernel sizes or image pyramids to handle scale variations, or using context or prior information to handle occlusions. Although density map estimation is well-studied, the generation of density maps is often overlooked and uses handcrafted designs without adequate investigation and analysis. The simplest approach to obtain a density map is to convolve the annotation dot map with a Gaussian with fixed width, i.e., place a Gaussian on each dot. Other works scale the Gaussian bandwidth according to the scene perspective, or adaptively use the local congestion level (or distance to nearest neighbors). Some works use human-shaped kernels, composed of two Gaussians, but this is less popular since the body of the person is often occluded in crowd images. In practice, the method for generating the density maps is crucial for crowd counting. Improperly generated density maps may dramatically hurt the counting performance. The choice of the kernel bandwidth or kernel shape used to generate the density map is often dataset dependent, and such choices often do not work across different datasets. In the era of deep learning, we may consider current density maps as a handcrafted intermediate representation, which is used as a target for training deep networks to count. From the standpoint of end-to-end training, these hand-designed intermediate representations may not be optimal for the particular network architecture and particular dataset. In this paper, we take the first step towards learnable density map representations, which are jointly trained with the density map estimator (counter). We propose two methods: 1) density map refinement, which is trained to improve existing traditional density maps; 2) adaptive density map generation, which learns a novel

density map representation using the annotation dot map as input. Both methods are jointly trained with the density map estimator, yielding superior performance compared with using traditional density maps.

Chapter 2: Literature Survey

Literature Survey 1	
Title	Tracking People in Dense Crowds using Supervoxels
Authors	Shota Takayama, Teppei Suzuki, Yoshimitsu Aoki, Sho Isobe, Makoto Masuda
Published Year	October 2018
Efficiency	• High performance is achieved • The tracking accuracy is 81.42% with mask area
Drawbacks	• High levels of inter-pedestrian occlusions cause tracking failure • Double-person detector increases the rate of false detection
Description	The demand for people tracking in dense crowds is increasing, but it is a challenging problem in the computer vision field. "Crowd tracking" is extremely difficult because of hard occlusions, various motions and posture changes. In particular, we need to handle occlusions for more robust tracking. This paper discusses robust crowd tracking based on a combination of supervoxels and optical flow tracking. The SLIC based supervoxel algorithm adaptively estimates the boundary between a person and a background. Therefore, the combination of supervoxels and optical flow tracking becomes a highly reliable approach for crowd tracking. In tracking experiments, high performance is achieved for the UCF crowd dataset.
Literature Survey 2	
Title	Scale-adaptive Real-time crowd detection and counting for drone images
Authors	Markus Kuchhold, Maik Simon, Volker Eiselein and Thomas Sikora
Published Year	February 2018
Efficiency	• Extremely fast counting strategy with high accuracy

	• Receives very precise counting results for different zoom factors, resolutions
Drawbacks	• Scale between different frames can vary because the drone • Non-crowded areas reducing the accuracy
Description	We propose a scale-adaptive crowd detection and counting approach for drone images. Based on local feature points and density estimation considering the image scale, we detect dense crowds over multiple distances and introduce an extremely fast counting strategy with high accuracy for our detected crowd regions. We compare our results with a recent CNN-based state-of-the-art approach and validate both methods for different scaling factors on a novel crowd dataset. The results show that our proposed method outperforms the pretrained CNN-based approach and receives very precise counting results for different zoom factors, resolutions and crowd sizes. Its low computational complexity makes it highly suitable for real-time analysis or embedded systems.
Literature Survey 3	
Title	A Large Scale Crowd Density Classification using Spatio-Temporal Local Binary Pattern
Authors	Sonu Lamba, Neeta Nain
Published Year	October 2017
Efficiency	• Low computational complexity with high efficiency • Facial expression recognition system by utilizing its temporal features
Drawbacks	• Incapable to retain the temporal information of moving crowd
Description	Increasing world wide population is leading to dense crowd gathering at public places. Due to mass gathering at large scale, crowd related disaster has been frequently occurred. In order to prevent crowd calamities, automated crowd scene analysis has been a topic of great interest. Density is the status of crowd which is essential to classify in visual surveillance system primarily for security aspects. Most of the existing techniques work on detection and tracking of individuals. Due to fewer pixels per target, multiple occlusion and perspective effects etc., detection and tracking of individuals is a complex task in dense crowd scenarios. This paper presents a novel strategy for large scale crowd density classification powered by dynamic texture analysis. This approach consists of an interest points detection followed by spatio-temporal feature extraction. A rotation invariant spatio-temporal local binary (RIST-LBP) pattern is proposed to extract dynamic texture of the moving crowd. Further, a multi-class support vector regression is adopted for density classification. We also include a tracking step which tracks the selected interest points over the video frames for crowd flow estimation. We validate our proposed approach on three different datasets such as PETS, UCF and CUHK which vary in density ranging from low to very dense. The performance of our proposed

	approach is compared with most commonly used pixel based statistics. Our approach has the advantage of low computational complexity with high efficiency in real world applications of video surveillance.
Literature Survey 4	
Title	Enhanced People Counting System based Head-Shoulder Detection in Dense Crowd Scenario
Authors	Mohammed Abul Hassan, Indratno Pardiansyah, Aamir Saeed Malik, Ibrahima Faye, Waqas Rasheed
Published Year	October 2016
Efficiency	• Detect people against the false positive and false negative issue • Give more precise on the estimation
Drawbacks	• Some occlusion still occurred in a dense crowded situation • Too many overlapping people in the sequence
Description	Counting precisely the number of people in a crowd is one of the most attractive issues for video analytics application. In this paper, an integrated method using Histogram of Oriented Gradient (HOG) and Completed Local Binary Pattern (CLBP) is proposed to detect a head-shoulder region of people within image or video sequence. Head-shoulder region is used as features to detect people against the false positive and false negative issue. HOG and CLBP are used to extract the edge contour and texture features of head-shoulder region, respectively. The two features are fused together to generate a combined feature vector. Support Vector Machine (SVM) is used to execute classification of the fusion features to classify people from a mixture of objects. The results show that the detection rate of the proposed method HOG-CLBP, on Recall value and Accuracy, achieves better performance compared to the current method for dense crowd scenario.

Chapter 3: System And Methodology

3.1 EXISTING SYSTEM:

Existing system follow a Bid generative adversarial networks framework as the basis of our method, which contains the encoder, generator (decoder) and discriminator. The structure can encode the selected image into a specific distribution, which is the necessary condition to interpolate two target images. The core framework of BiGAN indicates that the encoder and generator (decoder) will be trained separately.

To train a general deep learning model to explore dynamic patterns of density maps, it is important to create available datasets. The property of samples in training data should contain different sizes, shapes, and time intervals, and they should also provide smooth and continuous dynamic information. As a visualization system, we want to balance the conciseness and computational capabilities and to introduce as few interactions as possible. The dynamic change may be difficult to be identified, particularly when there are slight changes and noises. We enhance the visual effect to improve our system by using blending, field representation, and sampling. The next parts introduce how to apply these technologies to implement a general framework

3.1.1 DISADVANTAGE:

- Does not work well with high dimensionality
- Does not work well with large dataset
- If the points on the boundaries are not informative (e.g., due to noise), it will do not do well.
- Existing system is Opportunistic and uncontrollable
- Difficulties to obtain better performance
- Inaccurate estimations of the missing pixels

3.2 PROPOSED SYSTEM:

Density Map estimation method to estimate the density map, which takes advantage of the underlying local geometrical similarity between image patches and their density map patches. Thus, the estimated density map can better embody the spatial distribution information of objects and it's more beneficial to further object localization.

Density map gives the object locations, but it lacks the detailed information of objects from image domain, especially the object boundary which is significant for object detection. Inspired by the human visual attention mechanism, saliency map estimation which provides object boundaries is introduced to estimate the accurate bounding box with the support of pre-localization results.

The process of bounding box estimation is realized by superpixel level, which refers to perceptually meaningful patches formed by adjacent pixels. Compact and highly uniform superpixels that respect image boundaries generated by the simple linear iterative clustering

3.2.1 ADVANTAGES:

- Captures the spatial features from an image
- It is one of the most accurate learning algorithms available. For many data sets, it produces a highly accurate classifier.
- Performs better when the number of dimensions is higher than the number of available samples
- Flexibility in the choice of the threshold form

3.3 SYSTEM REQUIREMENTS:

3.3.1 HARDWARE REQUIREMENTS:

Disk Space 32 GB or more, 10 GB

Processor 1.4 GHz 64 bit

Ram 8GB

Display (800 × 600) Capable video adapter and monitor

3.3.2 SOFTWARE REQUIREMENTS:

Windows 8 or above

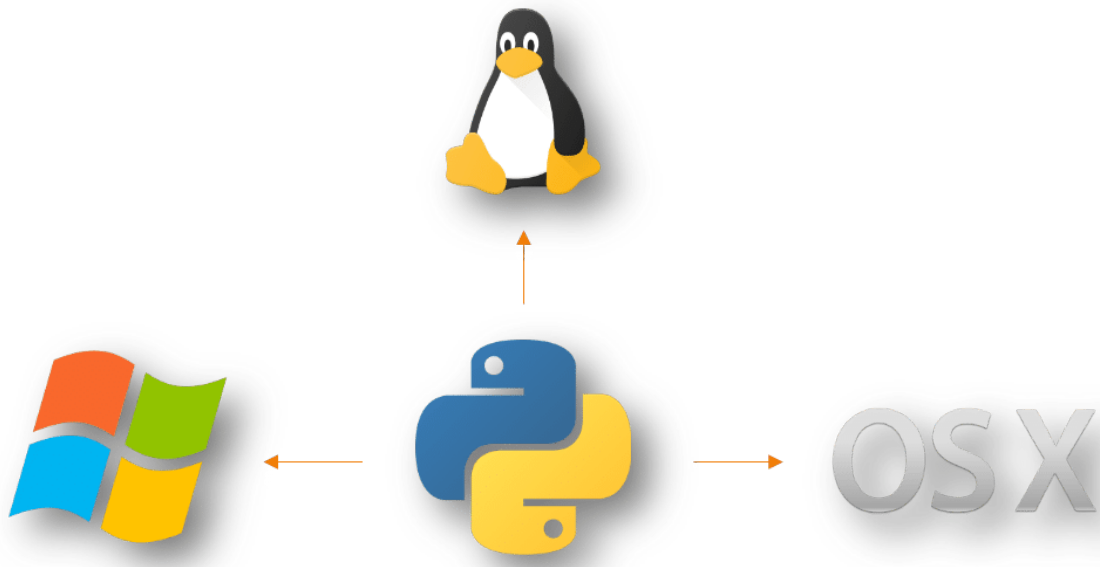
Python 3.7.7 or above

Open cv environment

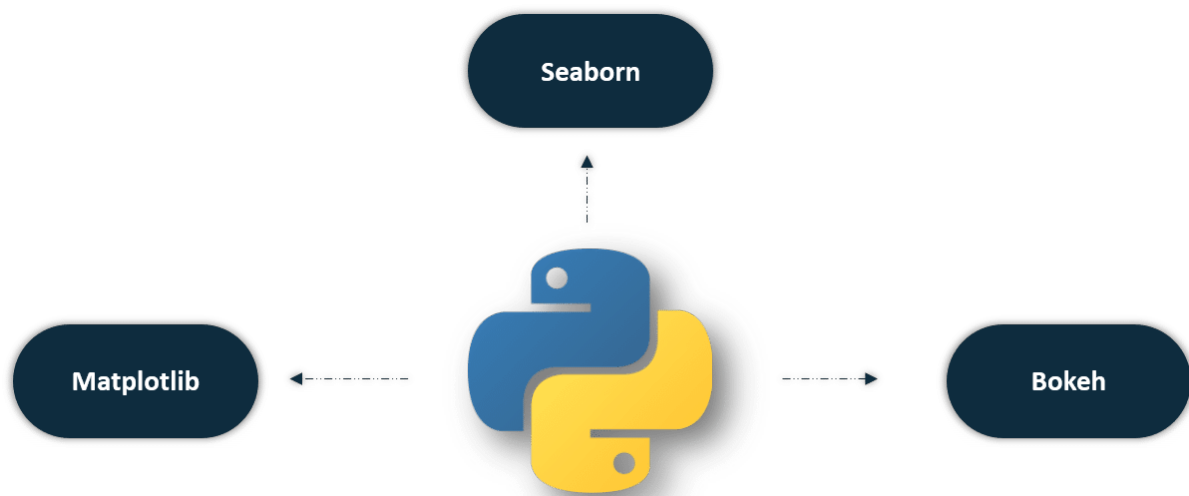
Anaconda prompt

3.4 SOFTWARE DESCRIPTION

Python is a free, open-source programming language. Therefore, all you have to do is install Python once, and you can start working with it. Not to mention that you can contribute your own code to the community. Python is also a cross-platform compatible language. So, what does this mean? Well, you can install and run Python on several operating systems. Whether you have a Windows, Mac or Linux, you can rest assured that Python will work on all these operating systems.



Python is also a great visualization tool. It provides libraries such as Matplotlib, seaborn and bokeh to create stunning visualizations.



In addition, Python is the most popular language for machine learning and deep learning. As a matter of fact, today, all top organizations are investing in Python to implement machine learning in the back-end.

Python Line Structure

Python coding style comprises physical lines as well as logical lines or statements. A physical line in a Python program is a sequence of characters, and the end of the line terminates the line sequence as opposed to some other languages, such as C and C++ where a semi-colon is used to mark the end of the statement. A logical line, on the other hand, is composed of one or more physical lines. The use of a semi-colon is not prohibited in Python, although it's not mandatory. The NEWLINE token denotes the end of the logical line. A logical line that only contains spaces, comments, or tabs are called blank lines and they are ignored by the interpreter.

As we saw that in Python, a new line simply means that a new statement has started. Although, Python does provide a way to split a statement into a multiline statement or to join multiple statements into one logical line. This can be helpful to increase the readability of the statement. Following are the two ways to split a line into two or more lines:

Explicit Line Joining

In explicit line joining, we use a backward slash to split a statement into a multiline statement.

Implicit Line Joining

Statements that reside inside [], {}, or () parentheses can be broken down into two or more physical lines without using a back slash.

Multiple Statements on a Single Line

In Python, it is possible to club multiple statements in the same line using a semi-colon; however, most programmers do not consider this to be a good practice as it reduces the readability of the code.

Whitespaces and Indentation

Unlike most of the programming languages, Python uses indentation to mark a block of code. According to Python coding style guideline or PEP8, we should keep an indent size of four.

Most of the programming languages provide indentation for better code formatting and don't enforce to have it. But in Python it is mandatory. This is why indentation is so crucial in Python.

Comments in any programming language are used to increase the readability of the code. Similarly, in Python, when the program starts getting complicated, one of the best ways to maintain the readability of the code is to use Python comments. It is considered a good practice to include documentations and notes in the python syntax since it makes the code way more readable and understandable to other programmers as well,

which comes in handy when multiple programmers are simultaneously working on the same project.

The code can only explain how it does something and not why it does that, but Python comments can do that. With Python comments, we can make documentations for various explanations in our code itself. Comments are nothing but tagged lines of codes which increase the readability of a code and make it self-explanatory. There are different ways of creating comments depending on the type of comment we want to include in our code. Following are different kinds of comments that can be included in our Python program:

1. Single Line Comments
2. Docstring Comments
3. Multiline Comments

Single line Python comments are marked with `#` character. These comments end at the end of the physical line, which means that all characters starting after the `#` character (and lasts till the end of the line) are part of the comment.

Python has the documentation strings (or docstrings) feature which is usually the first statement included in functions and modules. Rather than being ignored by the Python Interpreter like regular comments, docstrings can actually be accessed at the run time using the dot operator.

It gives programmers an easy way of adding quick notes with every Python module, function, class, and method. To use this feature, we use triple quotes in the beginning of the documentation string or comment and the closing triple quotes at the end of the documentation comment. Docstrings can be one-liners as well as multi-liners.

Unlike some programming languages that support multiline comments, such as C, Java, and more, there is no specific feature for multiline comments in Python. But that does not mean that it is totally impossible to make multiline comments in Python. There are two ways we can include comments that can span across multiple lines in our Python code.

Python Block Comments: We can use several single line comments for a whole block. This type of comment is usually created to explain the block of code that follows the Block comment. Python Block comment is the only way of writing a real comment that can span across multiple lines. It is supported and preferred by Python's PEP8 style guide since Block comments are ignored by Python interpreter or parser. However, nothing is stopping programmers from using the second 'non-real' way of writing multiline comments in Python which is explained below.

Using Docstrings: Docstrings are largely used as multiline comments in Python by many programmers since it is the closest thing to having a multiline comment feature in Python. While it is not wrong to use docstrings when we need to make multiline comments, it is important to keep in mind that there is a significant difference between docstrings and comments. Comments in Python are totally ignored by the Python

Interpreter, while docstrings, when used inside the Python function, can be accessed at the run time.

Python Data Types

One of the most crucial part of learning any programming language is to understand how data is stored and manipulated in that language. Users are often inclined toward Python because of its ease of use and the number of versatile features it provides. One of those features is dynamic typing.

In Python, unlike statically typed languages like C or Java, there is no need to specifically declare the data type of the variable. In dynamically typed languages such as Python, the interpreter itself predicts the data type of the Python Variable based on the type of value assigned to that variable.

Advantages of Python

- Universal Language Construct
- Support both High Level and Low Level Programming
- Language Interoperability
- Fastest Development life cycle therefore more productive coding environment
- Less memory used because a single container hold multiple data types and each type doesn't require its own function
- Learning Ease and open source development
- Speed and user-friendly data structure

- Extensive and extensible libraries.
- Simple & support IoT
- And many more

Based on the 3 languages that are described above, we decided to use Python as programming language for developing this e-Voting web based application. The important motives are

- Easy to learn, even non experienced programmers can use it. Ex: spacing and tabbing instead of extra syntax
- Interactive mode
- Large and comprehensive standard libraries
- Python programs resemble to that of pseudo-code. This makes it a basis and a must have for beginner programmers due to its extreme ease and difficulty when compared to C++ Java, Perl, and so forth

About Pandas

Pandas is a popular Python package for data science, and with good reason: it offers powerful, expressive and flexible data structures that make data manipulation and analysis easy, among many other things. The DataFrame is one of these structures.

Pandas is a high-level data manipulation tool developed by Wes McKinney. It is built on the Numpy package and its key data structure is called the DataFrame. DataFrames

allow you to store and manipulate tabular data in rows of observations and columns of variables.

Pandas is built on top of the NumPy package, meaning a lot of the structure of NumPy is used or replicated in Pandas. Data in pandas is often used to feed statistical analysis in SciPy, plotting functions from Matplotlib, and machine learning algorithms in Scikit-learn.

Jupyter Notebooks offer a good environment for using pandas to do data exploration and modeling, but pandas can also be used in text editors just as easily.

Jupyter Notebooks give us the ability to execute code in a particular cell as opposed to running the entire file. This saves a lot of time when working with large datasets and complex transformations. Notebooks also provide an easy way to visualize pandas' DataFrames and plots. As a matter of fact, this article was created entirely in a Jupyter Notebook.

There are two types of data structures in pandas: Series and DataFrames.

1. Series: a pandas Series is a one dimensional data structure (“a one dimensional ndarray”) that can store values — and for every value it holds a unique index, too.
2. DataFrame: a pandas DataFrame is a two (or more) dimensional data structure — basically a table with rows and columns. The columns have names and the rows have indexes.

Those who are familiar with R know the data frame as a way to store data in rectangular grids that can easily be overviewed. Each row of these grids corresponds to measurements or values of an instance, while each column is a vector containing data for a specific variable. This means that a data frame's rows do not need to contain, but can contain, the same type of values: they can be numeric, character, logical, etc.

Now, DataFrames in Python are very similar: they come with the Pandas library, and they are defined as two-dimensional labeled data structures with columns of potentially different types.

In general, you could say that the Pandas DataFrame consists of three main components: the data, the index, and the columns.

Firstly, the DataFrame can contain data that is:

- a Pandas DataFrame
- a Pandas Series: a one-dimensional labeled array capable of holding any data type with axis labels or index. An example of a Series object is one column from a DataFrame.
- a NumPy ndarray, which can be a record or structured
- a two-dimensional ndarray
- dictionaries of one-dimensional ndarray's, lists, dictionaries or Series.

Some of the key features of Python Pandas are as follows:

It provides DataFrame objects with default and customized indexing which is very fast and efficient.

- There are tools available for loading data of different file formats into in-memory data objects.
- It is easy to perform data alignment and integrated handling of missing data in Python Pandas.
- It is very simple to perform pivoting and reshaping of data sets in Pandas.
- It also provides indexing, label-based slicing, and sub-setting of large data sets.
- We can easily insert and delete columns from a data structure.
- Data aggregation and transformations can be done using group by.
- High-performance merging and joining of data can be done using Pandas.
- It also provides time series functionality.
- Inserting and deleting columns in data structures.
- Merging and joining data sets.
- Reshaping and pivoting data sets.

- Aligning data and dealing with missing data.
- Manipulating data using integrated indexing for DataFrame objects.
- Performing split-apply-combine on data sets using the group by engine.
- Manipulating high-dimensional data in a data structure with a lower dimension using hierarchical axis indexing.
- Subsetting, fancy indexing, and label-based slicing data sets that are large in size.
- Generating data range, converting frequency, date shifting, lagging, and other time-series functionality.
- Reading from files with CSV, XLSX, TXT, among other formats.
- Arranging data in an order ascending or descending.
- Filtering data around a condition.
- Analyzing time series.
- Iterating over a data set.

Numpy

Numpy is the core library for scientific computing in Python. It provides a high-performance multidimensional array object, and tools for working with these arrays.

If you are already familiar with MATLAB, you might find this tutorial useful to get started with Numpy.

A numpy array is a grid of values, all of the same type, and is indexed by a tuple of nonnegative integers. The number of dimensions is the rank of the array; the shape of an array is a tuple of integers giving the size of the array along each dimension.

NumPy is, just like SciPy, Scikit-Learn, Pandas, etc. one of the packages that you just can't miss when you're learning data science, mainly because this library provides you with an array data structure that holds some benefits over Python lists, such as: being more compact, faster access in reading and writing items, being more convenient and more efficient.

NumPy is a Python library that is the core library for scientific computing in Python. It contains a collection of tools and techniques that can be used to solve on a computer mathematical models of problems in Science and Engineering. One of these tools is a high-performance multidimensional array object that is a powerful data structure for efficient computation of arrays and matrices. To work with these arrays, there's a vast amount of high-level mathematical functions operate on these matrices and arrays.

an array is basically nothing but pointers. It's a combination of a memory address, a data type, a shape, and strides:

- The data pointer indicates the memory address of the first byte in the array,

- The data type or dtype pointer describes the kind of elements that are contained within the array,
- The shape indicates the shape of the array, and
- The strides are the number of bytes that should be skipped in memory to go to the next element. If your strides are (10,1), you need to proceed one byte to get to the next column and 10 bytes to locate the next row.

Or, in other words, an array contains information about the raw data, how to locate an element and how to interpret an element.

With NumPy, we work with multidimensional arrays. We'll dive into all of the possible types of multidimensional arrays later on, but for now, we'll focus on 2-dimensional arrays. A 2-dimensional array is also known as a matrix, and is something you should be familiar with. In fact, it's just a different way of thinking about a list of lists. A matrix has rows and columns. By specifying a row number and a column number, we're able to extract an element from a matrix.

We can create a NumPy array using the `numpy.array` function. If we pass in a list of lists, it will automatically create a NumPy array with the same number of rows and columns. Because we want all of the elements in the array to be float elements for easy computation, we'll leave off the header row, which contains strings. One of the limitations of NumPy is that all the elements in an array have to be of the same type, so if we include the header row, all the elements in the array will be read in as strings.

Because we want to be able to do computations like find the average quality of the wines, we need the elements to all be floats.

NumPy has several advantages over using core Python mathematical functions, a few of which are outlined here:

1. NumPy is extremely fast when compared to core Python thanks to its heavy use of C extensions.
2. Many advanced Python libraries, such as Scikit-Learn, Scipy, and Keras, make extensive use of the NumPy library. Therefore, if you plan to pursue a career in data science or machine learning, NumPy is a very good tool to master.

NumPy comes with a variety of built-in functionalities, which in core Python would take a fair bit of custom code.

Matplotlib

Plotting of data can be extensively made possible in an interactive way by Matplotlib, which is a plotting library that can be demonstrated in Python scripts. Plotting of graphs is a part of data visualization, and this property can be achieved by making use of Matplotlib.

Matplotlib makes use of many general-purpose GUI toolkits, such as wxPython, Tkinter, QT, etc., in order to provide object-oriented APIs for embedding plots into applications. John D. Hunter was the person who originally wrote Matplotlib, and its lead developer

was Michael Droettboom. One of the free and open-source Python library which is basically used for technical and scientific computing is Python SciPy. Matplotlib is widely used in SciPy as most scientific calculations require plotting of graphs and diagrams.

Matplotlib is a plotting library like GNUplot. The main advantage towards GNUplot is the fact that Matplotlib is a Python module. Due to the growing interest in python the popularity of matplotlib is continually rising as well.

Another reason for the attractiveness of Matplotlib lies in the fact that it is widely considered to be a perfect alternative to MATLAB, if it is used in combination with Numpy and Scipy. Whereas MATLAB is expensive and closed source, Matplotlib is free and open source code. It is also object-oriented and can be used in an object oriented way. Furthermore it can be used with general-purpose GUI toolkits like wxPython, Qt, and GTK+. There is also a procedural "pylab", which designed to closely resemble that of MATLAB. This can make it extremely easy for MATLAB users to migrate to matplotlib.

Matplotlib can be used to create publication quality figures in a variety of hardcopy formats and interactive environments across platforms.

Another characteristic of matplotlib is its steep learning curve, which means that users usually make rapid progress after having started. The official website has to say the following about this: "matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, errorcharts, scatterplots, etc, with just a few lines of code."

Python IDE

IDE – Integrated Development Environment. It is software suite. It contains different instruments for web application software engineers to compose and test applications.

Programming Web application software engineers utilize wide scope of devices habitually all through application source code creation, building and application testing. Programming Development apparatuses frequently incorporate word processors for source code creation, code libraries for reuse, compilers and test stages.

An Integrated Development Environment (IDE) is a product application that gives a programming domain to streamline creating and troubleshooting programming. Instead of playing out every one of the means required to make an executable program as disconnected individual assignments, it brings every one of the devices required into one application and workspace. Every one of the devices has a familiarity with the earth, and they cooperate to display a consistent improvement set for the designer.

An IDE typically has an editor and a compiler window. There, we can write our code, run it, and compile it. Hence, it becomes much easier to develop applications using an IDE.

The major reasons for using an IDE are given below:

- We can easily write our code in its text editor window.

- To store the resource files of a project, we have a project editor window.

- It makes debugging of the code easier.

- An IDE's autosuggest feature helps us in writing efficient code.

We can import other projects for integrating with the existing one.

Without an IDE, a product web application software engineer must choose, send, incorporate and deal with these instruments independently. An IDE brings huge numbers of those improvement related instruments all together system, application or administration. The incorporated toolset is intended to streamline programming advancement and can distinguish and limit coding missteps and grammatical errors.

- Komodo
- NetBeans
- PyCharm
- Python for Visual Studio.NET
- PyDev
- PyStudio
- LiClipse (Eclipse)

LiClipse is at the core for all Python based application development activities. It's been available for many years. LiClipse is a set of plugins to enhance Eclipse. It improves the overall Eclipse experience. It is best IDE tool to develop Python Applications.

Advantages of LiClipse

- Perfect Debugger
- Error checking features
- An enhanced ability to analyze the code for quality and security concerns
- Auto complete feature

Based on the various IDE that are described above, we decided to use LiClipse as IDE (Integrated Development Environment) for developing this e-Voting web based application.

Important Features That Make LiClipse our IDE of Choice

- Rich Reusable Libraries are available
- Model Driven Development
- Enterprise Python Tooling
- Powerful Debugging options

IDE is a program that is used to write applications (other programs) in a variety of programming languages. It allows you to complete many distinct program development steps in one place (program window). A program which allows you to do this is known as an integrated development environment, or IDE. The program is free and easy to import onto your home computer.

LiClipse is used for writing programs in Python. There are several steps that must be completed to set up LiClipse and further steps to start up a new Python project (set of Python components which work together to make up an application).

PyCharm is an Integrated Development Environment (IDE) that provides tools for writing efficient code. It supports various frameworks such as Django, Flask, and Pyramid. PyCharm is majorly used by developers for application development with Python language.

Let's suppose, we are making a project using Python programming language. We know that Python is extensively popular among the developer's community, as well as in industries, due to its following features:

- Simple to code: Python is a very simple language compared to other programming languages. Also, it is a developer-friendly language.
- Object-oriented language: Python is an object-oriented language that uses concepts of classes and objects.
- GUI programming support: We can create graphical user interfaces using Python Tkinter library. Besides, we can use modules such as PyQt5, PyQt4, and wxPython to create a GUI application.
- High-level language: Python is a high-level programming language. We do not need to take care of managing memory or the system architecture while writing Python codes.
- Portability: Python is a portable programming language, i.e., we can run Python codes on any platform. For example, we can run a code written for Windows in Linux, UNIX, and Mac, without changing it.
- Integrated language: Python is an integrated programming language. We can easily integrate C, C++, etc. with Python.

- Interpreted language: Python is an interpreted programming language as well. Python's source code is changed to bytecode. Due to this, it is directly executed without compiling.
- Large set of standard libraries: Python consists of a rich set of modules, packages, and libraries. It helps in various stages of application development.

"notebook" or "notebook documents" denote documents that contain both code and rich text elements, such as figures, links, equations, ... Because of the mix of code and text elements, these documents are the ideal place to bring together an analysis description, and its results, as well as, they can be executed perform the data analysis in real time.

The Jupyter Notebook App produces these documents.

As a server-client application, the Jupyter Notebook App allows you to edit and run your notebooks via a web browser. The application can be executed on a PC without Internet access, or it can be installed on a remote server, where you can access it through the Internet.

Its two main components are the kernels and a dashboard.

A kernel is a program that runs and introspects the user's code. The Jupyter Notebook App has a kernel for Python code, but there are also kernels available for other programming languages.

The dashboard of the application not only shows you the notebook documents that you have made and can reopen but can also be used to manage the kernels: you can which ones are running and shut them down if necessary.

CHAPTER 4 : SOFTWARE DEVELOPMENT METHODOLOGY

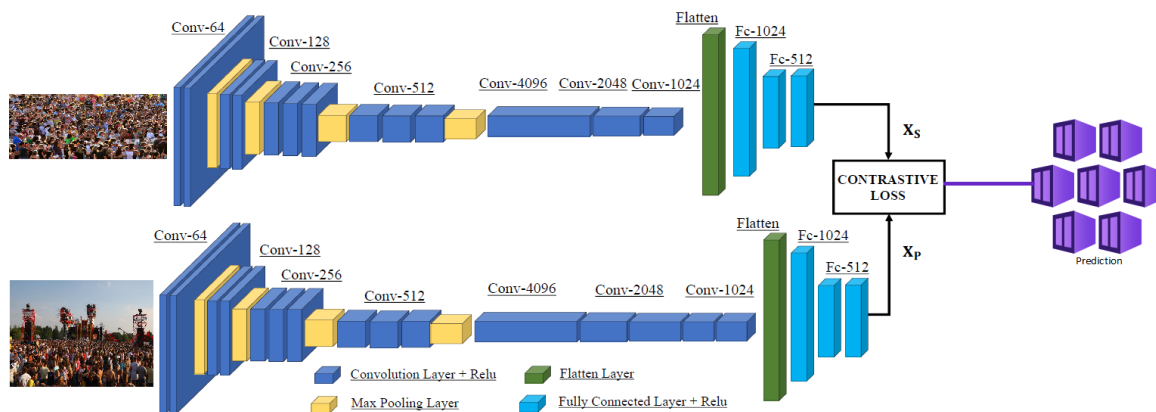
4.1 SYSTEM DESIGN:

System Design involves identification of classes their relationship as well as their collaboration. In objector, classes are divided into entity y classes and control classes. The Computer Aided Software Engineering (CASE) tools that are available commercially do not provide any assistance in this transition. CASE tools take advantage of Meta modelling that is helpful only after the construction of the class diagram. In the FUSION method some object-oriented approach likes Object Modelling Technique (OMT), Classes, and Responsibilities. Collaborators (CRC), etc, are used. Objector used the term” agents” to represent some of the hardware and software system. In Fusion method, there is no requirement phase, where a user will supply the initial requirement document. Any software project is worked out by both the analyst and the designer. The analyst creates the user case diagram. The designer creates the class diagram. But the designer can do this only after the analyst creates the use case diagram. Once the design is over, it is essential to decide which software is suitable for the application

4.2 UML DIAGRAM

UML is a standard language for specifying, visualizing, and documenting of software systems and created by Object Management Group (OMG) in 1997. There are three important type of UML modelling are Structural model, Behavioural model, and Architecture model . To model a system the most important aspect is to capture the dynamic behaviour which has some internal or external factors for making the interaction. These internal or external agents are known as actors. It consists of actors, use cases and their relationships. In this fig we represent the Use Case diagram for our project.

4.3 ARCHITECTURE:



4.4 MODULE DESCRIPTION:

4.4.1 EXPLORATORY DATA ANALYSIS(EDA):

Exploratory Data Analysis (EDA) is the first step in your data analysis process. Here, you make sense of the data you have and then figure out what questions you want to ask and how to frame them, as well as how best to manipulate your available data sources to get the answers you need. You do this by taking a broad look at patterns, trends, outliers, unexpected results and so on in your existing data, using visual and quantitative methods to get a sense of the story this tells.

Exploratory Data Analysis is valuable to data science projects since it allows to get closer to the certainty that the future results will be valid, correctly interpreted, and applicable to the desired business contexts. Such level of certainty can be achieved only after raw data is validated and checked for anomalies, ensuring that the data set was collected without errors. EDA also helps to find insights that were not evident or worth investigating to business stakeholders and data scientists but can be very informative about a particular business.

EDA is performed in order to define and refine the selection of feature variables that will be used for machine learning. Once data scientists become familiar with the data set, they often have to return to feature engineering step, since the initial features may turn out not to be serving their intended purpose. Once the EDA stage is complete, data scientists get a firm feature set they need for supervised and unsupervised machine learning.

4.4.2 PRE PROCESSING:

Sometimes you may find some data are missing in the dataset. We need to be equipped to handle the problem when we come across them. Obviously you could remove the entire line of data but what if you are unknowingly removing crucial information? Of course we would not want to do that. One of the most common idea to handle the problem is to take a mean of all the values of the same column and have it to replace the missing data.

The library that we are going to use for the task is called Scikit Learn preprocessing. It contains a class called Imputer which will help us take care of the missing data.

Sometimes our data is in qualitative form, that is we have texts as our data. We can find categories in text form. Now it gets complicated for machines to understand texts and process them, rather than numbers, since the models are based on mathematical equations and calculations. Therefore, we have to encode the categorical data.

Now we need to split our dataset into two sets — a Training set and a Test set. We will train our machine learning models on our training set, i.e our machine learning models will try to understand any correlations in our training set and then we will test the models on our test set to check how accurately it can predict. A general rule of the thumb is to allocate 80% of the dataset to training set and the remaining 20% to test set. For this task, we will import `test_train_split` from `model_selection` library of scikit

4.4.3 FEATURE ENGINEERING:

Filter methods are generally used as a preprocessing step. The selection of features is independent of any machine learning algorithms. Instead, features are selected on the basis of their scores in various statistical tests for their correlation with the outcome variable. The correlation is a subjective term here. For basic guidance, you can refer to the following table for defining correlation co-efficients.

Pearson's Correlation: It is used as a measure for quantifying linear dependence between two continuous variables X and Y. Its value varies from -1 to +1.

LDA: Linear discriminant analysis is used to find a linear combination of features that characterizes or separates two or more classes (or levels) of a categorical variable.

ANOVA: ANOVA stands for Analysis of variance. It is similar to LDA except for the fact that it is operated using one or more categorical independent features and one continuous dependent feature. It provides a statistical test of whether the means of several groups are equal or not.

Chi-Square: It is a statistical test applied to the groups of categorical features to evaluate the likelihood of correlation or association between them using their frequency distribution

4.4.4 PREDICTION:

Once training is complete, it's time to see if the model is any good, using Evaluation. This is where that dataset that we set aside earlier comes into play. Evaluation allows us to test our model against data that has never been used for training. This metric

allows us to see how the model might perform against data that it has not yet seen. This is meant to be representative of how the model might perform in the real world.

A good rule of thumb I use for a training-evaluation split somewhere on the order of 80/20 or 70/30. Much of this depends on the size of the original source dataset. If you have a lot of data, perhaps you don't need as big of a fraction for the evaluation dataset.

Once you've done evaluation, it's possible that you want to see if you can further improve your training in any way. We can do this by tuning our parameters. There were a few parameters we implicitly assumed when we did our training, and now is a good time to go back and test those assumptions and try other values.

A tree has many analogies in real life, and turns out that it has influenced a wide area of machine learning, covering both classification and regression. In decision analysis, a decision tree can be used to visually and explicitly represent decisions and decision making. As the name goes, it uses a tree-like model of decisions. A decision tree is drawn upside down with its root at the top. In the image on the left, the bold text in black represents a condition/internal node, based on which the tree splits into branches/edges. The end of the branch that doesn't split anymore is the decision/leaf, in this case, whether the passenger died or survived, represented as red and green text respectively.

Although, a real dataset will have a lot more features and this will just be a branch in a much bigger tree, but you can't ignore the simplicity of this algorithm. The feature importance is clear and relations can be viewed easily. This methodology is more

commonly known as learning decision tree from data and above tree is called Classification tree as the target is to classify passenger as survived or died. Regression trees are represented in the same manner, just they predict continuous values like price of a house. In general, Decision Tree algorithms are referred to as CART or Classification and Regression Trees.

CHAPTER 5 : RESULTS AND DISCUSSION

Our project detects the objects in images by forming there density maps.The dataset images are data cleaned by images cropping and other modules and datasets are pre-processed the pre processed dataset is gone through many mathematical implementations and module is trained by that. Once the module is formed it is applied to all the images in datasets and output of every image is given in .H5 format where all data of image is given as arrays . then all the images MAE and MSE is calculated. This helps to do many other implementations

CHAPTER 6: CONCLUSION AND FUTURE EHANCEMETS

6.1 CONCLUSION:

In this paper , we analyzed the images and and created a module which helps is extracting any images density maps.which helps in detecting objects basis of destiny mapping every image density values are given in array format for each and every pixels and this data is stored in form of .h5 format . we presented this module which is a better implementation than generative maps.

6.2 FUTURE ENHANCEMENT:

With the help of density maps produced in our module we can create many different applications which can help in identify changes in any two similar images which can be a key implementation of Weather detection ,deforestation like application

REFERENCES

1. Bansal and K. Venkatesh, "People counting in high density crowds from still images," arXiv preprint arXiv:1507.08445, 2015.
2. V. Bastani, D. Campo, L. Marcenaro, and C. Regazzoni, "Online pedestrian group walking event detection using spectral analysis of motion similarity graph," in Advanced Video and Signal Based Surveillance (AVSS), 2015 12th IEEE International Conference on. IEEE, 2015, pp. 1–5.
3. O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein et al., "Imagenet large scale visual recognition challenge," IJCV, vol. 115, no. 3, pp. 211–252, 2015.
4. K. Tota and H. Idrees, "Counting in dense crowds using deep features," 2015.
5. Y. Hu, H. Chang, F. Nian, Y. Wang, and T. Li, "Dense crowd counting from still images with convolutional neural networks," Journal of Visual Communication and Image Representation, vol. 38, pp. 530–539, 2016.
6. Y. Zhang, D. Zhou, S. Chen, S. Gao, and Y. Ma, "Single-image crowd counting via multi-column convolutional neural network," in CVPR, 2016, pp. 589–597.
7. M. Marsden, K. McGuinness, S. Little, and N. E. O'Connor, "Fully convolutional crowd counting on highly congested scenes," arXiv preprint arXiv:1612.00220, 2016.
8. L. Boominathan, S. S. Kruthiventi, and R. V. Babu, "Crowdnet: a deep convolutional network for dense crowd counting," in Proceedings of the 2016 ACM on Multimedia Conference. ACM, 2016, pp. 640–644..
9. J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in CVPR, 2015, pp. 3431–3440.

10. W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in ECCV. Springer, 2016, pp. 21–37.
11. J. Redmon and A. Farhadi, "Yolo9000: better, faster, stronger," arXiv preprint arXiv:1612.08242, 2016.
12. S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards realtime object detection with region proposal networks," in NIPS, 2015, pp. 91–99.
13. N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting." JMLR, vol. 15, no. 1, pp. 1929–1958, 2014.
14. M. Oquab, L. Bottou, I. Laptev, and J. Sivic, "Learning and transferring mid-level image representations using convolutional neural networks," in CVPR, 2014, pp. 1717–1724
15. Wang, H. Zhang, L. Yang, S. Liu, and X. Cao, "Deep people counting in extremely dense crowds," in Proceedings of the 23rd ACM international conference on Multimedia. ACM, 2015, pp. 1299–1302.

APPENDIX

SOURCE CODE:

```
import os

import numpy as np

import pandas as pd

import glob

from tqdm import tqdm

import glob

from sklearn.model_selection import train_test_split

import json

from tensorflow.keras import backend as K

from tensorflow.keras.models import load_model, model_from_json

from tensorflow.keras.applications.vgg16 import VGG16, preprocess_input

from tensorflow.keras.losses import mean_squared_error

from tensorflow.keras import backend as K

from tensorflow.keras.models import Model

from tensorflow.keras.layers import Conv2D, UpSampling2D, BatchNormalization,
Activation

from tensorflow.keras.initializers import RandomNormal
```

```

from tensorflow.keras.optimizers import SGD, Adam

from tensorflow.keras.utils import Sequence

import h5py

import PIL.Image as Image

import PIL.Image as Image

from matplotlib import pyplot as plt

from matplotlib import cm as CM

DATA_PATH = "dataset/"

MODEL_PATH = "densitymap1.model"

MODEL_JSON_PATH = "densitymap1.json"


def create_training_image_list(data_path):

    DATA_PATH = data_path

    image_path_list = glob.glob(os.path.join(DATA_PATH, "images", "*.jpg"))

    return image_path_list


def get_train_val_list(data_path):

    DATA_PATH = data_path

    image_path_list = glob.glob(os.path.join(DATA_PATH, "images", "*.jpg"))

    train, val = train_test_split(image_path_list, test_size=0.1)

```

```
print("train size ", len(train))
```

```
print("val size ", len(val))
```

```
return train, val
```

```
def get_test_list(data_path):
```

```
    image_path_list = glob.glob(os.path.join(data_path, "images", "*.jpg"))
```

```
    print("test size ", len(image_path_list))
```

```
    return image_path_list
```

```
def create_density_path_list(image_path_list):
```

```
    gt_paths = []
```

```
    for img_path in image_path_list:
```

```
        gt_path = img_path.replace('.jpg', '.h5').replace('images', 'ground-truth-h5')
```

```
        gt_paths.append(gt_path)
```

```
    return gt_paths
```

```
def mae_metric(y_true, y_pred):
```

```
    return K.abs(K.sum(y_true) - K.sum(y_pred))
```

```
def euclidean_distance_loss(y_true, y_pred):  
  
    return K.sqrt(K.sum(K.sum(K.sum(K.square(y_pred - y_true), axis=3), axis=2),  
axis=1))
```

```
def build_model():
```

```
    sgd = SGD(lr=1e-7, decay=5*1e-4, momentum=0.95)
```

```
    adam = Adam()
```

```
    vgg16_model = VGG16(weights='imagenet', include_top=False)
```

```
    x = vgg16_model.get_layer('block4_conv3').output
```

```
    x = BatchNormalization()(x)
```

```
    x = UpSampling2D(size=(8, 8))(x)
```

```
    x = Conv2D(filters=512, kernel_size=(3, 3), dilation_rate=2, padding='same',  
use_bias=False, kernel_initializer=RandomNormal(stddev=0.01))(x)
```

```
    x = BatchNormalization()(x)
```

```
    x = Activation('relu')(x)
```

```
    x = Conv2D(filters=512, kernel_size=(3, 3), dilation_rate=2, padding='same',  
use_bias=False, kernel_initializer=RandomNormal(stddev=0.01))(x)
```

```
    x = BatchNormalization()(x)
```

```
    x = Activation('relu')(x)
```

```
    x = Conv2D(filters=512, kernel_size=(3, 3), dilation_rate=2, padding='same',  
use_bias=False, kernel_initializer=RandomNormal(stddev=0.01))(x)
```

```

x = BatchNormalization()(x)

x = Activation('relu')(x)

x = Conv2D(filters=256, kernel_size=(3, 3), dilation_rate=2, padding='same',
use_bias=False, kernel_initializer=RandomNormal(stddev=0.01))(x)

x = BatchNormalization()(x)

x = Activation('relu')(x)

x = Conv2D(filters=128, kernel_size=(3, 3), dilation_rate=2, padding='same',
use_bias=False, kernel_initializer=RandomNormal(stddev=0.01))(x)

x = BatchNormalization()(x)

x = Activation('relu')(x)

x = Conv2D(filters=64, kernel_size=(3, 3), dilation_rate=2, padding='same',
use_bias=False, kernel_initializer=RandomNormal(stddev=0.01))(x)

x = BatchNormalization()(x)

x = Activation('relu')(x)

x = Conv2D(filters=1, kernel_size=(1, 1), dilation_rate=1, padding='same',
use_bias=True, kernel_initializer=RandomNormal(stddev=0.01))(x)

x = Activation('relu')(x)

model = Model(inputs=vgg16_model.input, outputs=x)

model.compile(optimizer=sgd,

               loss=mean_squared_error, metrics=[mae_metric])

return model

```

```
def save_model(model , weight_path , json_path):
```

```
    model.save_weights(weight_path)
```

```
    model_json = model.to_json()
```

```
    with open(json_path, "w") as json_file:
```

```
        json_file.write(model_json)
```

```
def load_json_model(weight_path, json_path):
```

```
    json_file = open(json_path, 'r')
```

```
    loaded_model_json = json_file.read()
```

```
    json_file.close()
```

```
    loaded_model = model_from_json(loaded_model_json)
```

```
    loaded_model.load_weights(weight_path)
```

```
    return loaded_model
```

```
model = build_model()
```

```
x = np.random.rand(1, 224, 224, 3)
```

```
pred = model.predict(x)
```

```
print(pred.shape)
```

```
(1, 224, 224, 1)
```

```
def load_density(file_path):
```

```
    gt_file = h5py.File(file_path, 'r')
```

```
    groundtruth = np.asarray(gt_file['density'])
```

```
    return groundtruth
```

```
def random_crop(img, density_map, random_crop_size):
```

```
    assert img.shape[2] == 3
```

```
    height, width = img.shape[0], img.shape[1]
```

```
    dy, dx = random_crop_size
```

```
    x = np.random.randint(0, width - dx + 1)
```

```
    y = np.random.randint(0, height - dy + 1)
```

```
    return img[y:(y+dy), x:(x+dx), :], density_map[y:(y+dy), x:(x+dx), :]
```

```
class DatasetSequence(Sequence):
```

```
    def __init__(self, image_path_list, density_path_list, random_crop_size=None):
```

```
        self.image_path_list = image_path_list
```

```
        self.density_path_list = density_path_list
```

```
        self.random_crop_size = random_crop_size
```

```

self.batch_size = 1

def __len__(self):
    return len(self.image_path_list)

def __getitem__(self, idx):
    try:
        image_path = self.image_path_list[idx]
        density_path = self.density_path_list[idx]

        density = load_density(density_path)

        image = np.array(Image.open(image_path, "r").convert("RGB"))

        print(density.shape)

        density = np.expand_dims(density, axis=2)

        if self.random_crop_size is not None:
            image, density = random_crop(image, density, self.random_crop_size)

        im = image

        im = im/255.0

        im[:, :, 0] = (im[:, :, 0] - 0.485) / 0.229

```

```
im[:, :, 1] = (im[:, :, 1] - 0.456) / 0.224
```

```
im[:, :, 2] = (im[:, :, 2] - 0.406) / 0.225
```

```
image = im
```

```
image = np.expand_dims(image, axis=0)
```

```
density = np.expand_dims(density, axis=0)
```

```
return image, density
```

```
except Exception as ex:
```

```
    print(ex)
```

```
    pass
```

```
def get_non_preprocess(self, idx):
```

```
#     try:
```

```
    image_path = self.image_path_list[idx]
```

```
    density_path = self.density_path_list[idx]
```

```
    density = load_density(density_path)
```

```
    image = np.array(Image.open(image_path, "r").convert("RGB"))
```

```
    density = np.expand_dims(density, axis=2) # add channel dim
```

```

# density = np.expand_dims(density, axis=3) # add channel dim

image = np.expand_dims(image, axis=0) # add batch dim

density = np.expand_dims(density, axis=0) # add batch dim


return image, density

#     except:

#         pri

#         pass


def get_random_crop_image(self, idx):

#     try:

image_path = self.image_path_list[idx]

density_path = self.density_path_list[idx]


density = load_density(density_path)

image = np.array(Image.open(image_path, "r").convert("RGB"))

density = np.expand_dims(density, axis=3)


if self.random_crop_size is not None:

```

```
image, density = random_crop(image, density, self.random_crop_size)
```

```
im = image
```

```
im = im/255.0
```

```
im[:, :, 0] = (im[:, :, 0] - 0.485) / 0.229
```

```
im[:, :, 1] = (im[:, :, 1] - 0.456) / 0.224
```

```
im[:, :, 2] = (im[:, :, 2] - 0.406) / 0.225
```

```
image = im
```

```
image = np.expand_dims(image, axis=0)
```

```
density = np.expand_dims(density, axis=0)
```

```
return image, density
```

```
def get_random_crop_image_batch(self, idx, batch_size):
```

```
    image_batch = []
```

```
    density_batch = []
```

```
    for i in range(batch_size):
```

```
        image, density = self.get_random_crop_image(idx)
```

```
        image_batch.append(image)
```

```
density_batch.append(density)
```

```
images = np.concatenate(image_batch, axis=0)
```

```
densities = np.concatenate(density_batch, axis=0)
```

```
return images, densities
```

```
def get_all(self, crop_per_img):
```

```
    image_list = []
```

```
    density_list = []
```

```
    for i in tqdm(range(len(self.image_path_list))):
```

```
        try:
```

```
            image, density = self.get_random_crop_image_batch(i, crop_per_img)
```

```
            image_list.append(image)
```

```
            density_list.append(density)
```

```
        except:
```

```
            print("exception at image ", i)
```

```
    image_mat = np.concatenate(image_list, axis = 0)
```

```
    density_mat = np.concatenate(density_list, axis = 0)
```

```
    return image_mat, density_mat
```

```

image_list = glob.glob(DATA_PATH+"*.jpg")

density_list = list(map(lambda s: s.replace(".jpg", ".h5"), image_list))

len(image_list)

len(density_list)

train_dataset = DatasetSequence(image_list, density_list, random_crop_size=(448,
448))

img, density = train_dataset[0]

print(img.shape)

print(density.shape)

for i in train_dataset:

    print(i)

    model = build_model()

    epoch = 1

    print("total epoch ", epoch)

    model.fit_generator(train_dataset, shuffle=True, epochs = epoch)

    save_model(model, MODEL_PATH, MODEL_JSON_PATH)

import sys

sys.stdout.flush()

```

```

loaded_model = load_json_model(MODEL_PATH, MODEL_JSON_PATH)

def save_density_map(density_map, name):

    plt.figure(dpi=600)

    plt.axis('off')

    plt.margins(0, 0)

    plt.imshow(density_map, cmap=CM.jet)

    plt.savefig(name, dpi=600, bbox_inches='tight', pad_inches=0)

img_train, density_train = train_dataset.get_non_preprocess(0)

pil_img = Image.fromarray(img_train[0])

model = load_json_model(MODEL_PATH, MODEL_JSON_PATH)

print("label ", density_train.sum())

pred = model.predict(img_train)

print("predict ", np.squeeze(pred[0], axis=2).shape, np.squeeze(pred[0], axis=2).sum())


print("-----")

pil_img.save("train.png")

plt.figure(dpi=600)

plt.axis('off')

plt.margins(0,0)

plt.imshow(Image.open("train.png"))

```

```
save_density_map(np.squeeze(density_train[0], axis=2), "label.png")
```

```
np.squeeze(pred[0], axis=2)
```

```
save_density_map(np.squeeze(pred[0], axis=2), "predict.png")
```

```
pred[0].shape
```

```
np.savetxt("pred_np.txt", np.squeeze(density_train[0], axis=2))
```

```
val_dataset = DatasetSequence(image_list, density_list)
```

```
n = 0
```

```
mae = 0
```

```
mse = 0
```

```
for i in tqdm(range(len(val_dataset))):
```

```
    img, density = val_dataset[i]
```

```
    pred = model.predict(img)
```

```
    pred_values = pred.sum()
```

```
    truth = density.sum()
```

```
    mae = mae + abs(truth - pred_values)
```

```
    n += 1
```

```
    mse = mse + (truth - pred_values) * (truth - pred_values)
```

```
val_result_mae = mae / n
```

```

val_result_mse = np.sqrt(mse/n)

print('val mae ', val_result_mae)

print('val mse ', val_result_mse)

TEST_PATH = "test_data"

test_img_path = get_test_list(TEST_PATH)

print(test_img_path)

test_density_path = create_density_path_list(test_img_path)

test_dataset = DatasetSequence(test_img_path, test_density_path)

n = 0

mae = 0

mse = 0

for i in tqdm(range(len(test_dataset))):

    img, density = test_dataset[i]

    pred = model.predict(img)

    pred_values = pred.sum()

    truth = density.sum()

    mae = mae + abs(truth - pred_values)

    n += 1

    mse += (truth - pred_values) * (truth - pred_values)

mae = mae / n

```

```
mse = np.sqrt(mse / n)
```

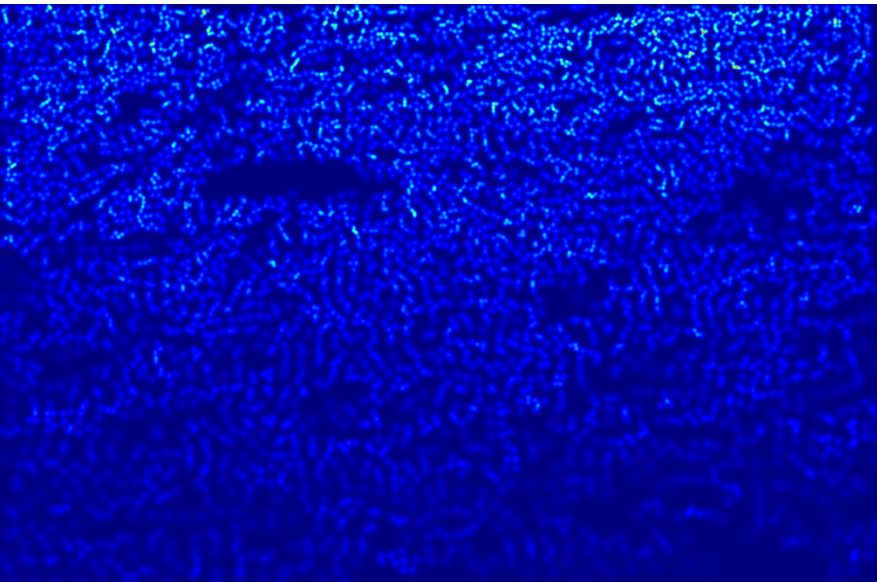
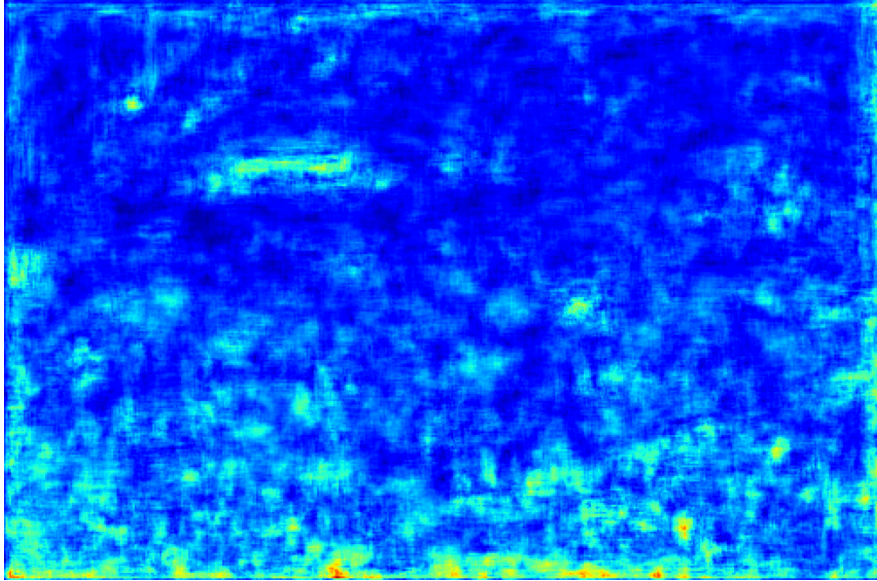
```
print(mae)
```

```
print(mse)
```

SNAPSHOTS

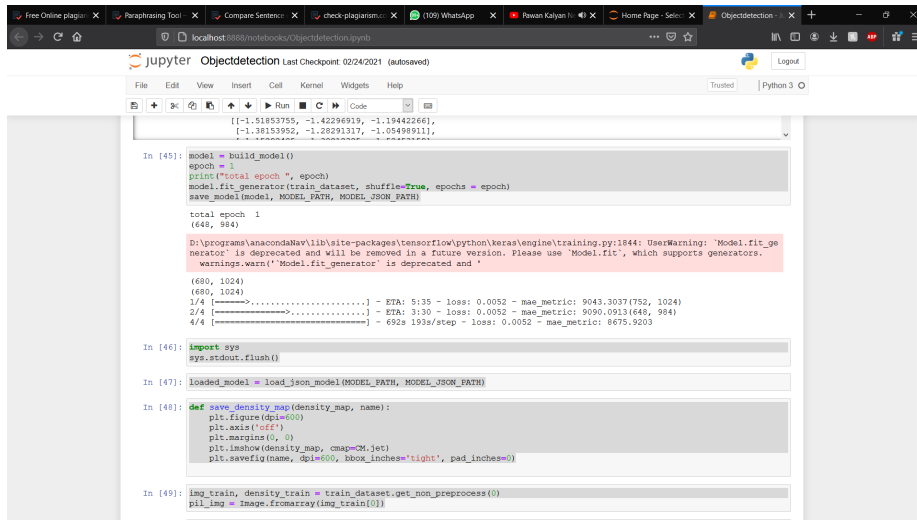
PREDICTED IMAGE

DENSITY MAP



DATASETS

TRAINING EPOCHS



The screenshot shows a Jupyter Notebook interface with the following code cells:

```
In [45]: model = build_model()
epoch = 1
print('total epoch ', epoch)
model.fit_generator(train_dataset, shuffle=True, epochs = epoch)
save_model(model, MODEL_PATH, MODEL_JSON_PATH)

total epoch 1
(648, 984)

D:\Programs\anaconda\envs\lib\site-packages\tensorflow\python\keras\engine\training.py:1844: UserWarning: 'Model.fit_generator' is deprecated and will be removed in a future version. Please use 'Model.fit', which supports generators.
warnings.warn('Model.fit_generator' is deprecated and

(680, 1024)
(680, 1024)
1/4 |-----> ETA: 5:35 - loss: 0.0052 - mae_metric: 9043.3037(752, 1024)
2/4 |-----> ETA: 3:30 - loss: 0.0052 - mae_metric: 9090.0913(648, 984)
4/4 |-----> ETA: 4:33 - loss: 0.0052 - mae_metric: 8675.9203

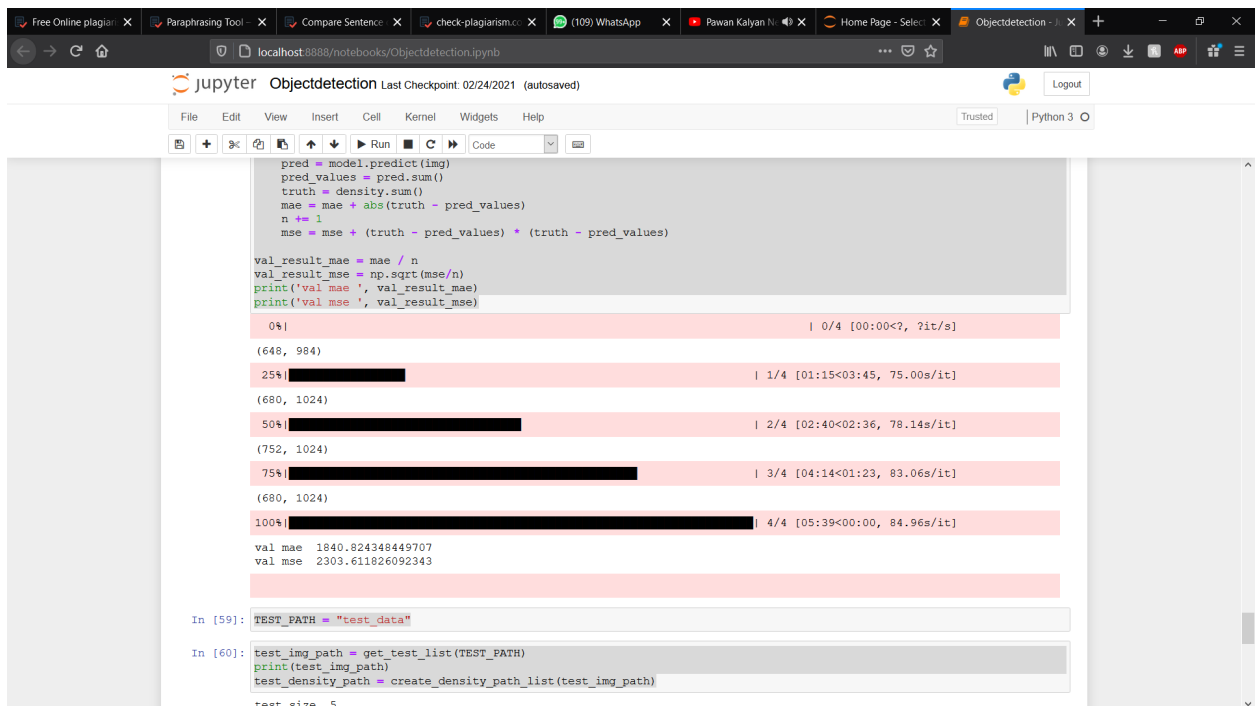
In [46]: import sys
sys.stdout.flush()

In [47]: loaded_model = load_json_model(MODEL_PATH, MODEL_JSON_PATH)

In [48]: def save_density_map(density_map, name):
plt.figure(dpi=600)
plt.axis('off')
plt.imshow(density_map, cmap=CM.jet)
plt.savefig(name, dpi=600, bbox_inches='tight', pad_inches=0)

In [49]: img_train, density_train = train_dataset.get_non_preprocess(0)
pil_img = Image.fromarray(img_train[0])
```

CALCULATING VALIDATED MAE&MSE



The screenshot shows a Jupyter Notebook interface with the following code cells:

```
pred = model.predict(img)
pred_values = pred.sum()
truth = density.sum()
mae = mae + abs(truth - pred_values)
n += 1
mse = mse + (truth - pred_values) * (truth - pred_values)

val_result_mae = mae / n
val_result_mse = np.sqrt(mse/n)
print('val mae ', val_result_mae)
print('val mse ', val_result_mse)

0%| | 0/4 [00:00<?, ?it/s]

(648, 984)
25%| | 1/4 [01:15<03:45, 75.00s/it]

(680, 1024)
50%| | 2/4 [02:40<02:36, 78.14s/it]

(752, 1024)
75%| | 3/4 [04:14<01:23, 83.06s/it]

(680, 1024)
100%| | 4/4 [05:39<00:00, 84.96s/it]

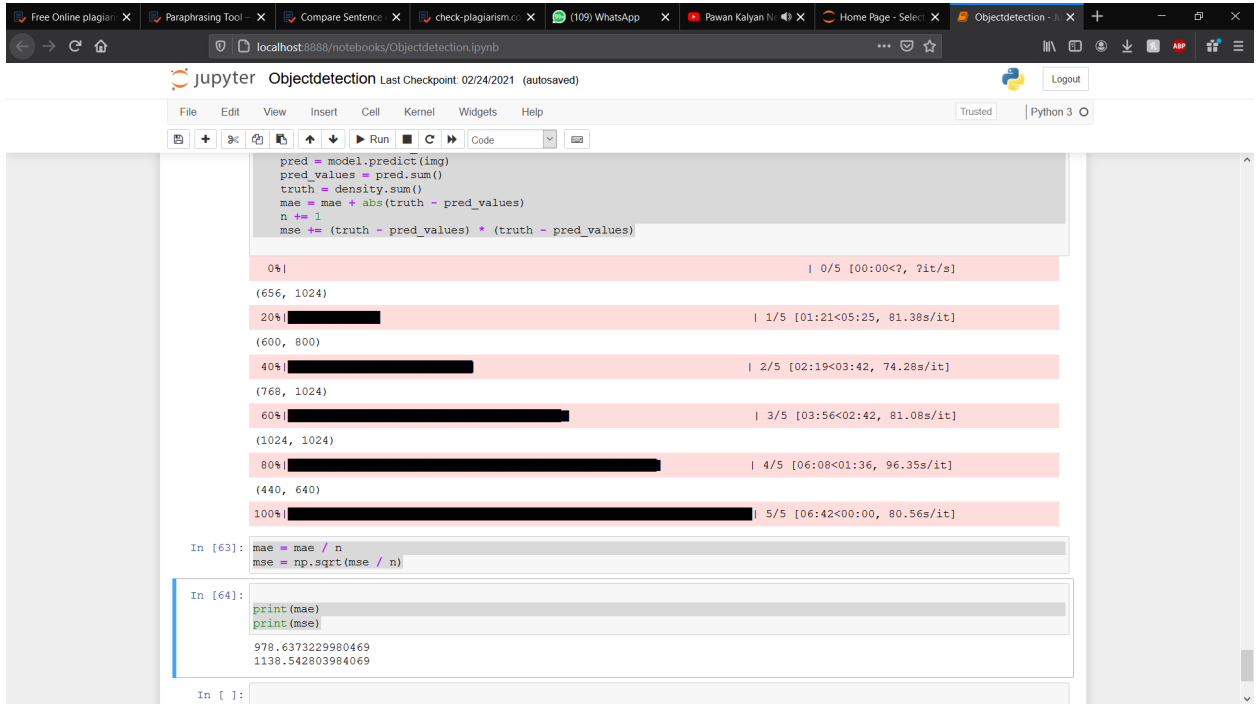
val mae 1840.824348449707
val mse 2303.611826092343

In [59]: TEST_PATH = "test_data"

In [60]: test_img_path = get_test_list(TEST_PATH)
print(test_img_path)
test_density_path = create_density_path_list(test_img_path)

test size 5
```

FINAL MAE & MSE



The screenshot shows a Jupyter Notebook titled "ObjectDetection" with a last checkpoint of 02/24/2021. The notebook is running on a local host at 8888. The code in the first cell calculates the Mean Absolute Error (MAE) and Mean Squared Error (MSE) for a model. The progress bar shows the training progress from 0% to 100% over 5 epochs. The final MAE and MSE values are displayed in the output of the second cell.

```
pred = model.predict(img)
pred_values = pred.sum()
truth = density.sum()
mae = mae + abs(truth - pred_values)
n += 1
mse += (truth - pred_values) * (truth - pred_values)
```

0% | 0/5 [00:00<?, ?it/s]

(656, 1024)

20% | 1/5 [01:21<05:25, 81.38s/it]

(600, 800)

40% | 2/5 [02:19<03:42, 74.28s/it]

(768, 1024)

60% | 3/5 [03:56<02:42, 81.08s/it]

(1024, 1024)

80% | 4/5 [06:08<01:36, 96.35s/it]

(440, 640)

100% | 5/5 [06:42<00:00, 80.56s/it]

```
In [63]: mae = mae / n
mse = np.sqrt(mse / n)
```

```
In [64]: print(mae)
print(mse)
```

978.6373229980469
1138.542803984069

In []: