



International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Crowd Counting And Monitoring Using Deep Learning

Prajakta Jadhav^a, Sankalp Walke^b, Kshitij Tayade^c, Praharsh Parchake^d, Prof.(Dr) H. D. Kale^e

B.E. Students, Department of Information Technology,^{a,b,c,d}

Associate Professor, Department of Information Technology^e

Prof Ram Meghe Institute of Technology and Research Badnera, Maharashtra, India

ABSTRACT:

Crowd counting and monitoring systems play a crucial role in various domains, such as public safety, event management, and urban planning. These systems aim to accurately estimate the number of people in a crowd and monitor crowd density levels to ensure efficient crowd management and safety. This abstract provides an overview of crowd counting and monitoring systems, highlighting their significance, challenges, and key methodologies.

Crowd counting involves estimating the number of individuals within a crowd, which is a challenging task due to variations in crowd density, occlusions, and complex spatial configurations. Various approaches have been proposed, including regression-based methods, density estimation techniques, and deep learning-based models that leverage convolutional neural networks (CNNs) and recurrent neural networks (RNNs). These methods have shown promising results in accurately counting people in crowded scenes.

Keywords: computer vision, machine learning, android studio, deep learning

1. Introduction

The basic concept of crowd counting and monitoring involves accurately estimating the number of individuals within a crowd and analyzing their spatial distribution and movement patterns. Here's an overview of the key concepts involved:

1. Crowd Counting: Crowd counting aims to estimate the total number of people in a given area or scene. This can be achieved using various techniques, including:
 - Density Estimation: One approach is to estimate the density of individuals in a crowd. This involves creating a density map where each pixel represents the local crowd density. The density map can be obtained by analyzing the distribution of people or by applying regression or classification algorithms.
2. Crowd Monitoring: Crowd monitoring involves analyzing the spatial and temporal behavior of individuals within a crowd. It aims to understand crowd dynamics, detect anomalies, and ensure public safety. Some key aspects of crowd monitoring include:

2. Literature Review

In this chapter a review of research papers used during project work is presented.

Real-time head detection with counting based on crowded scenes is a very challenging and computationally complex task in the case of lengthy surveillance videos. Existing head detection methods suffer from slow detection and a high rate of missed detection, especially in the case of congested crowd regions as well as occluded heads. In this work, we performed performance analysis of various YOLO (You Look Only Once) architectures for real-time head detection and counting [1].

In order to maintain crowd control and assure safety, contemporary security systems require effective monitoring tools for sensitive and high-traffic locations. In particular, headcount estimate in sensitive areas like high-security buildings, crowded areas is the subject of this research's novel using deep learning algorithms for real-time object identification in surveillance settings. The main goal is to create a reliable system that can count the

number of people in a certain area, recognize their presence, and sound an alarm when the number of people in the crowd rises over pre-established levels [2].

Crowd counting is applied in many areas including efficient resources allocation and effective management of emergency situations. In this paper, we survey and compare various crowd counting methods. Additionally, we identify the limitations of existing approaches and sketch an agenda for future work to address the identified open research challenges. Furthermore, we present an enhanced deep learning-based solution for crowd counting at bus stops [3].

Existing crowd counting technology mainly uses one single detection model or density map regression model to obtain the number of people in an image, but a single model is difficult to adapt to different camera position, viewing angle, resolution, and different crowd density. We believe that fusion of two different methods for prediction can greatly improve the accuracy of prediction in various situations. In this paper, we propose a method that improves the existing counting method of detection counting method, and then develops a network to fuse the results of regression prediction and detection prediction. Experiments show that with the same model and parameters, the counting accuracy of the method proposed in this paper is much higher than any single model [4].

Crowd counting is a crucial computer vision task with numerous practical applications. Convolutional neural networks (CNNs) are a common foundation for crowd counting. This approach makes use of deep learning to automatically learn attributes from photos that can be used to gauge the crowd size. To forecast the density map of a given crowd image, a network is often trained as part of the CNN-based crowd-counting method. The crowd density at each spot in the image is continuously estimated by the density map. The density map is integrated over the full image to produce the final crowd count. The CNN architecture has undergone a number of tweaks and improvements to further demonstrate the method's robustness [5].

3. Analysis of Problem

Crowd counting and monitoring systems require meticulous consideration of various factors to effectively meet the needs of diverse applications. Firstly, accuracy is paramount, ensuring precise crowd size estimates and movement tracking. Real-time performance enables timely responses to dynamic situations, while scalability ensures the system can handle crowds of varying sizes and densities across different environments. Robustness is essential for coping with challenging conditions like lighting variations and occlusions, maintaining accurate counting under any circumstance. Adaptability ensures the system can accommodate different scenarios and behaviors, whether indoor, outdoor, or in transportation hubs. Data privacy and security measures are crucial, safeguarding individuals' identities and adhering to privacy regulations. User-friendly interfaces and visualization tools enhance usability, providing clear insights for decision-making. Integration with existing infrastructure allows seamless incorporation into operational systems, enhancing overall efficiency.

Maintenance and support ensure the system remains reliable and up-to-date, with regular updates and bug fixes. Lastly, ethical considerations, including fairness, accountability, and bias mitigation, are integral to responsible deployment. By considering these requirements, crowd counting and monitoring systems can be designed and developed to meet the specific needs of different stakeholders and applications, with input from end-users and domain experts ensuring tailored solutions for each deployment context.

4. Proposed Work

The data flow for crowd counting and monitoring involves acquiring data from various sources, preprocessing it to enhance quality, detecting individuals or objects of interest, extracting relevant features, and using crowd counting algorithms to estimate crowd size. Analyzed data is visualized for insights, aiding real-time decision-making for stakeholders like security personnel or event organizers. While the process follows a generalized overview, specific components and flow may vary based on system requirements and data characteristics.

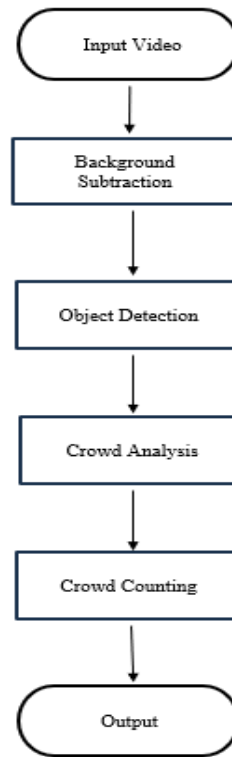


Fig. 1. Flowchart of System

5. Objective

Deep learning is revolutionizing crowd counting and monitoring by leveraging complex data to achieve high accuracy. Initially, data is collected from diverse sources and preprocessed for quality enhancement. Deep neural network architectures like Convolutional Neural Networks (CNNs) detect individuals in images or video frames, while Recurrent Neural Networks (RNNs) handle sequential data for temporal analysis. Hybrid architectures combine CNNs and RNNs for improved accuracy. Models are trained with labelled data using optimization algorithms like stochastic gradient descent. Evaluation metrics like mean absolute error assess model performance. Deployed models analyze live data for real-time insights into crowd size, density, and movement patterns.

6. System Requirement

- ◆ Cython
- ◆ numpy
- ◆ opencv-python
- ◆ torch
- ◆ matplotlib
- ◆ pillow
- ◆ tensorboard
- ◆ PyYAML
- ◆ torchvision
- ◆ scipy

7. Implementation

System implementation for crowd counting and monitoring involves several crucial steps, beginning with data collection. Diverse datasets of crowded scenes are gathered, ensuring coverage of various scenarios, lighting conditions, and crowd densities. Next, data preprocessing enhances image quality and uniformity through resizing, normalization, and augmentation, promoting dataset diversity.

Annotation is pivotal for training supervised learning models, providing ground truth counts for each image or video frame. Model selection involves choosing an appropriate deep learning architecture tailored for crowd counting, such as Convolutional Neural Networks (CNNs) or specialized architectures like CSRNet or HydraNet.

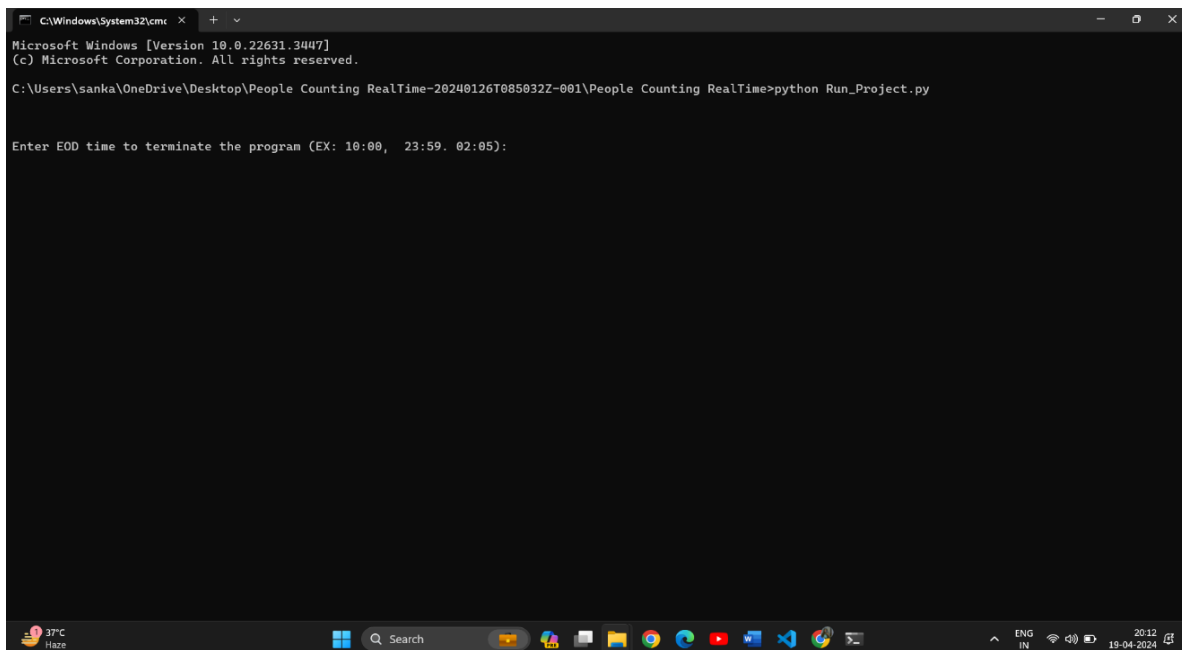
Training the selected model entails feeding it with input images and corresponding count labels, optimizing parameters to minimize the disparity between predicted and ground truth counts. Validation on a separate dataset assesses model performance on unseen data, with hyperparameter tuning if necessary. Testing on a distinct dataset evaluates the model's generalization ability.

Deployment of the trained model enables real-time crowd counting and monitoring, integrating it into applications or systems capable of processing live video feeds or static images. Continuous monitoring and evaluation ensure system performance, with user feedback driving improvements. Optimization techniques like model compression and deployment on specialized hardware enhance efficiency, scalability, and robustness.

System implementation for crowd counting and monitoring involves several crucial steps, beginning with data collection. Diverse datasets of crowded scenes are gathered, ensuring coverage of various scenarios, lighting conditions, and crowd densities. Next, data preprocessing enhances image quality and uniformity through resizing, normalization, and augmentation, promoting dataset diversity.

Integration with surveillance systems enhances situational awareness and decision-making, enriching existing infrastructure. Throughout the implementation process, considerations for privacy, ethics, and bias mitigation are paramount, particularly in surveillance applications. Regular updates and improvements maintain system effectiveness and relevance to evolving needs and challenges, ensuring sustained performance and utility.

User Interface of real time crowd counting:



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.22631.3447]
(c) Microsoft Corporation. All rights reserved.

C:\Users\sanka\OneDrive\Desktop\People Counting RealTime-20240126T085032Z-001\People Counting RealTime>python Run_Project.py

Enter EOD time to terminate the program (EX: 10:00, 23:59, 02:05):
```

Fig 2. Homepage

```

C:\Users\sanka\AppData\Local\Programs\Python\Python38\Lib\site-packages\torch\serialization.py:671: SourceChangeWarning: source code of class 'torch.nn.module.container.Sequential' has changed. you can retrieve the original source code by accessing the object's source attribute or set 'torch.nn.Module.dump_patches = True' and use the patch tool to revert the changes.
  warnings.warn(msg, SourceChangeWarning)
C:\Users\sanka\AppData\Local\Programs\Python\Python38\Lib\site-packages\torch\serialization.py:671: SourceChangeWarning: source code of class 'torch.nn.module.conv.Conv2d' has changed. you can retrieve the original source code by accessing the object's source attribute or set 'torch.nn.Module.dump_patches = True' and use the patch tool to revert the changes.
  warnings.warn(msg, SourceChangeWarning)
C:\Users\sanka\AppData\Local\Programs\Python\Python38\Lib\site-packages\torch\serialization.py:671: SourceChangeWarning: source code of class 'torch.nn.module.batchnorm.BatchNorm2d' has changed. you can retrieve the original source code by accessing the object's source attribute or set 'torch.nn.Module.dump_patches = True' and use the patch tool to revert the changes.
  warnings.warn(msg, SourceChangeWarning)
  return _wrapFunc(a, 'argmax', axis=axis, out=out, **kwargs)
  File "C:\Users\sanka\AppData\Local\Programs\Python\Python38\Lib\site-packages\numpy\core\fromnumeric.py", line 54, in _wrapfunc
    return _wrapit(obj, method, *args, **kwargs)
  File "C:\Users\sanka\AppData\Local\Programs\Python\Python38\Lib\site-packages\numpy\core\fromnumeric.py", line 43, in _wrapit
C:\Users\sanka\AppData\Local\Programs\Python\Python38\Lib\site-packages\torch\serialization.py:671: SourceChangeWarning: source code of class 'torch.nn.module.activation.LeakyReLU' has changed. you can retrieve the original source code by accessing the object's source attribute or set 'torch.nn.Module.dump_patches = True' and use the patch tool to revert the changes.
  warnings.warn(msg, SourceChangeWarning)
  result = getattr(asarray(obj), method)(*args, **kwargs)
ValueError: attempt to get argmax of an empty sequence
C:\Users\sanka\AppData\Local\Programs\Python\Python38\Lib\site-packages\torch\serialization.py:671: SourceChangeWarning: source code of class 'torch.nn.module.container.ModuleList' has changed. you can retrieve the original source code by accessing the object's source attribute or set 'torch.nn.Module.dump_patches = True' and use the patch tool to revert the changes.
  warnings.warn(msg, SourceChangeWarning)
C:\Users\sanka\AppData\Local\Programs\Python\Python38\Lib\site-packages\torch\serialization.py:671: SourceChangeWarning: source code of class 'torch.nn.module.pooling.MaxPool2d' has changed. you can retrieve the original source code by accessing the object's source attribute or set 'torch.nn.Module.dump_patches = True' and use the patch tool to revert the changes.
  warnings.warn(msg, SourceChangeWarning)
C:\Users\sanka\AppData\Local\Programs\Python\Python38\Lib\site-packages\torch\serialization.py:671: SourceChangeWarning: source code of class 'torch.nn.module.upsampling.Upsample' has changed. you can retrieve the original source code by accessing the object's source attribute or set 'torch.nn.Module.dump_patches = True' and use the patch tool to revert the changes.
  warnings.warn(msg, SourceChangeWarning)
1/1: 0... success (640x480 at 30.00 FPS).

19_April_2024 08:15:19.478071PM 0
19_April_2024 08:15:24.511877PM 0
19_April_2024 08:15:29.536502PM 0
    
```

Fig 3. Activity Page (Background Frame Counting)

Output Of Real Time Crowd Counting:

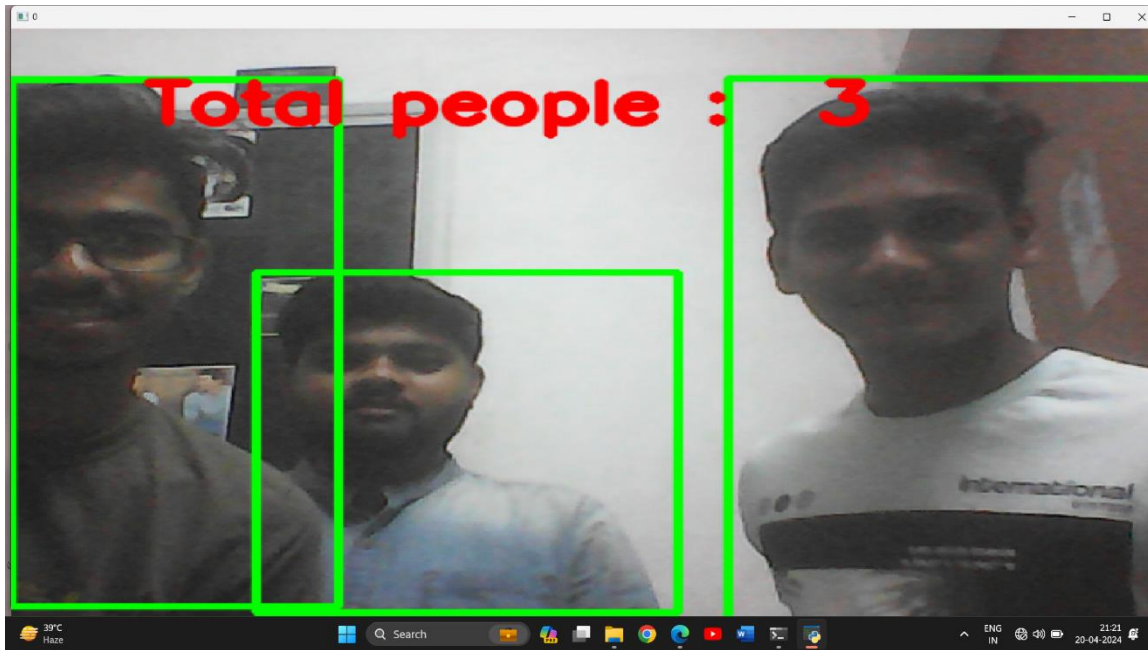


Fig 4 Real time crowd counting

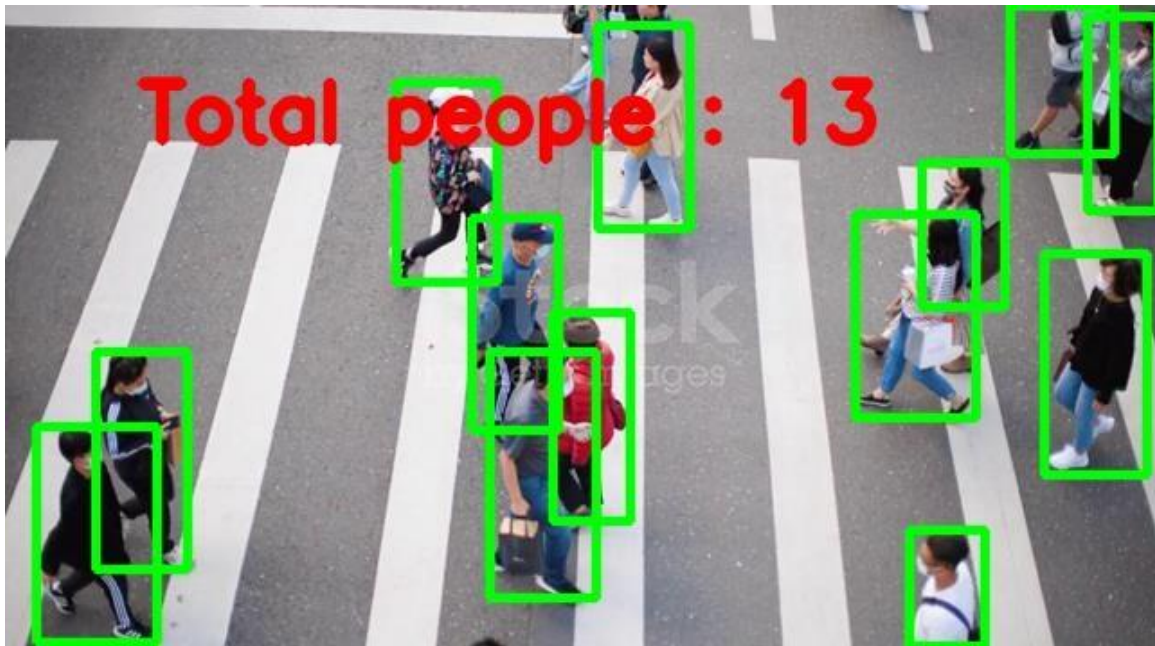
Result of recorded video crowd counting :

Fig 5 Recorded Video Crowd Counting

8. Conclusion

Overall, a Crowd Monitoring System offers several advantages, including improved public safety, efficient crowd management, quick response to emergencies, real-time data analysis, cost-effectiveness, and valuable insights. With careful consideration of the potential disadvantages, a Crowd Monitoring System can be a valuable tool for managing crowds and ensuring public safety.

REFERENCES

-
- [1] A. Ranjan, N. Pathare, S. Dhavale and S. Kumar, "Performance Analysis of YOLO Algorithms for Real-Time Crowd Counting," 2022 2nd Asian Conference on Innovation in Technology (ASIANCON), Ravet, India, pp. 1-8, 2022.
 - [2] A. Pandey, H. Kandpal, S. S. Rawat, H. Vaidya, R. Rawat and B. V. Kumar, "Deep Learning-Based Object Detection System for Crowd Monitoring in High-Security Areas," 2024 International Conference on Intelligent and Innovative Technologies in Computing, Electrical and Electronics (IITCEE), pp. 1-6, 2024.
 - [3] M. Abdou and A. Erradi, "Crowd Counting: A Survey of Machine Learning Approaches," 2020 IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIoT), Doha, Qatar, pp. 48-54, 2020.
 - [4] H. Wu, "High-Accuracy Crowd Counting Method Based on Mixed Labeled Dataset," 2021 IEEE 2nd International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE), Nanchang, China, pp. 829-833, 2021.
 - [5] P. Sivaprakash, M. Sankar, R. Chithambaramani and D. Marichamy, "A Convolutional Neural Network Approach for Crowd Counting," 2023 4th International Conference on Smart Electronics and Communication (ICOSEC), Trichy, India, pp. 1515-1520, 2023.