International Research Journal in Global Engineering and Sciences. (IRJGES)
ISSN : 2456-172X | Vol. 6, No. 1, March, 2021 – May,2021 | Pages 30-37

IMAGE SEGMENTATION WITH MACHINE LEARNING

Revant Chillpuri (Student) Dr. Thanveer Jahan Dr. C. N. Ravi
Department of Computer Science and Engineering, Vaagdevi College of Engineering

Abstract
Real-time object detection is a vast and sophisticated area of computer vision aimed at identifying and recognizing objects. Object detection utilizes OpenCV (Open Source Computer Vision), a library of programming functions primarily designed for real-time computer vision in digital images and videos. Visually challenged individuals face difficulty distinguishing objects around them. The main objective of real-time object detection is to assist the blind in overcoming this challenge. Real-time object detection finds applications in various fields such as object tracking, video surveillance, pedestrian detection, people counting, self-driving cars, face detection, ball tracking in sports, and more. This is accomplished using Convolutional Neural Networks (CNN), a key tool in Deep Learning. This project serves as an assisting tool for visually challenged individuals. Keywords: "Convolutional Neural Network", "OpenCV", "Deep Learning", "AI", "SSD".

1. INTRODUCTION

Object detection is a technology used to detect various objects in digital images and videos. It is particularly useful in applications such as self-driving cars and face detection, where objects need to be continuously monitored. The algorithm employed for object detection in this project is Convolutional Neural Networks (CNN), a class of Deep Learning. This utilizes the MobileNet SSD technique, where MobileNet is a neural network used for image classification and recognition, and SSD is a framework used to achieve the multi-box detector. The combination of MobileNet and SSD enables object detection. The main advantage of utilizing Deep Learning is that it eliminates the need for feature extraction from data compared to traditional machine learning methods.
Haar-like features play a crucial role in object detection in images. They scan the entire image, comparing every small box with trained data to identify even small-detailed objects present within the images.

1.1 HARDWARE REQUIREMENTS:

Processor: Intel i3 and above RAM: 4GB and Higher Hard Disk: Minimum 500GB

1.2 SOFTWARE REQUIREMENTS:

Programming Language / Platform: Python IDE: PyCharm / Jupyter

1.3 EXISTING SYSTEM:

Blind individuals lead normal lives with their own methods, but they face challenges due to inaccessible infrastructure and social obstacles. The biggest challenge for a blind person, especially those with complete vision loss, is navigation in unfamiliar places. While blind

individuals navigate easily within their homes, finding objects around them presents difficulties. Therefore, we decided to develop a REAL-TIME OBJECT DETECTION System after reviewing papers in this area, aiming to assist blind individuals in recognizing objects in real-time environments.

1.4 PROPOSED SYSTEM:

Deep learning, a subset of machine learning and artificial intelligence (AI), involves networks capable of learning from unstructured or unlabeled data. The approach utilized in this project is Convolutional Neural Networks (CNN), which employs Haar-cascade classifiers for object detection.

CNN:

The Convolutional Neural Network (CNN) is a specialized type of neural network model designed for working with two-dimensional image data, although it can be used with one-dimensional and three-dimensional data as well.

At the heart of the convolutional neural network lies the convolutional layer, which is responsible for performing the operation known as "convolution".

In the context of a convolutional neural network, convolution is a linear operation that involves the multiplication of a group of weights with the input, much like in a standard neural network. However, since this technique is designed for two-dimensional input, the multiplication occurs between an array of input data and a two-dimensional array of weights, referred to as a filter or kernel.

The filter is smaller than the input data, and thus the type of multiplication applied between a filter-sized patch of the input and the filter is a scalar product. A scalar product involves element-wise multiplication between the filter-sized patch of the input and the filter, resulting in a single value when summed. Due to this single resulting value, the operation is conventionally denoted as the "scalar product".

The intentional use of a smaller filter compared to the input allows the same filter (set of weights) to be multiplied by the input array multiple times at different points across the input. Specifically, the filter is systematically applied to every overlapping part or filter-sized patch of the input data, moving from left to right and top to bottom.

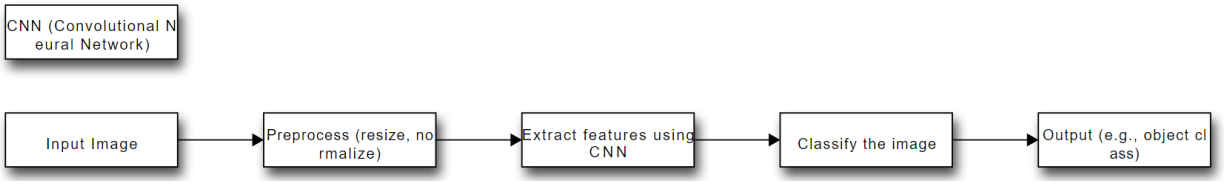
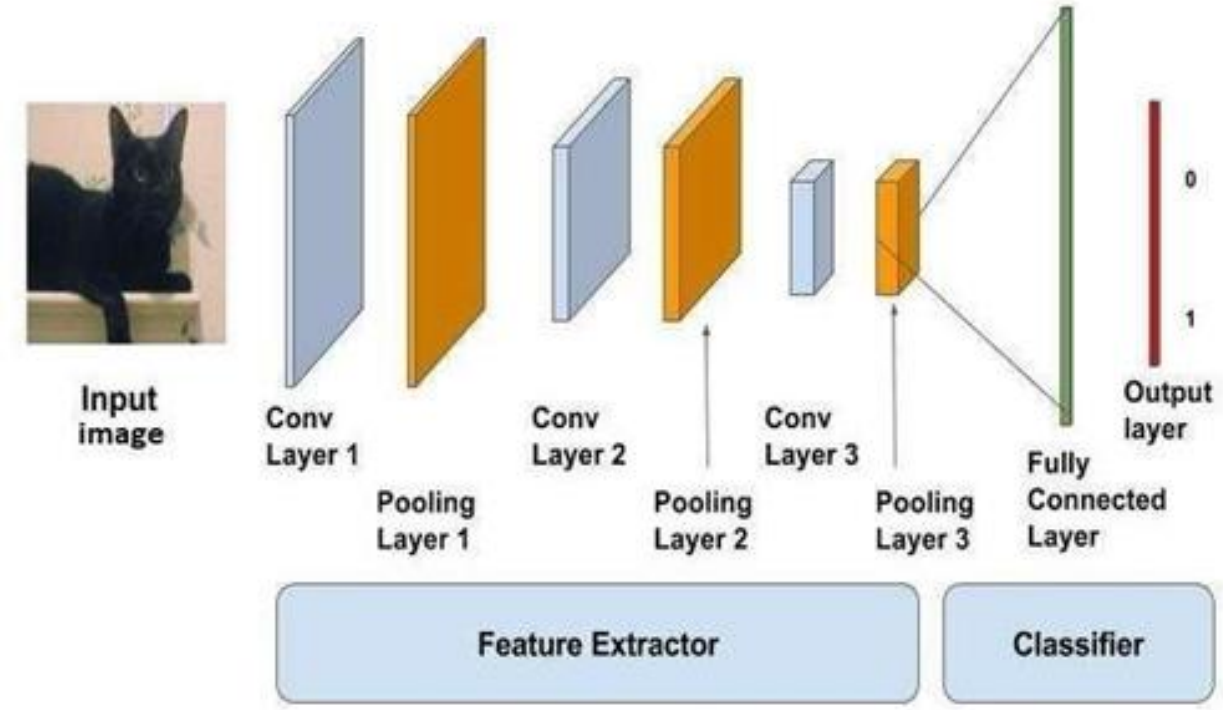


Fig 1.1 Sample block diagram indicating the flow of image processing using CNN

The systematic application of an identical filter across an image is a powerful concept. If the filter is designed to detect a specific type of feature within the input, then applying that filter systematically across the entire input image allows the filter to potentially identify that feature anywhere within the image.

This capability is commonly referred to as translation invariance, meaning that the overall concern lies in whether the feature is present rather than where exactly it is located within the image.



1.5 OpenCV:

OpenCV, short for Open Source Computer Vision, is a collection of libraries in Python designed for image manipulation tasks such as image scaling. It facilitates the development of real-time computing applications and focuses on image processing, video capture, and analysis. OpenCV offers various features including face and object detection and supports multiple programming languages like C++, Python, and Java. It is compatible with different platforms such as Windows, Linux, OS X, and Android.

1.6 Training the Dataset:

The dataset refers to a collection of data, which can include images, alphabets, numbers, documents, and files. In object detection, the dataset comprises images of the objects to be identified, with multiple images typically available for each object to improve accuracy. It is essential for the data in the dataset to be labeled. Generally, there are three types of datasets: training, validation, and testing. The training dataset, which contains around 85-90% of the labeled data, is used to train the machine and obtain the model. The validation dataset, comprising approximately 5-10% of the labeled data, is used for validation purposes. The testing dataset evaluates the performance of the machine.

1.7 Developing a Real-Time Object Detector:

To develop a real-time object detector using deep learning and OpenCV, it is necessary to access the webcam effectively and apply object detection to each frame. OpenCV must be installed on the system, along with the deep neural network module. The following steps outline the process:

1. Import necessary packages, including VideoStream, FPS, numpy, argparse, imutils, time, and cv2.
2. Construct and parse the argument to provide paths to the Caffe prototxt file and pre-trained model, as well as set the minimum probability threshold for weak detections.
3. Initialize class labels and corresponding random colors for detected objects.
4. Load the model using the provided prototxt and model files.
5. Read the video stream and set the frames per second using imutils.
6. Process each frame by providing it as input to the model, which detects objects and surrounds them with rectangular boxes while displaying their labels.
7. Observe the output video stream with detected objects, rather than the input stream.

We start by looping over our detections, bearing in mind that multiple objects can be detected in a single image. We also apply a check to the confidence (i.e., probability) associated with each detection. If the confidence is high enough (i.e., above the threshold), then we display the prediction in the terminal as well as draw the prediction on the image with text and a colored bounding box.
The remaining steps in the frame capture loop involve (1) displaying the frame, (2) checking for a quit key, and (3) updating our frames per second counter.

Real-time Deep Learning Object Detection Results
To witness our real-time deep-learning-based object detector in action, ensure you download the example code + pre-trained Convolutional Neural Network from the "Downloads" section of this guide.
Next, open up a terminal and execute the following command:
python real_time_object_detection.py \ --prototxt MobileNetSSD_deploy.prototxt.txt \ --model MobileNetSSD_deploy.caffemodel
You should see the output video frame with any detected objects if OpenCV can access your webcam. Below are sample results of applying deep learning object detection to an example video:

Figure 1: A short clip of real-time object detection with deep learning and OpenCV + Python

TESTING

In this project, system testing was conducted to ensure that the object detection was performed according to the pre-set requirements. Various test cases were developed for different classes of objects to ensure that the system was effectively detecting objects and producing satisfactory output.

TABLE 1

Testing For Object Detection

Sample Input	**Expected Output**	**Obtained Output**	**Confidence Score**	**Status**
Video of a person	Person detected	Person	0.98	Success
Video of a chair	Chair detected	Chair	0.96	Success
Video of a background	Background detected	Background	0.95	Success
Video of an aeroplane	Aeroplane detected	Aeroplane	0.90	Success
Video of a bicycle	Bicycle detected	Bicycle	0.63	Success
Video of a bottle	Bottle detected	Bottle	0.96	Success
Video of a car	Car detected	Car	0.85	Success
Video of a cat	Cat detected	Cat	0.95	Success
Video of a dog	Dog detected	Dog	0.91	Success
Video of a cow	Cow detected	Cow	0.92	Success
Video of a horse	Horse detected	Horse	0.86	Success
Video of a bus	Bus detected	Bus	0.97	Success
Video of a boat	Boat detected	Boat	0.90	Success
Video of a sheep	Sheep detected	Sheep	0.87	Success
Video of a bird	Bird detected	Bird	0.91	Success
Video of a motorbike	Motorbike detected	Motorbike	0.94	Success
Video of a potted plant	Potted plant detected	Potted plant	0.97	Success
Video of a train	Train detected	Train	0.95	Success
Video of a TV	TV detected	TV	0.98	Success

2. RESULT

In this paper, we considered detecting around 15 to 20 different objects during the training phase. Some of these objects include 'person', 'car', 'train', 'bird', 'sofa', 'dog', 'plant', 'aeroplane', 'bicycle', 'bus', 'motorbike', etc.

The output of this project displays the detected objects with a rectangular bounding box around each object, along with a label indicating its name and the confidence score of the detection. The system can detect any number of objects present in a single image with a high level of accuracy.

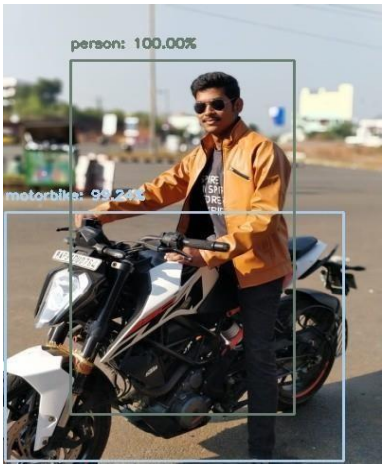


Fig 1

fig 2

Fig 3

Figure 5

3. APPLICATIONS

Here are some future implementations of object detection:

3.1 Face Detection and Recognition: Face detection is a separate category of object detection. Applications like Facebook and FaceApp utilize face detection and recognition technology. This is evident in everyday scenarios such as unlocking mobile phones and enhancing security systems.

3.2 Object Tracking: Object detection is employed in tracking objects, such as monitoring a person's movements or tracking a ball in sports like football or cricket. This tracking capability enhances user experience and provides additional insights, especially in sports analysis.

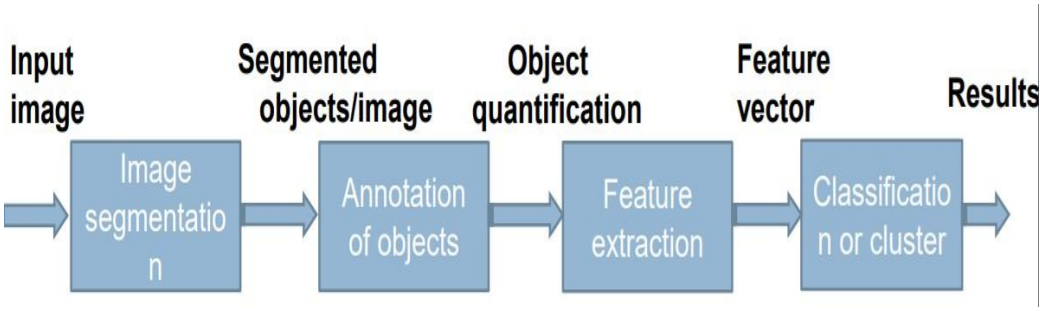
3.3 Self-Driving Cars: Object detection is crucial for the development of self-driving cars. It enables vehicles to autonomously navigate, make decisions, and detect obstacles in their environment, ensuring safety and efficiency.

3.4 Emotion Detection: Emotion detection allows systems to recognize the emotions expressed on a person's face. Companies like Apple have explored this technology to detect user emotions and translate them into corresponding emojis on smartphones.

3.5 Biometric Identification through Retina Scan: Retina scanning, based on iris code, is a highly accurate and unique biometric identification technique used in high-security systems.

3.6 Smart Text Search and Selection (Google Lens): Applications like Google Lens can recognize text and images, enabling users to search for relevant information effortlessly using their smartphones.

SYSTEM ARCHITECTURE



CONCLUSION

Deep learning based object detection has emerged as a significant research area in recent years. This project focuses on generic object detection pipelines, which serve as foundational architectures for various related tasks. By combining object detection with deep learning and OpenCV, along with efficient, threaded video streams, the project successfully achieves common tasks such as object detection, face detection, and pedestrian detection.

However, it's important to note that the accuracy of object detection can be affected by factors such as camera sensor noise and lighting conditions. These variables may introduce challenges in accurately recognizing objects. Despite this, the project demonstrates efficient execution using CPUs instead of GPUs, which are typically required for such tasks.

Object detection algorithms effectively combine image classification and object localization. They analyze input images and produce output with bounding boxes corresponding to detected objects, along with class labels. This approach provides valuable information about the position, height, and width of identified objects, facilitating various applications in computer vision and beyond.

REFERENCES

[1] Geethapriya S, N. Duraimurugan, S.P. Chokkalingam, "Real-Time Object Detection with YOLO," International Journal of Engineering and Advanced Technology (IJEAT).

[2] Abdul Vahab, Maruti S Naik, Prasanna G Raikar, Prasad S R4, "Applications of Object Detection Systems," International Research Journal of Engineering and Technology (IRJET).

[3] Hammad Naeem, Jawad Ahmad, Muhammad Tayyab, "Real-Time Object Detection and Tracking," IEEE.

[4] Meera M K, Shajee Mohan B S, "Object Recognition in Images," International Conference on Information Science (ICIS), 2016.

[5] Astha Gautam, Anjana Kumari, Pankaj Singh, "The Concept of Object Recognition," International Journal of Advanced Research in Computer Science and Software Engineering, vol. 5, no. 3, March 2015.

[6] Joseph Redmon, Santosh Divvala, Ross Girshick, "You Only Look Once: Unified, Real-Time Object Detection," The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 779-788.

[7] V. Gajjar, A. Gurnani, Y. Khandhediya, "Human Detection and Tracking for Video Surveillance: A Cognitive Science Approach," in 2017 IEEE International Conference on Computer Vision Workshops.