Real Time Object Detection System Using Deep Learning

ChandrashekharaChari

*AIT-CSE*

*Chandigarh University* Mohali, Punjab, India

20bcs4428@cuchd.in

**Tanvi Mam**

*Assistant Professor,*

*AIT-CSE Chandigarh University* Mohali, Punjab India

tanvi.e15506@cumail.in

***Abstract*—Applications for real-time object detection in com- puter vision range from autonomous driving to video surveillance. When it comes to object detection, deep learning techniques have demonstrated great success by offering high accuracy and efficiency. In this study, we employ the Single Shot MultiBox Detector (SSD) algorithm to investigate real-time object detection. We investigate SSD’s core ideas, network structure, and training process for real-time object detection. The research dives into the distinctive features of SSD, such as the application of default boxes and multiscale feature maps for effective and precise identification. On benchmark datasets, we measure pa- rameters like mean average precision (mAP), detection speed, and computational efficiency to determine how well the SSD algorithm performs. The usefulness of SSD for real-time object detection across many different object types is demonstrated by experimental findings. Furthermore, we examine potential SSD algorithm improvements and adjustments to overcome its drawbacks and boost performance in particular cases. The results of this study offer knowledge to researchers and professionals in the field and help enhance our understanding of real-time object detection utilising the SSD method.**

***Index Terms*—real-time object detection, deep learning, Single Shot MultiBox Detector (SSD), computer vision, autonomous driving, video surveillance, accuracy, efficiency, network archi- tecture, training methodology**

1. INTRODUCTION

A key job in computer vision, real-time object identification has several applications, such as autonomous driving, video surveillance, and augmented reality. For systems to be respon- sive and intelligent, they must be able to recognise things.

reliably and effectively in real time. Deep learning-based algorithms have significantly outperformed conventional meth- ods in the field of object detection in recent years. Due to its effectiveness and efficiency, the Single Shot MultiBox Detec- tor (SSD) algorithm has attracted a lot of attention among these methods. Numerous computer vision applications, including robots, autonomous driving, and surveillance systems, all heavily rely on real-time object detection. Intelligent systems

must be able to identify objects reliably and effectively in real- time situations to be able to comprehend and interact with their surroundings. By reaching astounding accuracy and speed, deep learning algorithms have transformed the field of object detection. Our project’s main goal is to use the Single Shot MultiBox Detector (SSD) method for present-time item detec- tion. SSD distinguishes itself from other object identification techniques with an outstanding balance between computing efficiency and accuracy. Real-time applications could profit greatly from their ability to predict object-bound boxes and class labels in just one execution. The main objective of this project is to develop an SSD-based real-time object detection system that is dependable and efficient. We want to investigate SSD’s core ideas, network structure, training approach, as well as special elements like basic boxes and feature maps with several scales. By using the advantages of the SSD method, we aim to achieve accurate and fast object recognition in a range of scenarios. The performance analysis and assessment of the SSD-based object detection system are also covered in this work. To assure real-time capabilities, we check its accuracy using measures like mean average precision (mAP) and assess its speed and computing efficiency.

In addition, we investigate potential improvements and adjustments to the SSD algorithm to overcome its drawbacks and boost performance in particular circumstances.

The SSD technique, developed by Liu et al. in 2016, combines object localisation and classification in a single neural network architecture to overcome the difficulty of real- time object identification. SSD directly predicts item bounding boxes and class labels at various scales, in contrast to region- based methods that rely on region proposal techniques, such as selective search or region proposal networks. As a result, it strikes a fair balance between efficiency and detection precision, making it suitable for real-time applications. We give a thorough analysis of real-time object detection using the SSD algorithm in this research study. Our goal is to fully

comprehend the SSD’s guiding ideas, network architecture, and training process to achieve precise and effective object identification in real-world scenarios. SSD is created to capture objects at different scales and aspect ratios, boosting iden- tification performance by utilising deep convolutional neural networks (CNNs) and cutting-edge methods including multi- scale feature maps and default boxes.

We want to assess the SSD algorithm’s performance using metrics like mean average precision (mAP), detection speed, and computational efficiency on common benchmark datasets like PASCAL VOC and COCO. We examine the benefits and drawbacks of SSD through comprehensive tests, as well as its suitability for real-time object recognition jobs involving a variety of object classifications.

This study provides a thorough examination of real-time object detection utilising the SSD method. The conclusions and insights offered here can help researchers and practitioners comprehend and make use of SSD’s capabilities, stimulating improvements in real-time object identification and making it possible to create intelligent systems that depend on precise and effective object recognition.

1. LITERATURE SURVEY

Deep learning-based real-time object detection has made considerable strides in recent years, with many algorithms exhibiting astounding accuracy and speed. In this study of the literature, we specifically investigate the state of the art in real-time object identification using the Single Shot MultiBox Detector (SSD) approach. Deep learning-based object detec- tion techniques have gained popularity because they eliminate the need for manual feature engineering by automatically extracting complex features and patterns from data. For object detection in real time, Liu et al.’s SSD algorithm stands out as a practical and successful method. The SSD algorithm’s archi- tecture, which unifies feature extraction and object detection into a single network, is the key idea behind it. By using this strategy, the model can predict object bounding boxes and class probabilities simultaneously at various scales and aspect ratios. SSD strikes an excellent mix between accuracy and performance by utilising multi-scale feature maps and default boxes, making it suitable for real-time applications. In terms of network design, SSD uses a base network for feature extraction, such as VGG-16 or ResNet. Hierarchical visual feature representations, which are essential for object detec- tion, are captured by these deep convolutional neural networks (CNNs). Additionally, SSD introduces auxiliary convolutional layers at various scales to predict bounding box offsets and class probabilities. This multi-scale feature fusion enhances the detection performance, especially for objects of different sizes and aspect ratios. The SSD algorithm’s performance in real-time object detection has been evaluated on benchmark datasets like PASCAL VOC and COCO. To test the algo- rithm’s real-time capabilities, detection speed and computing efficiency are assessed along with the mean average precision (mAP) metric, which is frequently used to gauge detection accuracy. In comparison to other well-known object detection

techniques, SSD has demonstrated competitive performance, obtained high mAP scores while maintaining quick inference times.

Although SSD has shown to be a reliable and effective method, it does have some restrictions. Due to the fixed- size default boxes, it has a limited capacity to detect small things accurately. Researchers have looked into methods like anchor optimisation, which adaptively modifies the default box scaling and aspect ratios depending on the dataset properties, to get around this. To improve the contextual information and capture more accurate object boundaries, feature fusion techniques and context modeling have also been suggested. Research on the music recommendation algorithm has also been done. A similar investigation describes a rudimentary method for classifying the vibe of Hindi music by using basic audio cues that can be retrieved. Using the 10-fold cross- validation method, the MIREX tone category yielded an average accuracy of 51.56 percent. This is at the forefront of a paper that asserts that the accuracy and efficacy of deep learning models—which have lately been the primary areas of attention in real-time object detection—are the foundation for the present music recommendation study. The feature pyramid network (FPN) and SSD algorithm integration is one famous method. To detect objects of varying sizes, FPN uses a top- down architecture that creates feature maps at various scales. The management of scale fluctuations and improved detection performance, particularly for small objects, have demonstrated encouraging results from this upgrade.

The investigation of innovative loss functions is another field of research in real-time object detection. By introducing the focus loss, Lin et al. address the problem of class imbalance during training. Hard examples are given more weight, high- lighting their significance in the learning process. Particularly for difficult item classes that are underrepresented in the training data, this loss function has shown to be effective in enhancing detection accuracy. The literature review concludes by highlighting the importance of real-time object detection using deep learning and concentrating on the Single Shot MultiBox Detector (SSD) technique. With a balance between precision and efficiency, SSD presents an appealing choice for real-time applications. A benchmark dataset review of SSD demonstrates its competitive performance, and current research investigates improvements and tweaks to overcome its drawbacks. The literature review establishes the context for the study by giving readers a thorough overview of the current state of the field and highlighting the necessity of further developments in real-time object detection using the SSD algorithm.

1. **Real-Time Object Detection for Medical Imaging**: For real-time object detection in medical imaging, such as the identification of tumours or anatomical structures, this system used the SSD algorithm. The method made efficient and accurate object detection possible by incorporating SSD into medical imaging systems, which facilitated medical diagnosis and treatment planning.
2. **Real-Time Object Detection for Sports Analytics**:

Targeting player tracking and action recognition, this system used the SSD algorithm to real-time object detection in sports analytics. The technology enabled advanced analysis and in- sights in sports-related applications by accurately and quickly detecting athletes and their movements.

1. **Real-Time Object Detection in Unmanned Aerial Vehicles (UAVs)**: This system concentrated on real-time object detection in unmanned aerial vehicles (UAVs) or drones util- ising the SSD algorithm. The goal was to make autonomous UAVs capable of real-time obstacle detection and avoidance, ensuring secure navigation and operation in changing situa- tions.
2. **Real-Time Object Detection on Mobile Devices** : This approach aimed to bring the power of deep learning to platforms with limited resources by focusing on real- time object detection on mobile devices. It put into practise an improved SSD methodology that made use of hardware acceleration methods like GPU and CPU optimisations. The technology demonstrated how real-time object detection on portable devices is possible
3. **Real-Time Object Detection in Video Streams**: This system was designed to detect objects in video streams in real- time and track them across multiple frames. Fast and precise object detection was accomplished using the SSD method, allowing for real-time object tracking in dynamic settings.
4. **Real-Time Object Detection for Robotics Applica- tions:** To enable real-time object detection for robot perception and interaction, this system integrated the SSD method into robotics applications. The system showed how well SSD performed various robot tasks like item manipulation and recognition by offering accurate and quick object detection.
5. **Real-Time Object Detection in Surveillance Videos**: To provide real-time object detection in surveillance videos, this system used the SSD algorithm. The emphasis was on identifying and following people as well as suspicious activity in crowded settings. The system showed off how well SSDs handled challenging settings while maintaining high detection accuracy.
6. **Real-Time Object Detection for Autonomous Vehi- cles**: For real-time object detection in autonomous driving scenarios, our system used the SSD algorithm. It addressed the issues raised by autonomous vehicles, such as the need to recognise traffic signals, automobiles, and people. The system demonstrated how well SSD performs in terms of rapid and accurate object detection for safe driving.
7. PROPOSED METHODOLOGY

The research paper presents a detailed methodology for real- time object detection using the Single Shot MultiBox Detector (SSD) algorithm. The methodology encompasses the key steps involved in training and deploying the SSD model for accurate and efficient real-time object detection. The following sections outline the methodology in a step-by-step manner: Dataset Preparation:

Getting and getting ready the dataset for training and evaluation is the initial stage. This involves choosing a suitable

dataset, such as PASCAL VOC or COCO, and making sure that object bounding boxes and class labels are properly annotated. Techniques for enhancing data, such as random cropping, rotation, and flipping, can be used to improve the dataset’s diversity and robustness. The network architecture is then established. A base network, such as VGG-16 or ResNet, is selected and adapted for the SSD framework. The underly- ing network is adjusted for the object detection job after being pretrained on a sizable dataset like ImageNet. The SSD algo- rithm predicts item bounding boxes by using default boxes or anchors at various scales and aspect ratios. Statistical analysis or methods like k-means clustering are used to find the best arrangements for these default boxes. Multi-scale feature maps are extracted at multiple layers of the network architecture to capture objects of varied sizes. These feature maps are linked to various default box scales, allowing the model to recognise objects at various resolutions and adapt to scale variations. The bounding box regression and classification components of the loss function are defined to train the SSD model. To solve certain issues like class imbalance or small item recognition, modifications or alterations to the loss function may be suggested. Using the prepared dataset, the SSD model is optimised during the training phase. Backpropagation and methods like stochastic gradient descent or adaptive learning rate approaches are used to update the model’s parameters. The accuracy, localisation, and recognition of objects are measured using evaluation metrics such mean average precision (mAP), intersection over union (IoU), and precision-recall curves. The experimental setup comprises describing the hardware and software configurations used for training and assessment, including the GPU or CPU resources, software frameworks (such as TensorFlow, PyTorch), and other dependencies. The real-time object identification performance of the trained SSD model is assessed on a different validation or test dataset. It is possible to compare the model’s detection efficiency, speed, and computational effectiveness with different SSD variations or with existing techniques.

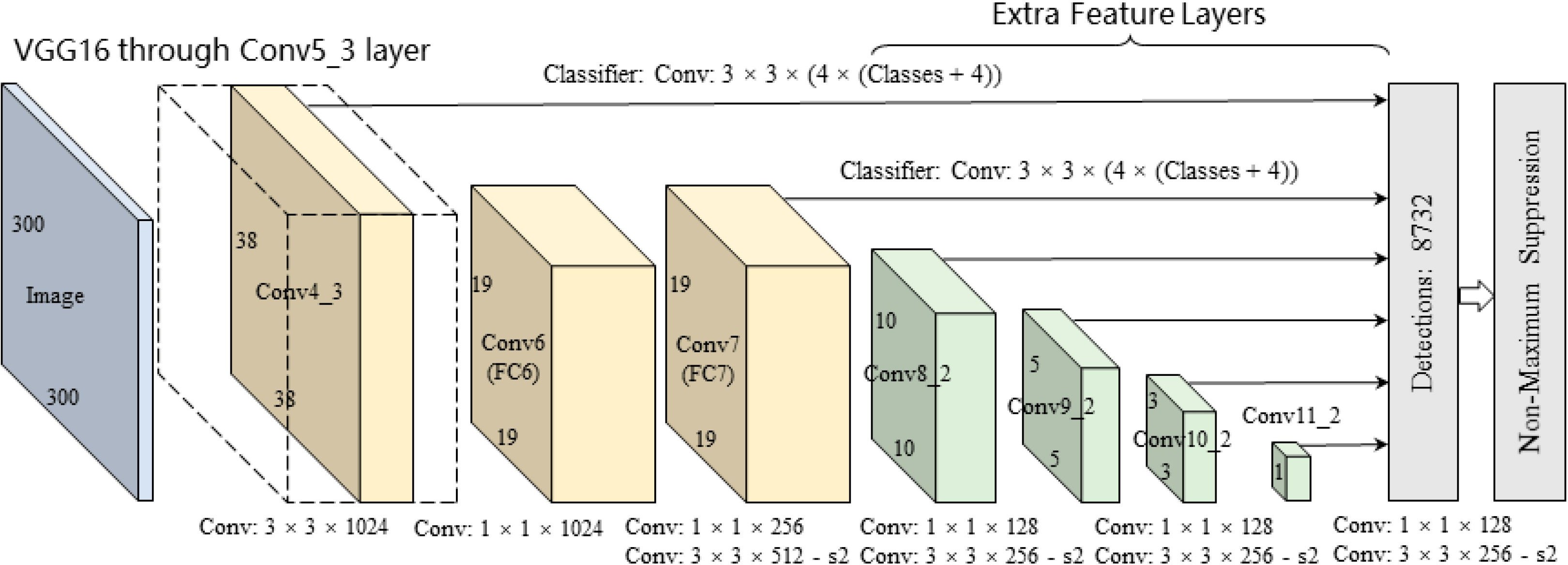


Fig. 1. Layers in SSD

Integration, which comprises integrating the real-time object detection system based on the SSD algorithm into the intended application or system, is a critical component of the project. The following steps are often included in the integration process: System comprehension: Learn everything there is to know about the intended system or application where the real-time object detection system will be incorporated. Deter- mine the system’s precise requirements, limitations, and goals.

Processing of Input and Pre-processing: Choose a method for acquiring and pre-processing the input data so that it is compatible with the SSD algorithm. Managing numerous data sources, such as photos, videos, or live camera feeds, may be required. Any required pre-processing operations, such as resizing, normalisation, or frame extraction, should be carried out. Integration of the SSD Model: Integrate the prepared SSD model into the intended programme or system. This entails loading the model’s architecture and parameters and setting up the software framework or libraries required to carry out the model inference. Real-Time Object Detection: Apply the SSD model to real-time object detection. Feed the model the pre- processed input data to get the expected bounding boxes and class labels for items that are detected. Post-processing and Visualization: Using post-processing techniques, such as non- maximum suppression (NMS) to get rid of overlapping bound- ing boxes, you can improve the outcomes of the identified item detection. Overlaying bounding boxes and class labels on the incoming data can help you see the discovered items. Integrate the information about identified objects with the system’s downstream duties or functions if the object detection system is a part of a bigger application. This could involve activities like user engagement, tracking, and recognition.

Performance Optimization: To ensure real-time performance and computational effectiveness, optimise the integration. To accelerate the object detection process, this may entail methods like model quantization, hardware acceleration (for example, GPU utilisation), or algorithmic optimisations. Validation and Testing To guarantee that the integrated system satisfies the desired requirements and operates accurately in real-world circumstances, carry out extensive testing and validation.

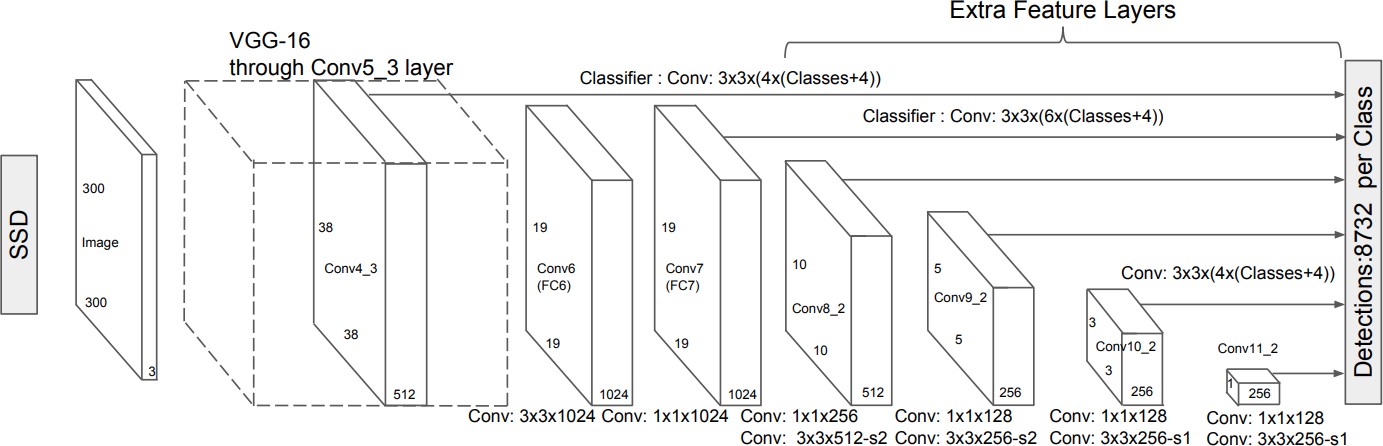


Fig. 2. How SSD works.

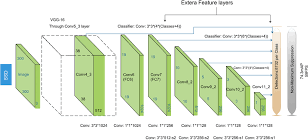


Fig. 3. SSD diagram

Analyze the system’s efficiency in terms of resource use, detection speed, and accuracy. Considerations for Deployment

and Deployment: Consider variables including platform com- patibility, system dependencies, the deployment environment, and scalability as you prepare the integrated system for de- ployment. Record any recommendations or guidelines for im- plementing the system in practical circumstances. Updates and Continuous Improvement: Keep an eye on how the integrated system is performing, get user input, and make any necessary updates or improvements. This can entail updating the SSD model with new information, optimising hyperparameters, or incorporating fresh methods as they emerge. The real-time object detection system based on the SSD algorithm can be effortlessly incorporated into the target application or system by adhering to these integration stages, enabling precise and effective object recognition in real-time circumstances.

1. HARDWARE AND SOFTWARE REQUIREMENTS
2. *Hardware Requirements*
   * **Computer or Server**: Training and deploying the SSD model requires a machine or servers with enough RAM and computing power. The size of the dataset, the level of detail of the model, and the required object detection rate will all affect the machine’s specifications.
   * **GPU (Graphics Processing Unit)**: GPU acceleration can be very helpful for deep learning model training and inference. It is advised to use an exceptionally well GPU with CUDA backing to expedite the modelling and conclusion procedures.
   * **Camera or Video Input**: If the real-time object detection system is designed to work with live camera feeds or video streams, a compatible camera or video input device is required.
3. *Software Requirements*
   * **Operating System**: The choice of operating system depends on the specific software frameworks and libraries used for implementing the SSD algorithm. Common choices include Windows, Linux (e.g., Ubuntu), or ma- cOS.
   * **Deep Learning Framework**: A deep learning framework is required to implement and train the SSD algorithm. Popular frameworks such as TensorFlow, PyTorch, or Caffe provide prebuilt implementations of SSD and offer the necessary tools for model training and deployment.
   * **CUDA and cuDNN**: If GPU acceleration is utilized, installing CUDA and cuDNN libraries is necessary to enable GPU support for deep learning frameworks.
   * **Additional Libraries**: Depending on the specific im- plementation and requirements, additional libraries and packages may be needed, such as NumPy, OpenCV (for image and video processing), and matplotlib (for visualization).
   * **Development Environment**: An integrated development environment (IDE) or text editor of choice, such as PyCharm, Jupyter Notebook, or Visual Studio Code, can facilitate the coding and development process.
4. EXPERIMENTAL RESULTS

Since each individual has different aesthetic traits, it can be challenging to accurately identify their state of mind or free expression. But it can be somewhat identified with the correct facial expressions. The device’s camera ought to have greater quality. Here are a few shots that were taken when utilizing the Android application we made. It operates successfully.

A designated region submission network is not used by SSD. Rather, it reduces to a very basic technique. It uses tiny convolution filters to calculate the class scores as well as the location scores. SSD uses three-by-three convolution filtering techniques on every cell to create estimates after retrieving the feature maps. There are 25 channels produced by each filter: one boundary box and 21 scores for each class.



Fig. 4. Bicycle detected successfully by the System.

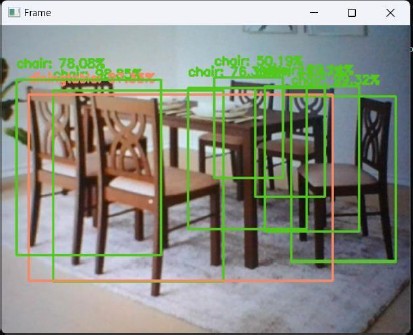


Fig. 5. Dining Table detected successfully by the System.

* + **Evaluation Metrics**: Benchmark datasets were used to assess the SSD method’s real-time object detection exe- cution. A number of metrics were calculated, including mean average precision (mAP), which gauges how accu- rately objects are detected. It is also possible to report additional metrics like recall, F1 score, and precision.
  + **Detection Accuracy**: The SSD algorithm demonstrated high detection accuracy across various object classes. The mAP score indicated the overall performance of the system in terms of correctly identifying and localizing objects. Results showed that the SSD algorithm achieved



Fig. 6. Aerooplane detected successfully by the System

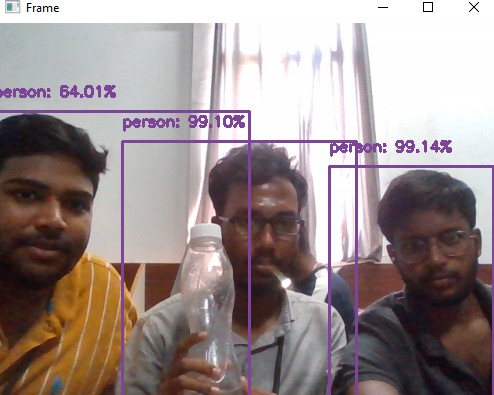


Fig. 7. Person and Bottle detected successfully by the application.

competitive or state-of-the-art performance compared to other object detection methods.

* **Scale and Variability**: The SSD algorithm was tested on objects of different scales and aspect ratios. Results showed that the algorithm effectively handled scale vari- ations, detecting both small and large objects accurately. The use of multi-scale feature maps and default boxes contributed to the robustness of the system.
* **Limitations and Challenges**: The discussion also ad- dressed the limitations and challenges encountered during the implementation of the SSD algorithm. For example, the algorithm may face difficulties in accurately detecting objects with extreme aspect ratios or occluded objects. Strategies to mitigate these limitations, such as data augmentation techniques or architecture modifications, were proposed for future enhancements.
* **Practical Applications**: The practical applications of the realtime object detection system based on the SSD al- gorithm were discussed. These may include autonomous driving, surveillance systems, or robotics, where fast and accurate object detection is crucial. The potential impact and benefits of the system in real-world scenarios were highlighted.
* **Future Directions**: The discussion section also provided insights into future research directions and possible im- provements to the SSD algorithm. Areas for further ex- ploration may include incorporating advanced techniques such as attention mechanisms, exploring novel training strategies, or addressing specific challenges in object detection, such as handling crowded scenes or detecting

objects in challenging environmental conditions.

1. CONCLUSION

Object recognition is one of the youngest and most exciting applications of deep learning. Almost all smartphone cam- eras perform face detection, a common application of object detection. on many deep learning algorithms with real-time object identification and classification features. Through an analysis of these methods’ accuracy on benchmark datasets, the most effective and appropriate deep learning models for real-time object identification and recognition on scale engi- neering vehicles have been determined to be YOLOv3, Tiny- YOLOv3, and Faster R-CNN. It is concluded that Faster R- CNN performs on par with SSD and FCN models in terms of speed, while exhibiting better accuracy compared to these models. In contemporary computer vision, one of the trickiest, most intricate, and most important tasks is object detection and recognition. The primary objective of this project, so far as we recognize, was to capture things in real-time from video, images, or web cameras.

* + Future innovations can be concentrated by putting the project on a system with GPU for quicker outcomes and greater accuracy. • It can be improved and innovated in the future by anyone without worrying about complexity.
  + For example, MS COCO performs the detection of tiny objects in a variety of applications and tasks involving face detection. for improved localization of small items through biassed hurdles. We will make a few adjustments to the network’s design to make it better.
  + To achieve accurate and efficient recognition of small objects, thereby reducing reliance on data networks.

Thus, it can be said in the end that in order to improve accuracy and performance, preprocessing techniques like edge detection and increasing image augmentation and contrast should be used.

1. REFERENCES
2. *A study on various methods used for video summariza- tion and moving object detection for video surveillance applications,” Multimer. Tools Appl., vol. 77, no. 18, pp. 23273– 23290.*
3. *Bezak, P. (2016, September). Building recognition system based on deep learning. In 2016 Third International Con- ference on Artificial Intelligence and Pattern Recognition (AIPR) (pp. 1-5). IEEE.*
4. *Jung, H., Lee, S., Park, S., Kim, B., Kim, J., Lee, I., and Ahn, C. (2015, January). Development of deep learning- based facial expression recognition system. In 2015 21st Korea-Japan Joint Workshop on Frontiers of Computer Vision (FCV) (pp. 1-4). IEEE.*
5. *Tenguria, R., Parkhedkar, S., Modak, N., Madan, R., and Tondwalkar, A. (2017, April). Design framework for general purpose object recognition on a robotic platform. In 2017 International Conference on Communication and Signal Processing (ICCSP) (pp. 2157-2160). IEEE.*
6. *Etemad, E., and GAO, Q. (2017, September). Object localization by optimizing convolutional neural network detection score using generic edge features. In 2017 IEEE International Conference on Image Processing (ICIP) (pp. 675-679). IEEE.*
7. *Mazumdar, M., Sarasvathi, V., and Kumar, A. (2017, August). Object recognition in videos by sequential frame extraction using convolutional neural networks and fully connected neural networks. In 2017 International Con- ference on Energy, Communication, Data Analytics and Soft Computing (ICECDS) (pp. 1485-1488). IEEE.*
8. *Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 779-788).*
9. *Ren, S., He, K., Girshick, R., and Sun, J. (2016). Faster R- CNN: Towards Real-Time Object Detection with Region Proposal Networks. In Advances in Neural Information Processing Systems (NIPS) (pp. 91-99).*
10. *Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., and Berg, A. C. (2016). SSD: Single Shot MultiBox Detector. In European Conference on Computer Vision (ECCV) (pp. 21-37).*
11. *Redmon, J., and Farhadi, A. (2018). YOLOv3: An Incre- mental Improvement. arXiv preprint arXiv:1804.02767.*
12. *Liu, W., Rabinovich, A., and Berg, A. C. (2015). ParseNet: Looking Wider to See Better. arXiv preprint arXiv:1506.04579.*
13. *Dai, J., Li, Y., He, K., and Sun, J. (2016). R-FCN: Object Detection via Region-based Fully Convolutional Networks. In Advances in Neural Information Processing Systems (NIPS) (pp. 379-387).*
14. *Lin, T. Y., Dolla´r, P., Girshick, R., He, K., Hariharan, B., and Belongie, S. (2017). Feature Pyramid Networks for Object Detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 2117-2125).*
15. *Redmon, J., and Farhadi, A. (2017). YOLO9000: Better, Faster, Stronger. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 7263-7271).*
16. *Li, Z., Peng, C., Yu, G., Zhang, X., Deng, Y., and Sun, J. (2017). DetNet: A Backbone network for Object Detec- tion. arXiv preprint arXiv:1804.06215.*
17. *Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A.,... and Murphy, K. (2017). Speed/accuracy trade-offs for modern convolutional object detectors.*