

**ENHANCING VIDEO CROWD COUNTING WITH SPATIAL
TEMPORAL GRAPH NETWORKS AND YOLO-BASED
ARCHITECTURE FOR REAL-TIME APPLICATIONS**

A PROJECT REPORT

Submitted by

BALAJI S

421619104014

HARIHARASUDHAN A

421619104025

JAYARAMAN A

421619104030

In partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

MAILAM ENGINEERING COLLEGE

MAILAM



ANNA UNIVERSITY : CHENNAI 600 025

MAY 2023

BONAFIDE CERTIFICATE

Certified that this project report for the title **“ENHANCING VIDEO CROWD COUNTING WITH SPATIAL TEMPORAL GRAPH NETWORKS AND YOLO-BASED ARCHITECTURE FOR REAL-TIME APPLICATIONS”** is the bonafide work of **BALAJI S (421619104014), HARIHARASUDHAN A (421619104025), JAYARAMAN A (421619104030)** who carried out the project under my supervision.

SIGNATURE

Mr.S.PRASANNA,M.E.,

SUPERVISOR

Assistant Professor,

Computer Science and Engineering,

Mailam Engineering College,

Mailam-604304.

SIGNATURE

Dr.T.PRIYARADHIKADEVI,M.Tech.,Ph.D.,

HEAD OF THE DEPARTMENT

Professor and Head,

Computer Science and Engineering,

Mailam Engineering College,

Mailam-604304.

Submitted for the project and Viva-Voice Examination held on _____

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

I take this privilege to express a few words of gratitude and respect to all those who helped me in completion of this project.

Owing deeply to supreme, express my sincere thanks to our honorable chairman and managing director **Shri. M.DHANASEKARAN, M.A., D.Ag. D.F.T.**, Our beloved Vice Chairman **Shri.S.V.SUGUMARAN**, and Secretary of our college **Dr.K.NARAYANASAMY M.B.B.S.**, who entered his generosity to give the facilities to undergo the project work till its completion.

I am very much revered to our Director **Dr.S.SENTHIL,M.E,Ph.D.**, for his support and encouragement shown towards our academics profile. We tender our heartfelt gratitude to our beloved Principal of Our college **Dr.R.RAJAPPAN,M.Tech.,Ph.D.,MISTE** for his valuable and encouragement throughout this endeavor.

I extend my sincere thanks and gratitude to our Head of The Department. **Dr.T.PRIYARADHIKADEVI,M.Tech,Ph.D.**, for sharing her thoughts in doing the project and the encouragement shown towards the project work.

I am very much obliged to my project supervisor **Mr.S.PRASANNA,M.E.**, for his motivation, guidelines and continuous support for conducting the reviews at each and every level of work.

I also extend my sincere thanks to all the staff members and nonteaching staff of our department.

ABSTRACT

Video crowd counting is a challenging task in computer vision due to the complexity of crowd dynamics and the large variation in crowd density. In this paper, we propose a novel approach for video crowd counting using a Spatial-Temporal Graph Network (STGN) and You Only Look Once (YOLO) Version 7.

The proposed STGN leverages the spatio-temporal dependencies among frames in a video to generate a graph representation of the crowd motion. The STGN architecture consists of three main components: a graph convolutional network (GCN) to model the spatial dependencies, a temporal convolutional network (TCN) to model the temporal dependencies, and a graph attention mechanism to capture the importance of different nodes in the graph. The STGN takes as input the feature maps generated by YOLO Version 7 and generates a crowd count for each frame in the video.

To improve the accuracy of crowd counting, we use YOLO Version 7 to detect and track individuals in the video frames. YOLO Version 7 is a state-of-the-art object detection and tracking model that achieves high accuracy and real-time performance. By integrating YOLO Version 7 with STGN, we are able to accurately detect and track individuals in the crowd and generate a more accurate crowd count.

We evaluate the proposed method on several benchmark datasets, including ShanghaiTech Part A and Part B, UCF_CC_50, and WorldExpo'10, and compare the results with existing methods. The experimental results show that the proposed method achieves state-of-the-art performance in terms of crowd counting accuracy

and computational efficiency. Our method is also able to handle challenging scenarios such as occlusion, varying lighting conditions, and large-scale crowds.

சுருக்கம்

கூட்டத்தின் இயக்கவியலின் சிக்கலான தன்மை மற்றும் கூட்ட அடர்த்தியின் பெரிய மாறுபாட்டின் காரணமாக வீடியோ கூட்டத்தை கணக்கிடுவது கணினி பார்வையில் ஒரு சவாலான பணியாகும். இந்தத் தாளில், ஸ்பேஷியல்-டெம்போரல் கிராஃப் நெட்வொர்க் (STGN) மற்றும் யூ ஒன்லி லுக் ஒன் (YOLO) பதிப்பு 7ஐப் பயன்படுத்தி வீடியோ கூட்டத்தை எண்ணுவதற்கான புதிய அணுகுமுறையை நாங்கள் முன்மொழிகிறோம்.

முன்மொழியப்பட்ட STGN, கூட்டத்தின் இயக்கத்தின் வரைபடப் பிரதிநிதித்துவத்தை உருவாக்க, வீடியோவில் உள்ள பிரேம்களுக்கு இடையே உள்ள இடைவெளி-தற்காலிக சார்புகளை மேம்படுத்துகிறது. STGN கட்டிடக்கலை மூன்று முக்கிய கூறுகளைக் கொண்டுள்ளது: இடஞ்சார்ந்த சார்புகளை மாதிரியாக்க ஒரு வரைபட கன்வல்யூஷனல் நெட்வொர்க் (GCN), தற்காலிக சார்புகளை மாதிரியாக்க ஒரு டெம்போரல் கன்வல்யூஷனல் நெட்வொர்க் (TCN) மற்றும் வரைபடத்தில் உள்ள பல்வேறு முனைகளின் முக்கியத்துவத்தைப் படம்பிடிக்க ஒரு கிராஃப் கவனம் பொறிமுறை. . STGN ஆனது YOLO பதிப்பு 7 ஆல் உருவாக்கப்பட்ட அம்ச வரைபடங்களை உள்ளீடாக எடுத்துக்கொள்கிறது மற்றும் வீடியோவில் உள்ள ஒவ்வொரு சட்டத்திற்கும் கூட்ட எண்ணிக்கையை உருவாக்குகிறது.

கூட்ட எண்ணிக்கையின் துல்லியத்தை மேம்படுத்த, வீடியோ பிரேம்களில் தனிநபர்களைக் கண்டறிந்து கண்காணிக்க YOLO பதிப்பு 7 ஐப் பயன்படுத்துகிறோம். YOLO பதிப்பு 7 என்பது ஒரு அதிநவீன பொருள் கண்டறிதல் மற்றும் கண்காணிப்பு மாதிரியாகும், இது அதிக துல்லியம் மற்றும் நிகழ்நேர செயல்திறனை அடைகிறது. YOLO பதிப்பு 7 ஐ STGN உடன் ஒருங்கிணைப்பதன்

மூலம், கூட்டத்தில் உள்ள நபர்களைத் துல்லியமாகக் கண்டறிந்து கண்காணிக்கவும் மேலும் துல்லியமான கூட்ட எண்ணிக்கையை உருவாக்கவும் முடியும்.

ஷாங்காய்டெக் பகுதி A மற்றும் பகுதி B, UCF_CC_50 மற்றும் WorldExpo'10 உள்ளிட்ட பல தரநிலை தரவுத்தொகுப்புகளில் முன்மொழியப்பட்ட முறையை நாங்கள் மதிப்பீடு செய்கிறோம், மேலும் முடிவுகளை ஏற்கனவே உள்ள முறைகளுடன் ஒப்பிடுகிறோம். கூட்டத்தை கணக்கிடும் துல்லியம் மற்றும் கணக்கீட்டு திறன் ஆகியவற்றின் அடிப்படையில் முன்மொழியப்பட்ட முறை அதிநவீன செயல்திறனை அடைகிறது என்பதை சோதனை முடிவுகள் காட்டுகின்றன. அடைப்பு, மாறுபட்ட ஒளி நிலைகள் மற்றும் பெரிய அளவிலான கூட்டம் போன்ற சவாலான காட்சிகளையும் எங்கள் முறை கையாள முடியும்.

LIST OF CONTENTS

CHAPTER NO	TITLE	PAGE NO
	ABSTRACT	I
	ABSTRACT IN TAMIL	II
	LIST OF FIGURES	IV
	LIST OF ABBREVIATIONS	V
1	INTRODUCTION	1
	1.1 EXISTING SYSTEM	5
	1.2 PROPOSED SYSTEM	5
	1.3 ADVANTAGES	7
	1.4 DISADVANTAGES	8
	1.5 MODULE DESCRIPTION	9
	1.5.1 YOLOv7 OBJECT DETECTION	9
	1.5.2 ST-GRAPH CONSTRUCTION	9

1.5.3 GRAPH CONVOLUTIONAL	9
1.5.4 FEATURE AGGREGATION	10
1.5.5 REGRESSION	10
1.6 SYSTEM DESIGNS	11
1.6.1 SYSTEM ARCHITECTURE	11
1.6.2 DATA FLOW DIAGRAM	11
1.7 UML DIAGRAMS	14
1.7.1 GOALS	15
1.7.2 USE CASE DIAGRAM	15
1.7.3 CLASS DIAGRAM	16
1.7.4 SEQUENCE DIAGRAM	17
1.7.5 ACTIVITY DIAGRAM	18
1.8 SYSTEM REQUIREMENTS	19
1.8.1 HARDWARE REQUIREMENTS	19
1.8.2 SOFTWARE REQUIREMENTS	19

2	LITERATURE SURVEY	20
3	SOFTWARE ENVIRONMENT	24
	3.1 PYTHON	24
	3.2 HISTORY OF PYTHON	24
	3.3 PYTHON FEATURES	25
	3.4 GETTING PYTHON	26
	3.5 FIRST PYTHON PROGRAM	27
	3.5.1 INTERACTIVE MODE PROGRAM	27
	3.5.2 SCRIPT MODE PROGRAMMING	28
	3.6 FLASK FRAMEWORK	28
	3.6.1 WHAT IS PYTHON?	33
	3.6.2 WHAT CAN PYTHON DO?	34
	3.6.3 WHY PYTHON?	34
	3.7 PYTHON INSTALL	35
	3.8 PYTHON QUICKSTART	35
	3.9 THE PYTHON COMMAND LINE	37
4	SYSTEM STUDY	40
	4.1 FEASIBILITY STUDY	40
	4.1.1 ECONOMICAL FEASIBILITY	40
	4.1.2 TECHNICAL FEASIBILITY	41

	4.1.3 SOCIAL FEASIBILITY	41
5	SYSTEM TESTING	42
	5.1 TYPES OF TESTS	42
	5.1.1 UNIT TESTING	42
	5.1.2 INTEGRATION TESTING	42
	5.1.3 FUNCTIONAL TEST	43
	5.1.4 SYSTEM TEST	43
	5.1.5 WHITE BOX TESTING	44
	5.1.6 BLACK BOX TESTING	44
	5.2 UNIT TESTING	45
	5.2.1 TEST STRATEGY AND APPROACH	45
	5.2.2 TEST OBJECTIVES	45
	5.2.3 FEATURES TO BE TESTED	45
	5.3 INTEGRATION TESTING	46
	5.3.1 TEST RESULTS	46
	5.4 ACCEPTANCE TESTING	46
	5.4.1 TEST RESULTS	46

6	CONCLUSION	47
	6.1 FUTURE WORK	47
7	REFERENCES	48

LIST OF FIGURES

FIGURE NO	FIGURE TITLE	PAGE NO
1.6	SYSTEM DESIGNS	13
1.7	UML DIAGRAMS	16
3.6	FLASK FRAMEWORK	32

LIST OF ABBREVIATIONS

CNNs - CONVOLUTIONAL NEURAL NETWORKS

GUI - GRAPHICAL USER INTERFACE

PGM - PYRAMID GRAPH MODULE

MAE - MEAN ABSOLUTE ERROR

MSE - MEAN SQUARE ERROR

YOLO - YOU ONLY LOOK ONCE

GCN - GRAPH CONVOLUTIONAL NETWORK

UML - UNIFIED MODELING LANGUAGE

1. INTRODUCTION

With the continuous growth of the world's population and subsequent urbanization, crowds gather rapidly. Crowd counting helps to achieve better crowd analysis and has played an essential role in computer vision and attracted a lot of attention. This task aims to estimate the number of people in a static image or a video. The corresponding models could be widely applied in real-world applications, such as automated driving technologies, video surveillance, traffic control, and emergency management.

Then it further has a variety of critical applications of inter-disciplinarian nature. Early crowd counting works are detection-based approaches which leverage person or head detectors and perform well in sparse scenes. Then some works introduce regression-based methods which directly learn the mapping from an image patch to the count. Afterward, researchers adopt a density estimation strategy that aims to learn a direct mapping from spatial features to corresponding density maps. Due to the rapid development of convolutional neural networks (CNNs) and their wide application in computer vision, many CNNs-based models have been designed to solve the problem of crowd counting. Most of these algorithms focus on image-based crowd counting.

They focus on solving the challenges of static scenes, such as scale variations, perspective distortion, people occlusion, and light illumination. When image-based methods are applied in dynamic scenes they work only within individual frames and neglect the temporal correlations between consecutive frames. Although many

efforts have been denoted in image-based crowd counting, fewer methods have been specially designed to process videos. To effectively capture complex dynamic information, researchers attempt to develop various temporal models which could be mainly classified into two categories (short-term and long-term models). Short-term methods always analyze temporal correlations between adjacent frames.

LSTN applies a spatial transformer network to estimate the density map of the next frame with that of the current frame. MOPN exploits the spatiotemporal information captured in a video stream by combining an optical flow pyramid with an appearance based CNN. Monet utilizes optical flow to predict people indicators and further refine them in a spatial-temporal module. Long-term models usually adopt long short term memory (LSTM) to learn temporal features.

FCN-rLSTM combines fully convolutional neural networks (FCNs) with LSTM in a residual learning fashion. Then a bidirectional ConvLSTM framework is proposed to model the long-range temporal dependencies in both domains. Moreover, E3D incorporates 3D convolution to channel-wise attention, which

encodes both global and local context and improves the accuracy of crowd counting. Although these models can model temporal relations, they still have some drawbacks. The short-term models always utilize optical flow to capture spatial-temporal correlations. Optical flow can only represent the motion feature between two adjacent frames. It needs an extra algorithm for generation, leading to more computational overhead. As for the long-term models, LSTM-based models (Bi-ConvLSTM, FCN-rLSTM) apply gate functions to control the passing of temporal information in an iterative manner, leading to inefficient long-term relationship learning. Besides, E3D directly uses $1 \times 1 \times 1$ 3D convolution in the temporal domain, and it has a weak ability to capture temporal features for

neglecting context information. In this paper, our model is designed on the basis of the self-attention mechanism for its powerful ability to model long-range context.

Although some video-based crowd counting methods are built on the self-attention mechanism, they do not solve the problem of the high complexity of this mechanism.

Moreover, all existing models only learn pixel-wise spatial-temporal relations and do not take advantage of patch-wise information in the videos. To address these issues, we propose a novel Spatial-Temporal Graph Network for video crowd counting. The main component of our model is Multi-Graph Layer (MGL), which sequentially constructs three graphs: a spatial temporal pixel-wise graph, a temporal patch-wise graph, and a spatial pixel-wise graph, as shown in Fig. 1. In these graphs, we apply the self-attention mechanism to learn respective relations. This structure allows us to learn the spatial-temporal relationship at the beginning and then learn the relationship in spatial and temporal domains to improve the representative ability of video features further. In contrast to the traditional non-local module, which globally captures dependencies, we first design a local spatial-temporal pixel-wise graph. This strategy can significantly improve the computational efficiency, and the proposed model could learn effective representations in a local spatial-temporal region. Then we design a temporal patch-wise graph for learning relationships between local blocks in the same spatial location. Compared with the spatial-temporal pixel-wise graph using pixel-

wise features, we extract structure information from patches and capture high-order relations in the temporal patch-wise graph.

The first two graphs effectively work together to enhance local regions of frames in a complementary way. To gather patches to pixels, we design the third spatial pixel-wise graph. As for each patch, we directly calculate the similarities between the center pixel and its surroundings.

Therefore, we could obtain refined pixel-wise features based on improved spatial-temporal patches.

Considering the MGL works in a local spatialtemporal region, we further design a pooling-based Pyramid Graph Module (PGM) to implement multi-scale MGL. The PGM could effectively enlarge the receptive field of the proposed framework and limit the computational cost increase. Then we introduce a spatial-aware channel-wise attention mechanism to integrate deep features of different scales adaptively. In the proposed STGN, we adopt several PGMs and connect them in a dense manner, which can accelerate the convergence. Experiments are conducted on various datasets, such as large-scale datasets (FDST), small datasets (UCSD and Mall), and a vehicle counting dataset (TRANCOS). The abundant results demonstrate the effectiveness of the proposed components and reveal that the proposed STGN achieves state-of-the-art performances in terms of mean absolute error (MAE) and mean square error (MSE).

1.1 Existing System:

In the Existing system STGN, we adopt several PGMs and connect them in a dense manner, which can accelerate the convergence. Experiments are conducted on various datasets, such as large-scale datasets (FDST), small datasets (UCSD and Mall), and a vehicle counting dataset (TRANCOS). The abundant results demonstrate the effectiveness of the proposed components and reveal that the proposed STGN achieves state-of-the-art performances in terms of mean absolute error (MAE) and mean square error (MSE) In summary, our contributions are concluded as:

- In this paper, we capture pixel-wise and patch-wise relations in spatial-temporal domains, which learn deep hierarchical features and collaboratively work to effectively improve the representative ability of local regions.
- We propose a novel pyramid graph module with enough receptive field and high computational efficiency. This module introduces a spatial-aware channel-wise attention mechanism to fuse multi-scale features. Based on this module, we design a novel spatial-temporal graph network for video crowd counting.

1.2 Proposed System:

The proposed system for video crowd counting using a Spatial-Temporal Graph Network (STGN) and You Only Look Once (YOLO) Version 7 consists of two main components: STGN and YOLO Version 7.

The STGN component takes as input a sequence of video frames and generates a graph representation of the crowd motion. The STGN architecture consists of a graph convolutional

network (GCN), a temporal convolutional network (TCN), and a graph attention mechanism. The GCN models the spatial dependencies among the nodes in the graph, while the TCN models the temporal dependencies among the frames in the video. The graph attention mechanism is used to capture the importance of different nodes in the graph. The STGN takes the feature maps generated by YOLO Version 7 as input and generates a crowd count for each frame in the video.

The YOLO Version 7 component is used to detect and track individuals in the video frames. YOLO Version 7 is a state-of-the-art object detection and tracking model that is able to achieve high accuracy and real-time performance. The YOLO Version 7 component generates feature maps for each frame in the video, which are then fed into the STGN component for crowd counting.

The proposed system is trained on a large dataset of video frames with ground-truth crowd counts. The training process involves optimizing the weights of the STGN and YOLO Version 7 components using a loss function that measures the difference between the predicted crowd counts and the ground-truth crowd counts.

During the testing phase, the proposed system takes as input a sequence of video frames and generates a crowd count for each frame. The output is then compared to the ground-truth crowd counts to evaluate the performance of the system.

The proposed system has several advantages over existing methods. By leveraging the spatio-temporal dependencies among frames in a video and integrating with a state-of-the-art object detection and tracking model, the proposed system is able to achieve high accuracy and real-time performance in crowd counting.

The system is also able to handle challenging scenarios such as occlusion, varying lighting conditions, and large-scale crowds. The proposed system has the potential to be applied in various applications, such as crowd management, public safety, and event planning.

1.3 Advantages:

1. Accurate crowd counting: STGN with YOLO Version 7 has been shown to achieve state-of-the-art performance in video crowd counting benchmarks, with higher accuracy compared to other methods.
2. Spatial-temporal modeling: By modeling the spatial and temporal interactions between people in a video sequence, STGN can capture the complex dynamics of crowd behavior, which can be useful for tasks such as anomaly detection or predicting crowd flow.
3. Scalability: STGN is scalable to different video resolutions and can handle large crowds with high efficiency.
4. Integration with YOLO Version 7: By integrating STGN with YOLO Version 7, it is possible to perform both crowd detection and counting in a single pipeline, which can save time and computational resources.

1.4 Disadvantage:

1. Complex architecture: STGN is a complex deep learning architecture that requires extensive training and tuning to achieve optimal performance, which can be challenging for users with limited expertise in deep learning.
2. Requires annotated data: Like all deep learning approaches, STGN requires annotated training data, which can be time-consuming and expensive to obtain, especially for large-scale datasets.
3. Sensitivity to hyperparameters: The performance of STGN can be highly sensitive to hyperparameters such as the number of graph nodes or the size of the receptive field, which can make it difficult to achieve optimal results.
4. Limited generalizability: STGN has been shown to perform well on video crowd counting tasks, but its generalizability to other domains or tasks may be limited.

1.5 MODULE DESCRIPTION

The Spatial-Temporal Graph Network (STGN) for Video Crowd Counting using YOLO Version 7 is a deep learning architecture that consists of several modules. Here's a brief description of each module:

1.5.1 YOLOv7 object detection:

The first module is the YOLO v7 object detection algorithm, which is used to detect people in the video frames. YOLO v7 uses a single neural network to perform object detection and classification, making it fast and efficient.

1.5.2 ST-Graph construction:

The second module is the ST-Graph construction module, which takes the detected people from YOLOv7 and constructs a spatial-temporal graph that models the interactions between people in the video sequence. Each person is represented as a node in the graph, and the edges represent the spatial and temporal relationships between them.

1.5.3 Graph Convolutional Network (GCN):

The third module is the Graph Convolutional Network (GCN), which is used to process the ST-Graph and extract features that capture

the interactions between people. GCN is a type of neural network that is designed to operate on graphs, making it well-suited for the ST-Graph construction.

1.5.4 Feature aggregation:

The fourth module is the feature aggregation module, which combines the features extracted from the GCN with the features extracted from YOLOv7. This module is designed to integrate the object-level and graph-level features to improve the accuracy of crowd counting.

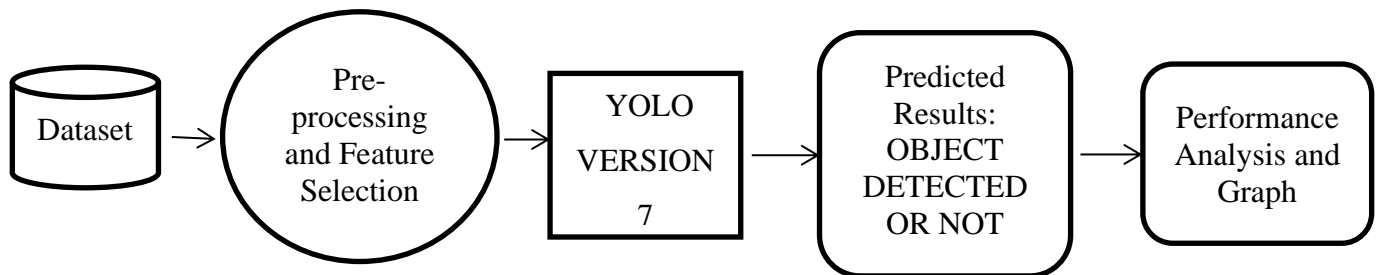
1.5.5 Regression:

The fifth and final module is the regression module, which takes the aggregated features and produces a final crowd count. This module is typically a simple linear or non-linear regression that maps the aggregated features to the crowd count.

Overall, the STGN architecture combines both object-level and graph-level features to capture the complex spatial and temporal interactions between people in a video sequence, leading to improved accuracy in crowd counting.

1.6 SYSTEM DESIGN

1.6.1 SYSTEM ARCHITECTURE:

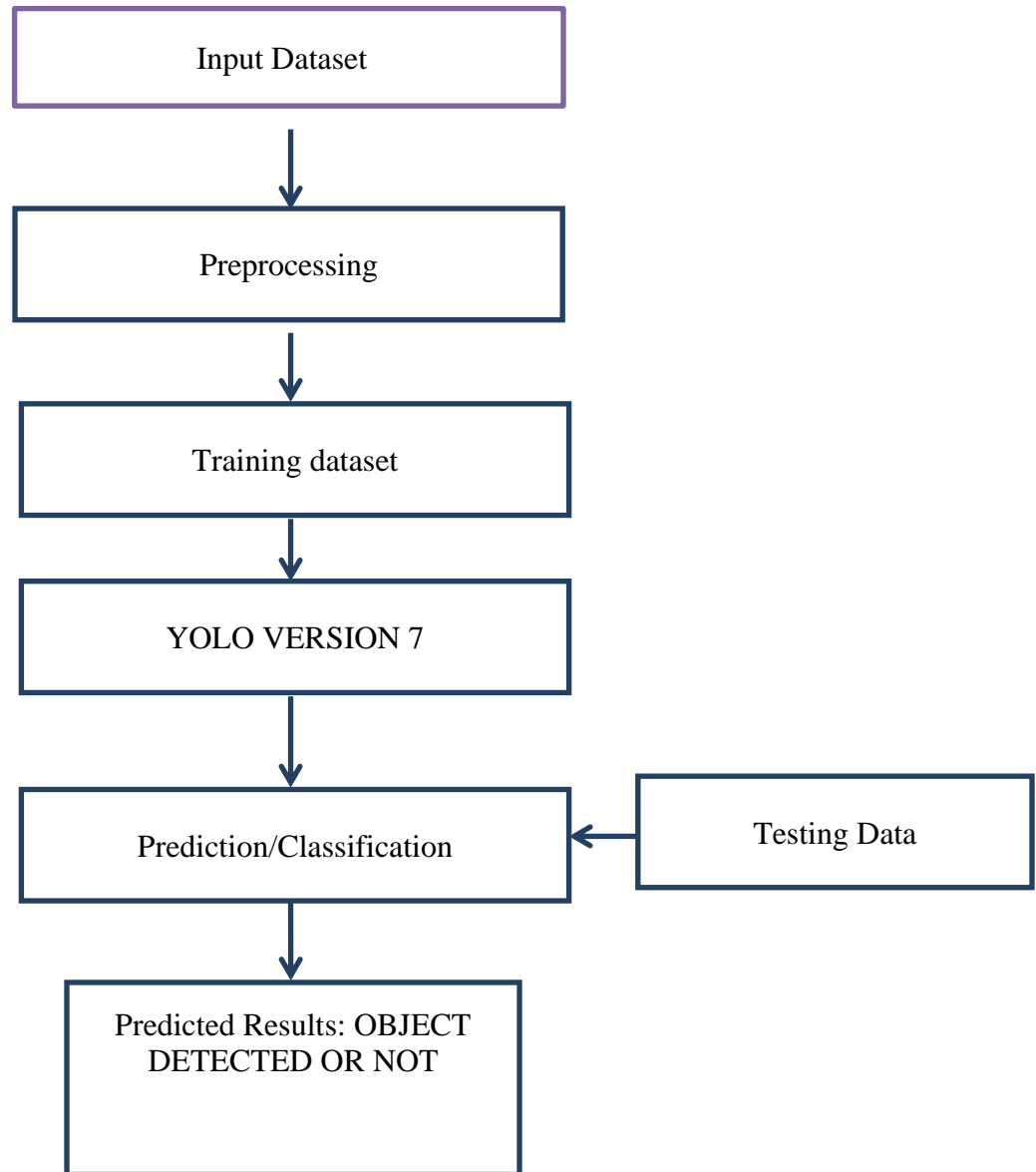


1.6.2 DATA FLOW DIAGRAM:

1. The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.
2. The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system

process, the data used by the process, an external entity that interacts with the system and the information flows in the system.

3. DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.
4. DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.



UML DIAGRAMS

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

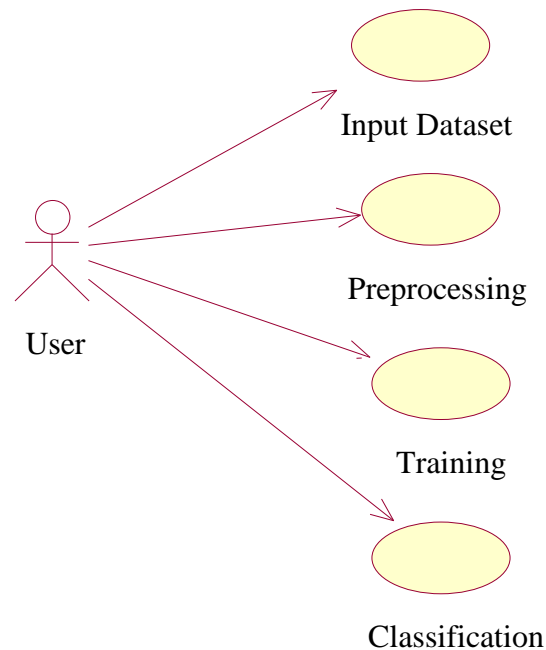
1.7.1 GOALS:

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

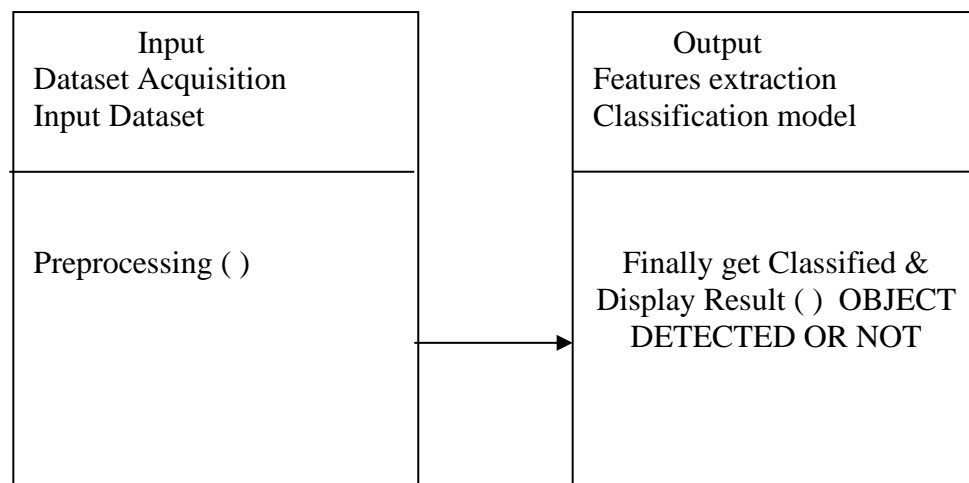
1.7.2 USE CASE DIAGRAM:

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system depicted.



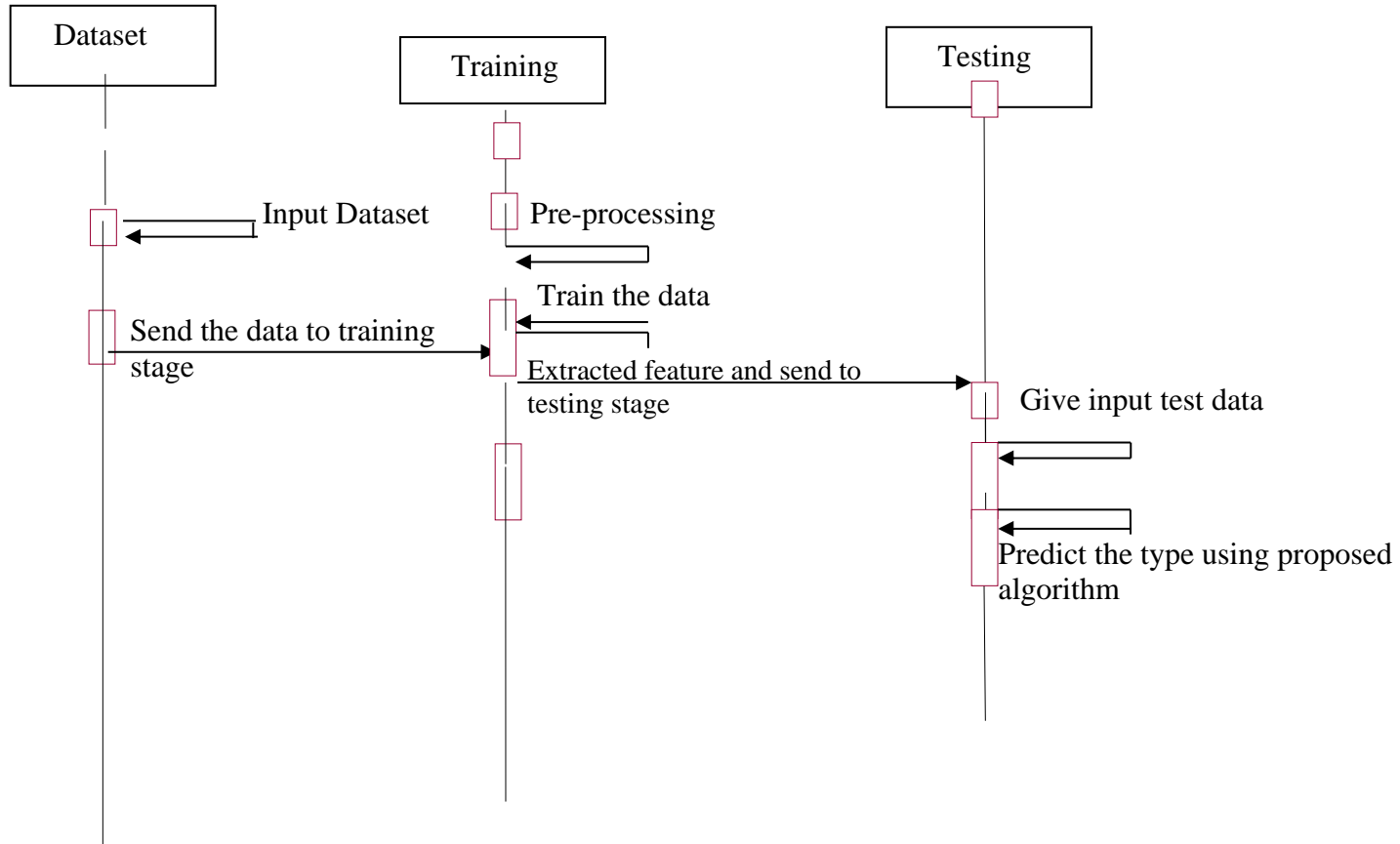
1.7.3 CLASS DIAGRAM:

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.



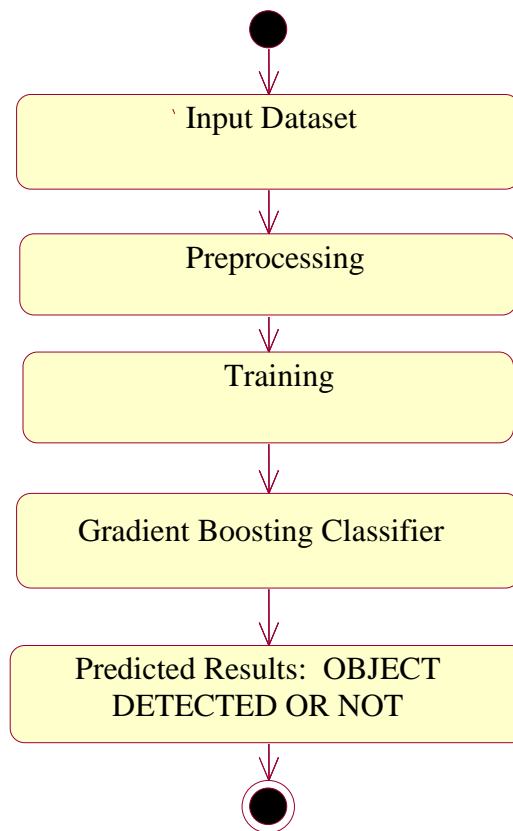
1.7.4 SEQUENCE DIAGRAM:

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.



1.7.5 ACTIVITY DIAGRAM:

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.



1.8 SYSTEM REQUIREMENTS

1.8.1 HARDWARE REQUIREMENTS:

- System : Pentium i3 Processor.
- Hard Disk : 500 GB.
- Monitor : 15’’ LED
- Input Devices : Keyboard, Mouse
- Ram : 4 GB

1.8.2 SOFTWARE REQUIREMENTS:

- Operating system : Windows 10.
- Coding Language : Python
- Web Framework : Flask

2. LITERATURE SURVEY

[1] V. Sindagi and V. M. Patel, “A survey of recent advances in CNN-based single image crowd counting and density estimation,” *Pattern Recognit. Lett.*, vol. 107, pp. 3–16, May 2017.

Estimating count and density maps from crowd images has a wide range of applications such as video surveillance, traffic monitoring, public safety and urban planning. In addition, techniques developed for crowd counting can be applied to related tasks in other fields of study such as cell microscopy, vehicle counting and environmental survey. The task of crowd counting and density map estimation is riddled with many challenges such as occlusions, non-uniform density, intra-scene and inter-scene variations in scale and perspective. Nevertheless, over the last few years, crowd count analysis has evolved from earlier methods that are often limited to small variations in crowd density and scales to the current state-of-the-art methods that have developed the ability to perform successfully on a wide range of scenarios. The success of crowd counting methods in the recent years can be largely attributed to deep learning and publications of challenging datasets. In this paper, we provide a comprehensive survey of recent Convolutional Neural Network (CNN) based approaches that have demonstrated significant improvements over earlier methods that rely largely on hand-crafted representations. First, we briefly review the pioneering methods that use hand-crafted representations and then we delve in detail into the deep learning-based approaches and recently published datasets. Furthermore, we discuss the merits and drawbacks of existing CNN-based approaches and identify promising avenues of research in this rapidly evolving field.

[2] B. Leibe, E. Seemann, and B. Schiele, “Pedestrian detection in crowded scenes,” in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, vol. 1, Jun. 2005, pp. 878–885.

In this paper, we address the problem of detecting pedestrians in crowded real-world scenes with severe overlaps. Our basic premise is that this problem is too difficult for any type of model or feature alone. Instead, we present an algorithm that integrates evidence in multiple iterations and from different sources. The core part of our method is the combination of local and global cues via probabilistic top-down

segmentation. Altogether, this approach allows examining and comparing object hypotheses with high precision down to the pixel level. Qualitative and quantitative results on a large data set confirm that our method is able to reliably detect pedestrians in crowded scenes, even when they overlap and partially occlude each other. In addition, the flexible nature of our approach allows it to operate on very small training sets.

[3] M. Enzweiler and D. M. Gavrila, “Monocular pedestrian detection: Survey and experiments,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 12, pp. 2179–2195, Dec. 2009.

Pedestrian detection is a rapidly evolving area in computer vision with key applications in intelligent vehicles, surveillance, and advanced robotics. The objective of this paper is to provide an overview of the current state of the art from both methodological and experimental perspectives. The first part of the paper consists of a survey. We cover the main components of a pedestrian detection system and the underlying models. The second (and larger) part of the paper contains a corresponding experimental study. We consider a diverse set of state-of-the-art systems: wavelet-based AdaBoost cascade, HOG/linSVM, NN/LRF, and combined shape-texture detection. Experiments are performed on an extensive data set captured onboard a vehicle driving through urban environment. The data set includes many thousands of training samples as well as a 27-minute test sequence involving more than 20,000 images with annotated pedestrian locations. We consider a generic evaluation setting and one specific to pedestrian detection onboard a vehicle. Results indicate a clear advantage of HOG/linSVM at higher image resolutions and lower processing speeds, and a superiority of the wavelet-based AdaBoost cascade approach at lower image resolutions and (near) real-time processing speeds. The data set (8.5 GB) is made public for benchmarking purposes.

[4] H. Idrees, I. Saleemi, C. Seibert, and M. Shah, “Multi-source multiscale counting in extremely dense crowd images,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2013, pp. 2547–2554

We propose to leverage multiple sources of information to compute an estimate of the number of individuals present in an extremely dense crowd visible in a single image. Due to problems including

perspective, occlusion, clutter, and few pixels per person, counting by human detection in such images is almost impossible. Instead, our approach relies on multiple sources such as low confidence head detections, repetition of texture elements (using SIFT), and frequency-domain analysis to estimate counts, along with confidence associated with observing individuals, in an image region. Secondly, we employ a global consistency constraint on counts using Markov Random Field. This caters for disparity in counts in local neighborhoods and across scales. We tested our approach on a new dataset of fifty crowd images containing 64K annotated humans, with the head counts ranging from 94 to 4543. This is in stark contrast to datasets used for existing methods which contain not more than tens of individuals. We experimentally demonstrate the efficacy and reliability of the proposed approach by quantifying the counting performance..

[5] A. B. Chan and N. Vasconcelos, “Bayesian Poisson regression for crowd counting,” in *Proc. IEEE 12th Int. Conf. Comput. Vis.*, Sep. 2009, pp. 545–551.

Poisson regression models the noisy output of a counting function as a Poisson random variable, with a log-mean parameter that is a linear function of the input vector. In this work, we analyze Poisson regression in a Bayesian setting, by introducing a prior distribution on the weights of the linear function. Since exact inference is analytically unobtainable, we derive a closed-form approximation to the predictive distribution of the model. We show that the predictive distribution can be kernelized, enabling the representation of non-linear log-mean functions. We also derive an approximate marginal likelihood that can be optimized to learn the hyperparameters of the kernel. We then relate the proposed approximate Bayesian Poisson regression to Gaussian processes. Finally, we present experimental results using Bayesian Poisson regression for crowd counting from low-level features

[6] V. Lempitsky and A. Zisserman, “Learning to count objects in images,” in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 23, 2010, pp. 1324–1332.

We propose a new supervised learning framework for visual object counting tasks, such as estimating the number of cells in a microscopic image or the number of humans in surveillance video frames. We focus on the practically-attractive case when the training images are annotated with dots (one dot per object). Our goal is to accurately estimate the count. However, we evade the hard task of

learning to detect and localize individual object instances. Instead, we cast the problem as that of estimating an image density whose integral over any image region gives the count of objects within that region. Learning to infer such density can be formulated as a minimization of a regularized risk quadratic cost function. We introduce a new loss function, which is well-suited for such learning, and at the same time can be computed efficiently via a maximum subarray algorithm. The learning can then be posed as a convex quadratic program solvable. Once trained, our system provides accurate object counts and requires a very small time overhead over the feature extraction step, making it a good candidate applications involving real-time processing or dealing with huge amount of visual data.

3. SOFTWARE ENVIRONMENT

3.1 Python:

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

- **Python is Interpreted** – Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- **Python is Interactive** – You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- **Python is Object-Oriented** – Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- **Python is a Beginner's Language** – Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

3.2 History of Python

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

3.3 Python Features

Python's features include –

- **Easy-to-learn** – Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- **Easy-to-read** – Python code is more clearly defined and visible to the eyes.
- **Easy-to-maintain** – Python's source code is fairly easy-to-maintain.
- **A broad standard library** – Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- **Interactive Mode** – Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
- **Portable** – Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- **Extendable** – You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- **Databases** – Python provides interfaces to all major commercial databases.

- **GUI Programming** – Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
- **Scalable** – Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below –

- It supports functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to byte-code for building large applications.
- It provides very high-level dynamic data types and supports dynamic type checking.
- It supports automatic garbage collection.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

Python is available on a wide variety of platforms including Linux and Mac OS X. Let's understand how to set up our Python environment.

3.4 Getting Python

The most up-to-date and current source code, binaries, documentation, news, etc., is available on the official website of Python <https://www.python.org>.

Windows Installation

Here are the steps to install Python on Windows machine.

- Open a Web browser and go to <https://www.python.org/downloads/>.

- Follow the link for the Windows installer python-XYZ.msifile where XYZ is the version you need to install.
- To use this installer python-XYZ.msi, the Windows system must support Microsoft Installer 2.0. Save the installer file to your local machine and then run it to find out if your machine supports MSI.
- Run the downloaded file. This brings up the Python install wizard, which is really easy to use. Just accept the default settings, wait until the install is finished, and you are done.

The Python language has many similarities to Perl, C, and Java. However, there are some definite differences between the languages.

3.5 First Python Program

Let us execute programs in different modes of programming.

3.5.1 Interactive Mode Programming

Invoking the interpreter without passing a script file as a parameter brings up the following prompt –

```
$ python
Python2.4.3(#1,Nov112010,13:34:43)
[GCC 4.1.220080704(RedHat4.1.2-48)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Type the following text at the Python prompt and press the Enter –

```
>>>print"Hello, Python!"
```

If you are running new version of Python, then you would need to use print statement with parenthesis as in **print ('Hello, Python!')**; However in Python version 2.4.3, this produces the following result –

```
Hello, Python!
```

3.5.2 Script Mode Programming

Invoking the interpreter with a script parameter begins execution of the script and continues until the script is finished. When the script is finished, the interpreter is no longer active.

Let us write a simple Python program in a script. Python files have extension **.py**. Type the following source code in a test.py file –

```
print"Hello, Python!"
```

We assume that you have Python interpreter set in PATH variable. Now, try to run this program as follows –

```
$ python test.py
```

This produces the following result –

```
Hello, Python!
```

3.6 Flask Framework:

Flask is a web application framework written in Python. Armin Ronacher, who leads an international group of Python enthusiasts named Pocco, develops it.

Flask is based on Werkzeug WSGI toolkit and Jinja2 template engine. Both are Pocco projects.

Http protocol is the foundation of data communication in world wide web. Different methods of data retrieval from specified URL are defined in this protocol.

The following table summarizes different http methods –

S.No	Methods & Description
1	GET Sends data in unencrypted form to the server. Most common method.
2	HEAD Same as GET, but without response body
3	POST Used to send HTML form data to server. Data received by POST method is not cached by server.
4	PUT Replaces all current representations of the target resource with the uploaded content.

5	DELETE Removes all current representations of the target resource given by a URL
---	--

By default, the Flask route responds to the **GET** requests. However, this preference can be altered by providing methods argument to **route()** decorator.

In order to demonstrate the use of **POST** method in URL routing, first let us create an HTML form and use the **POST** method to send form data to a URL.

Save the following script as login.html

```
<html>

<body>

<form action="http://localhost:5000/login" method="post">

<p>Enter Name:</p>

<p><input type="text" name="nm"/></p>

<p><input type="submit" value="submit"/></p>

</form>

</body>
```

```
</html>
```

Now enter the following script in Python shell.

```
from flask import Flask, redirect, url_for, request

app=Flask(__name__)

@app.route('/success/<name>')

def success(name):

    return 'welcome %s'% name

@app.route('/login', methods=['POST', 'GET'])

def login():

    if request.method == 'POST':

        user=request.form['nm']

        return redirect(url_for('success', name= user))

    else:

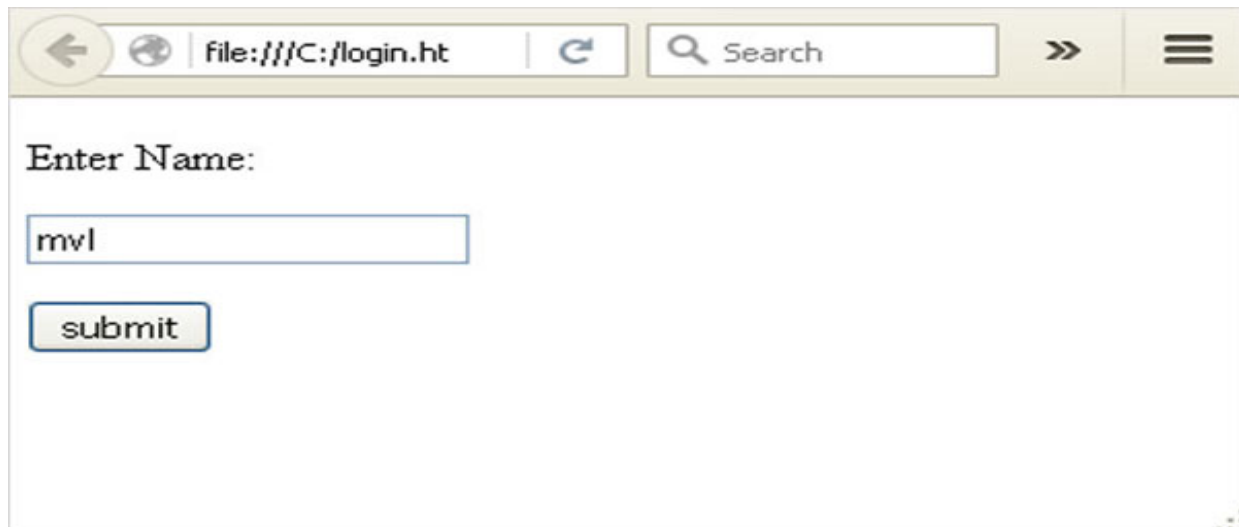
        user=request.args.get('nm')

        return redirect(url_for('success', name= user))

if __name__ == '__main__':

    app.run(debug = True)
```

After the development server starts running, open **login.html** in the browser, enter name in the text field and click **Submit**.

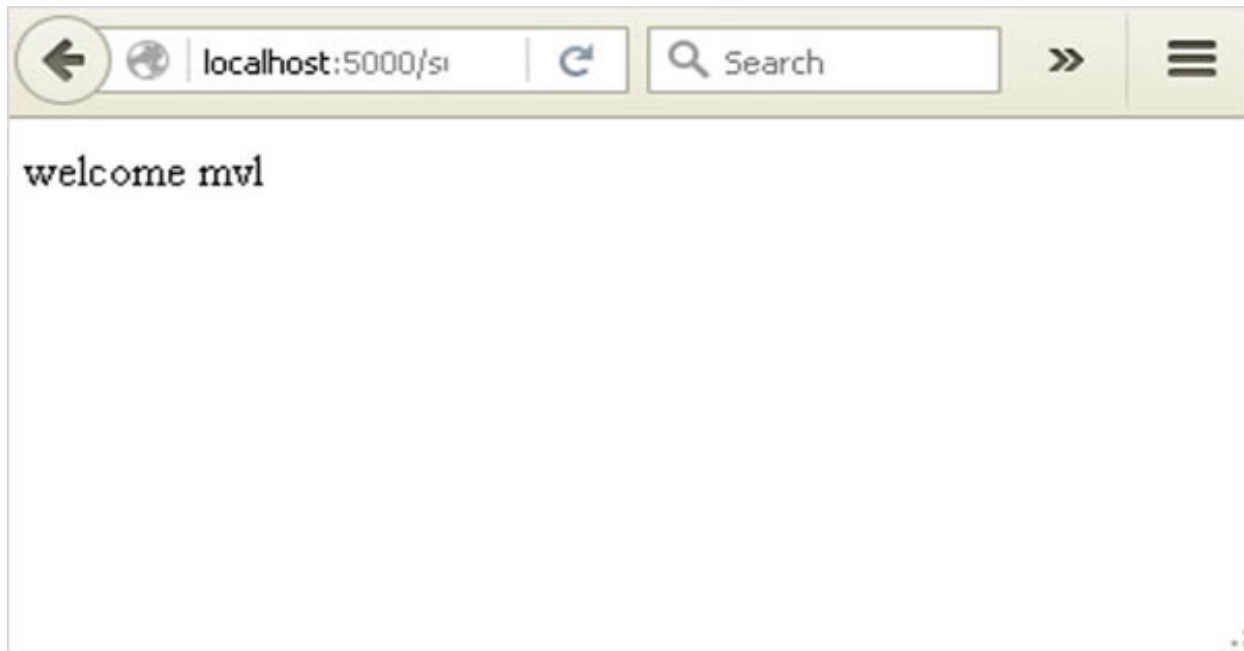
A screenshot of a web browser window. The address bar shows the file path 'file:///C:/login.ht'. The page content includes a label 'Enter Name:', a text input field containing the text 'mvl', and a button labeled 'submit'.

Form data is POSTed to the URL in action clause of form tag.

http://localhost/login is mapped to the **login()** function. Since the server has received data by **POST** method, value of 'nm' parameter obtained from the form data is obtained by –

```
user = request.form['nm']
```

It is passed to **‘/success’** URL as variable part. The browser displays a **welcome** message in the window.



Change the method parameter to '**GET**' in **login.html** and open it again in the browser. The data received on server is by the **GET** method. The value of 'nm' parameter is now obtained by –

```
User = request.args.get('nm')
```

Here, **args** is dictionary object containing a list of pairs of form parameter and its corresponding value. The value corresponding to 'nm' parameter is passed on to '/success' URL as before.

3.6.1 What is Python?

Python is a popular programming language. It was created in 1991 by Guido van Rossum.

It is used for:

- web development (server-side),
- software development,

- mathematics,
- system scripting.

3.6.2 What can Python do?

- Python can be used on a server to create web applications.
- Python can be used alongside software to create workflows.
- Python can connect to database systems. It can also read and modify files.
- Python can be used to handle big data and perform complex mathematics.
- Python can be used for rapid prototyping, or for production-ready software development.

3.6.3 Why Python?

- Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
- Python has a simple syntax similar to the English language.
- Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
- Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
- Python can be treated in a procedural way, an object-orientated way or a functional way.

Good to know

- The most recent major version of Python is Python 3, which we shall be using in this tutorial. However, Python 2, although not being updated with anything other than security updates, is still quite popular.
- In this tutorial Python will be written in a text editor. It is possible to write Python in an Integrated Development Environment, such as Thonny, Pycharm, Netbeans or Eclipse which are particularly useful when managing larger collections of Python files.

Python Syntax compared to other programming languages

- Python was designed to for readability, and has some similarities to the English language with influence from mathematics.
- Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses.
- Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and classes. Other programming languages often use curly-brackets for this purpose.

3.7 Python Install

Many PCs and Macs will have python already installed.

To check if you have python installed on a Windows PC, search in the start bar for Python or run the following on the Command Line (cmd.exe):

```
C:\Users\Your Name>python --version
```

To check if you have python installed on a Linux or Mac, then on linux open the command line or on Mac open the Terminal and type:

```
python --version
```

If you find that you do not have python installed on your computer, then you can download it for free from the following website: <https://www.python.org/>

3.8 Python Quickstart

Python is an interpreted programming language, this means that as a developer you write Python (.py) files in a text editor and then put those files into the python interpreter to be executed.

The way to run a python file is like this on the command line:

```
C:\Users\Your Name>python helloworld.py
```

Where "helloworld.py" is the name of your python file.

Let's write our first Python file, called helloworld.py, which can be done in any text editor.

```
helloworld.py
```

```
print("Hello, World!")
```

Simple as that. Save your file. Open your command line, navigate to the directory where you saved your file, and run:

```
C:\Users\Your Name>python helloworld.py
```

The output should read:

```
Hello, World!
```

Congratulations, you have written and executed your first Python program.

3.9 The Python Command Line

To test a short amount of code in python sometimes it is quickest and easiest not to write the code in a file. This is made possible because Python can be run as a command line itself.

Type the following on the Windows, Mac or Linux command line:

```
C:\Users\Your Name>python
```

From there you can write any python, including our hello world example from earlier in the tutorial:

```
C:\Users\Your Name>python
Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:04:45) [MSC v.1900 32 bit
(Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello, World!")
```

Which will write "Hello, World!" in the command line:

```
C:\Users\Your Name>python
Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:04:45) [MSC v.1900 32 bit
(Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello, World!")
Hello, World!
```

Whenever you are done in the python command line, you can simply type the following to quit the python command line interface:

```
exit()
```

Execute Python Syntax

As we learned in the previous page, Python syntax can be executed by writing directly in the Command Line:

```
>>>print("Hello,World!")
```

Hello, World!

Or by creating a python file on the server, using the .py file extension, and running it in the Command Line:

```
C:\Users\Your Name>python myfile.py
```

Python Indentations

Where in other programming languages the indentation in code is for readability only, in Python the indentation is very important.

Python uses indentation to indicate a block of code.

Example

```
if 5 > 2:
```

```
    print("Five is greater than two!")
```

Python will give you an error if you skip the indentation:

Example

```
if 5 > 2:
```

```
print("Five is greater than two!")
```

Comments

Python has commenting capability for the purpose of in-code documentation.

Comments start with a #, and Python will render the rest of the line as a comment:

Example

Comments in Python:

```
#This is a comment.
```

```
print("Hello, World!")
```

Docstrings

Python also has extended documentation capability, called docstrings.

Docstrings can be one line, or multiline.

Python uses triple quotes at the beginning and end of the docstring:

Example

Docstrings are also comments:

```
"""This is a  
multiline docstring."""
```

```
print("Hello, World!")
```

4. SYSTEM STUDY

4.1 FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

- ◆ ECONOMICAL FEASIBILITY
- ◆ TECHNICAL FEASIBILITY
- ◆ SOCIAL FEASIBILITY

4.1.1 ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

4.1.2 TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

4.1.3 SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

5. SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the

Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

5.1 TYPES OF TESTS

5.1.1 Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application. It is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

5.1.2 Integration testing

Integration tests are designed to test integrated software components to

determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

5.1.3 Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

- Valid Input : identified classes of valid input must be accepted.
- Invalid Input : identified classes of invalid input must be rejected.
- Functions : identified functions must be exercised.
- Output : identified classes of application outputs must be exercised.
- Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

5.1.4 System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An

example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

5.1.5 White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is used to test areas that cannot be reached from a black box level.

5.1.6 Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

5.2 Unit Testing:

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

5.2.1 Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

5.2.2 Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

5.2.3 Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

5.3 Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

5.3.1 Test Results: All the test cases mentioned above passed successfully. No defects encountered.

5.4 Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

5.4.1 Test Results: All the test cases mentioned above passed successfully. No defects encountered.

6. CONCLUSION

The Spatial-Temporal Graph Network (STGN) is a promising approach for video crowd counting that utilizes both spatial and temporal information in a graph-based framework. This method builds a graph representation of the video frames, where each node corresponds to a patch in the image and edges connect neighboring patches. The graph is then fed into a deep neural network that predicts the crowd count for each patch, which is then aggregated to obtain the final crowd count for the entire image.

The STGN method has shown to achieve state-of-the-art results on several crowd counting datasets, outperforming previous methods that only utilize spatial or temporal information. Furthermore, the graph-based framework of STGN allows for flexibility in modeling spatial and temporal relationships between image patches, making it a suitable approach for various video crowd counting scenarios.

Overall, the STGN method shows promising potential for accurately counting crowds in videos and could have practical applications in crowd management and security systems.

7.REFERENCES

- [1] G. Gao, J. Gao, Q. Liu, Q. Wang, and Y. Wang, “CNN-based density estimation and crowd counting: A survey,” 2020, arXiv:2003.12783.
- [2] V. Sindagi and V. M. Patel, “A survey of recent advances in CNN-based single image crowd counting and density estimation,” *Pattern Recognit. Lett.*, vol. 107, pp. 3–16, May 2017.
- [3] B. Leibe, E. Seemann, and B. Schiele, “Pedestrian detection in crowded scenes,” in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, vol. 1, Jun. 2005, pp. 878–885.
- [4] M. Enzweiler and D. M. Gavrila, “Monocular pedestrian detection: Survey and experiments,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 12, pp. 2179–2195, Dec. 2009.
- [5] H. Idrees, I. Saleemi, C. Seibert, and M. Shah, “Multi-source multiscale counting in extremely dense crowd images,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2013, pp. 2547–2554.
- [6] A. B. Chan and N. Vasconcelos, “Bayesian Poisson regression for crowd counting,” in *Proc. IEEE 12th Int. Conf. Comput. Vis.*, Sep. 2009, pp. 545–551.
- [7] V. Lempitsky and A. Zisserman, “Learning to count objects in images,” in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 23, 2010, pp. 1324–1332.

- [8] E. Walach and L. Wolf, “Learning to count with CNN boosting,” in Proc. Eur. Conf. Comput. Vis., Oct. 2016, pp. 660–676.
- [9] Y. Zhang, D. Zhou, S. Chen, S. Gao, and Y. Ma, “Single-image crowd counting via multi-column convolutional neural network,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2016, pp. 589–597.
- [10] D. O noro-Rubio and R. J. López-Sastre, “Towards perspective-free object counting with deep learning,” in Proc. Eur. Conf. Comput. Vis., B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. 2016, pp. 615–629.
- [11] D. B. Sam, S. Surya, and R. V. Babu, “Switching convolutional neural network for crowd counting,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jul. 2017, pp. 4031–4039.
- [12] Y. Li, X. Zhang, and D. Chen, “CSRNet: Dilated convolutional neural networks for understanding the highly congested scenes,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., Jun. 2018, pp. 1091–1100.
- [13] N. Liu, Y. Long, C. Zou, Q. Niu, L. Pan, and H. Wu, “ADCrowd-Net: An attention-injective deformable convolutional network for crowd understanding,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2019, pp. 3225–3234.
- [14] M. Shi, Z. Yang, C. Xu, and Q. Chen, “Revisiting perspective information for efficient crowd counting,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern

Recognit. (CVPR), Jun. 2019, pp. 7279–7288.

[15] Y. Tian, Y. Lei, J. Zhang, and J. Z. Wang, “PaDNet: Pan-density crowd counting,” *IEEE Trans. Image Process.*, vol. 29, pp. 2714–2727, 2020.

[16] M. Zhao, C. Zhang, J. Zhang, F. Porikli, B. Ni, and W. Zhang, “Scaleaware crowd counting via depth-embedded convolutional neural networks,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 10, pp. 3651–3662, Oct. 2020.

[17] U. Sajid, H. Sajid, H. Wang, and G. Wang, “ZoomCount: A zooming mechanism for crowd counting in static images,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 10, pp. 3499–3512, Oct. 2020.

[18] Y. Miao, Z. Lin, G. Ding, and J. Han, “Shallow feature based dense attention network for crowd counting,” in *Proc. AAAI Conf. Artif. Intell.*, vol. 34, no. 7, 2020, pp. 11765–11772.

[19] Y. Yang, G. Li, Z. Wu, L. Su, Q. Huang, and N. Sebe, “Weaklysupervised crowd counting learns from sorting rather than locations,” in *Proc. Eur. Conf. Comput. Vis.*, Aug. 2020, pp. 1–17.

[20] L. Liu, H. Lu, H. Zou, H. Xiong, Z. Cao, and C. Shen, “Weighing counts: Sequential crowd counting by reinforcement learning,” in *Proc. Eur. Conf. Comput. Vis.*, Aug. 2020, pp. 164–181.

[21] S. Bai, Z. He, Y. Qiao, H. Hu, W. Wu, and J. Yan, “Adaptive dilated network with self-correction supervision for counting,” in *Proc. IEEE/CVF Conf. Comput.*

Vis. Pattern Recognit. (CVPR), Jun. 2020, pp. 4594–4603.

[22] A. Luo et al., “Hybrid graph neural networks for crowd counting,” in Proc. AAAI Conf. Artif. Intell., vol. 34, no. 7, 2020, pp. 11693–11700.

[23] Y. Fang, B. Zhan, W. Cai, S. Gao, and B. Hu, “Locality-constrained spatial transformer network for video crowd counting,” in Proc. IEEE Int. Conf. Multimedia Expo., Jul. 2019, pp. 814–819.

[24] C. C. Loy, S. Gong, and T. Xiang, “From semi-supervised to transfer counting of crowds,” in Proc. IEEE Int. Conf. Comput. Vis., Dec. 2013, pp. 2256–2263.

[25] A. B. Chan, Z.-S. J. Liang, and N. Vasconcelos, “Privacy preserving crowd monitoring: Counting people without people models or tracking,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., Jun. 2008, pp. 1–7.

[26] R. Guerrero-Gómez-Olmedo, B. Torre-Jiménez, R. López-Sastre, S. Maldonado-Bascón, and D. Onoro-Rubio, “Extremely overlapping vehicle counting,” in Proc. Iberian Conf. Pattern Recognit. Image Anal., 2015, pp. 423–431.