Object Detection Using Deep Learning Algorithm CNN

S. Sumahasan[1], Udaya Kumar Addanki[2], Navya Irlapati[3], Amulya Jonnala[4]
[1,2]Assistant Professor, [3,4]Student

Department of Computer Science and Engineering

G.V.P.C.E.W, Visakhapatnam, India. #9966003543

Abstract
Object Detection is an emerging technology in the field of Computer Vision and Image Processing that deals with detecting objects of a particular class in digital images. It has considered being one of the complicated and challenging tasks in computer vision. Earlier several machine learning-based approaches like SIFT (Scale-invariant feature transform) and HOG (Histogram of oriented gradients) are widely used to classify objects in an image. These approaches use the Support vector machine for classification. The biggest challenges with these approaches are that they are computationally intensive for use in real-time applications, and these methods do not work well with massive datasets. To overcome these challenges, we implemented a Deep Learning based approach Convolutional Neural Network (CNN) in this paper. The Proposed approach provides accurate results in detecting objects in an image by the area of object highlighted in a Bounding Box along with its accuracy.

Keywords:
Object detection, CNN, SIFT, HOG

1. Introduction

Humans are very good at interpreting unstructured data, such as images and audio files. One of the best things about the rise of deep learning is that computers are much better at interpreting unstructured data compared to a few years ago. It creates opportunities for many new exciting applications like Object detection, Speech recognition, image recognition. Object Detection deals with identifying individual objects in an image along with its location.

Object Detection has two parts-Object Classification and Object Localization. Object Classification deals with classifying the object into one of the pre-defined classes. Object Localization deals with distinguishing the object along with its location.

The main idea of Object detection systems is to construct a model for an object class from a set of training examples. The idea is to provide a training data set and test the input image by comparing the objects in the images with the training dataset containing images of objects. The output displayed with the objects detected after comparing the input image with the training dataset.

Manuscript received May 5, 2024
Manuscript revised May 20, 2024
https://doi.org/10.22937/IJCSNS.2024.24.5.15

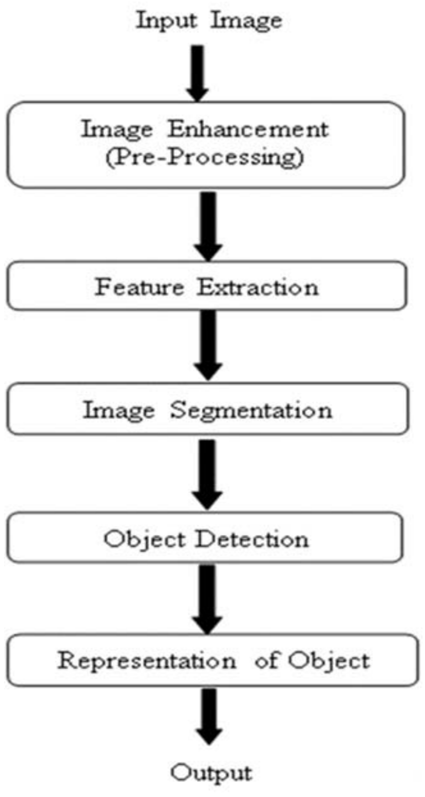


Fig 1: Model Diagram

As shown in **fig 1**, Object detection system takes an image as input. In the pre-processing stage, the image is resized and converted into a matrix. The system extract features from an image to recognize a particular object. It classifies the object to a finite set of classes. Also, it predicts the location of the object in terms of a bounding box.

Object Detection task can be done either by using machine learning or deep learning-based approaches. Some of them are SIFT (Scale-invariant feature transform) and HOG (Histogram of oriented gradients), which uses SVM (Support Vector Machine) for classification. All these previous algorithms share a common problem that they do not work well with more massive datasets. To overcome the problems, Convolutional Neural Network (CNN), a deep learning-based algorithm, has been used in this paper.

The contents of the paper are structured as follows- Section 2 Provides a brief overview of the proposed system. Section 3 Gives details about experimental results and analysis. Section 4 describes the conclusion of the paper.

2. Proposed Algorithm

Convolutional Neural Network used in our model. Convolutional Neural Network (CNN)[1] is a Deep Learning based algorithm that can take images as input, assign classes for the objects in the image. It differentiates one from the other. The pre-processing in a ConvNet is much lower when compared to other classification algorithms. ConvNet is used to reduce the images without losing features, which helps in getting the right prediction. It can successfully capture the Spatial and Temporal dependencies in an image through the application of relevant filters. This model is chosen due to three main advantages, namely, good detection accuracy, fewer training parameters, and runs on minimum computation hardware.

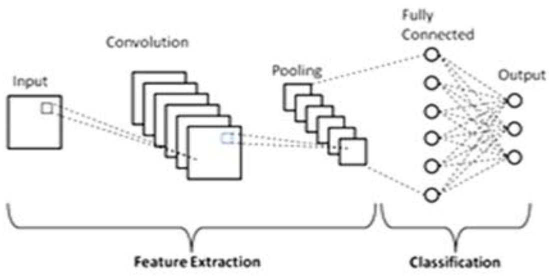


Fig 2: CNN Architecture

Fig 2 shows the architecture of the CNN Algorithm. The process of each layer explained as follows:

A. **Input Layer:** The process of training data starts with an input image provided by the user. As shown in **Fig 3,** the input image converted into a matrix, which may be of the form 128 x 128 x 3 where,3 represents RGB, which are the three-color planes - Red, Green, and Blue.[1]

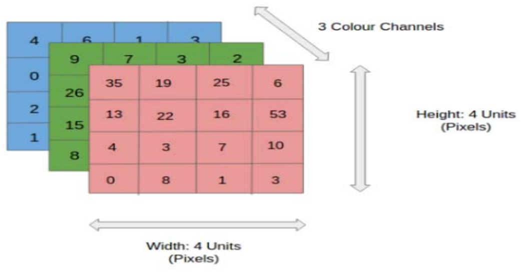


Fig 3: RGB Matrix

B. Convolutional Layer: The objective of the Convolution Layer is to extract features such as edges, color, gradient orientation from the input image. The convolutional operation is carried out with the help of an element called kernel/filter, as shown in **Fig 4** [1]. If the image has multiple channels (RGB), the Kernel has the same depth as that of the input image. In our program, we have implemented the function Conv2D() as an image is nothing but a 2-dimensional array. We can use Convolution 3D if we need to implement videos, where the third dimension is time. The convolution operation results in a feature map.

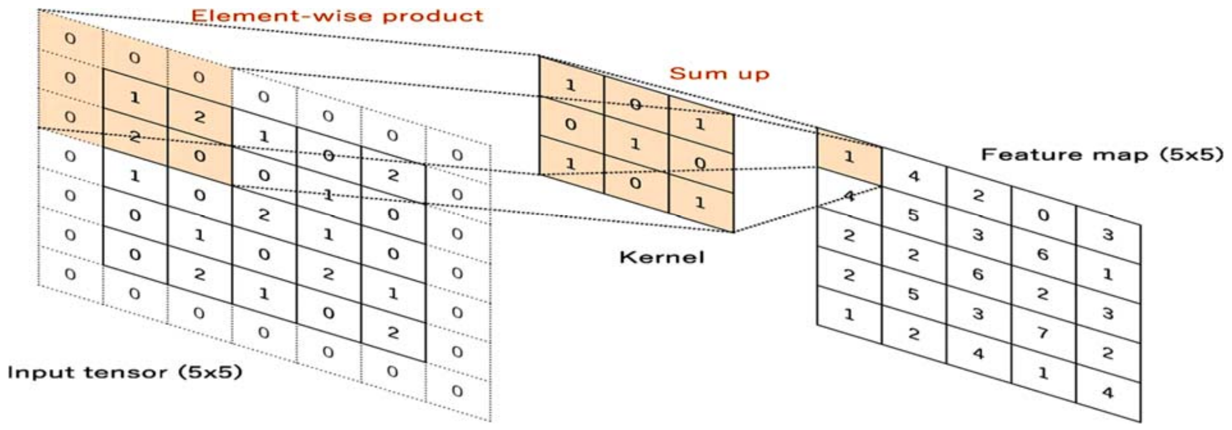


Fig 4: Feature Extraction (multiplying input matrix with filters)

C. ReLU Activation Function: ReLU stands for the Rectified Linear Unit for a non-linear operation. The output is ***f(x) = max (0, x).*** It is applied to the resultant feature map to convert the negative values into positive (0), as shown in **Fig 5**.[2].

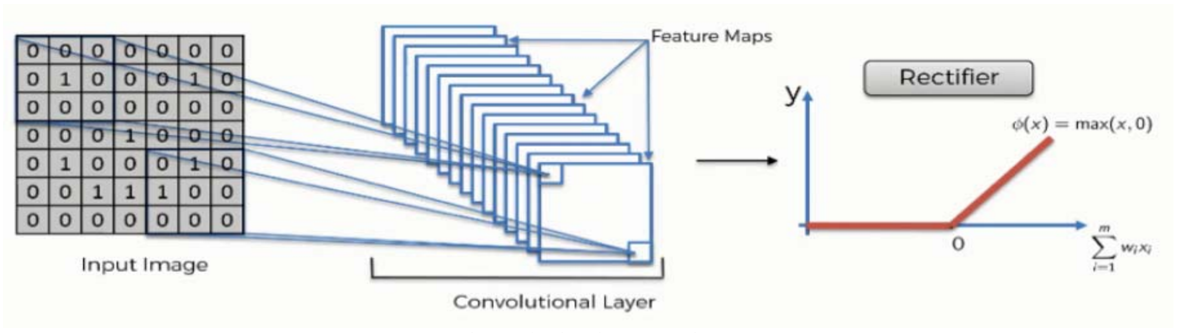


Fig 5: Activation Function

D. Pooling Layer: The different kinds of pooling layers are max pooling, min pooling, mean pooling, average pooling. We have used max-pooling because we need the maximum value pixel from the region of interest, as shown in **fig 6**. We have implemented the MaxPooling2D() function as we want to achieve minimum pixel loss and reduce the complexity of the model without reducing the performance. Max Pooling discards the noisy activations altogether and also performs noise reduction along with dimensionality reduction. The process of convolution-pooling is executed 3 times in our program [2][3].

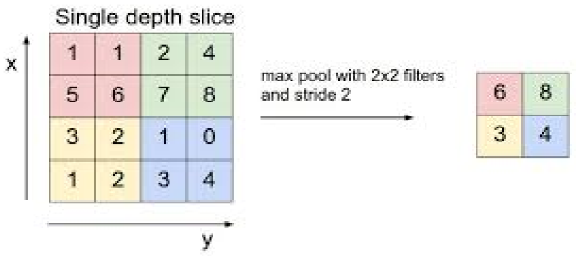


Fig 6: Max Pooling

E. Flattening: As shown in **Fig 7**, Flattening is a process of converting the resulting 2-dimensional arrays of the convolution-pooling operations into a single long continuous linear vector. To perform the process of flattening, we implement the flatten () function. We flatten our matrix into a vector and feed it into a fully connected layer like a neural network. [4]

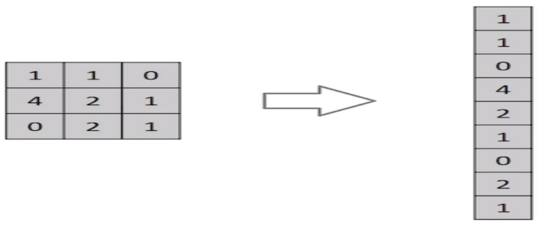


Fig 7: Flattening

F. Fully Connected Layer: The fully connected layer takes the flatted vector and uses them to classify the image into a label and passes it to the next layer, as shown in **Fig 8**. With the fully connected layers, we combined these features to create a model. After features extracted by the convolution layers and decreased by the pooling layers, these reduced images are mapped with the subgroup of a fully connected layer to the final output of the model, such as the probability of each class in classification tasks. The final output of a fully connected layer has the same number of output nodes as the number of classes [1][5].

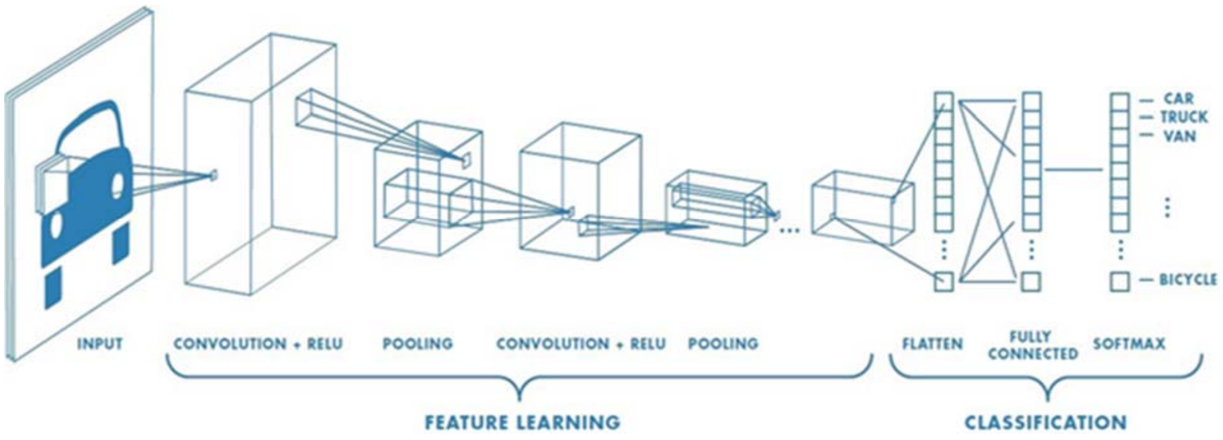


Fig 8:Block Diagram

G. SoftMax: An activation function applied to the multiple class classification task is a SoftMax function. The main goal of this function is to normalize the output values of the last fully connected layer to target class probabilities, where each value ranges between 0 and 1, as shown in **Fig 9**. In a series of epochs, the model can differentiate between dominating and certain low-level features in images and classify them using the SoftMax activation function [4].

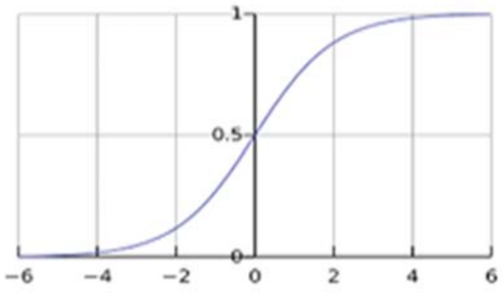


Fig 9: Softmax Activation

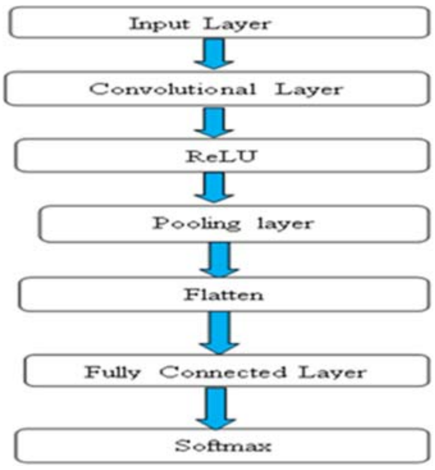


Fig 10: Flow Chart of CNN Algorithm

Dataset: The dataset used in our model contains 2500 images belonging to 5 different classes, which are Car, Cycle, Motorbike, Dog, Flowers. The dataset has divided into train sets and test sets. To train our model, we use images in train Sets. The Test sets contain the images to test our model prediction.

3. Experimental Results and Analysis

We have implemented PyQt5 for the frontend interface. The PyQt5 designer makes it easy to develop complex GUI apps in a short time. As shown in **fig 11**, the user can choose between building the model with the help of the CNN algorithm and applying object detection on an input image.

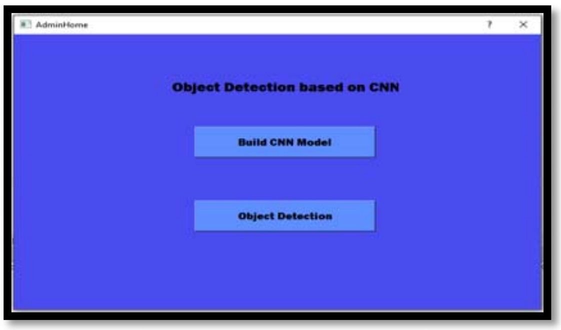
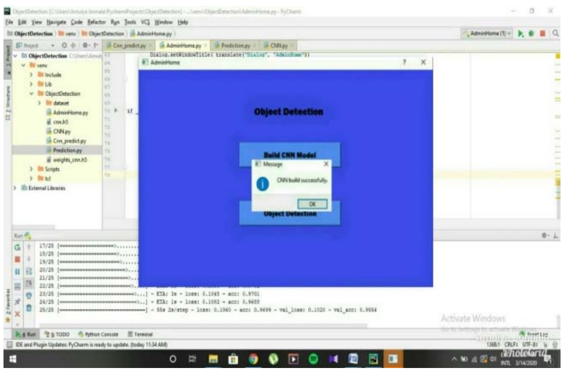


Fig 11: User Interface

When the user chooses to build a CNN model, the given dataset trained according to the CNN algorithm, we have implemented 5 datasets or classes. The datasets downloaded from TensorFlow Datasets, which is a collection of datasets ready to use. There are 2500 images found belonging to 5 classes, which are car, cycle, dog, flower, motorbike. The training process consists of 50 epochs with 25 epochs or steps per epoch, i.e., a loop within a loop.

After the CNN model built, the message "CNN built successfully" is displayed. The accuracy value can be seen in **fig 12**; the values of loss=0.1060, acc=0.9699, val_loss=0.1020 and val_acc=0.9554 are displayed. Here, loss and acc are applied to training data, while val_loss and val_acc applied to validation data.



```
23/25 [========================>...] - ETA: 1s - loss: 0.1045 - acc: 0.9701
24/25 [==========================>..] - ETA: 1s - loss: 0.1082 - acc: 0.9689
25/25 [==============================] - 55s 2s/step - loss: 0.1060 - acc: 0.9699 - val_loss: 0.1020 - val_acc: 0.9554
```

Fig 12: Accuracy of Model

When the user wants to perform object detection, the user can provide/select the input image, as shown **fig. 13**. Once the user clicks on the detect button, the input image is compared to the trained dataset/model and returns the results.

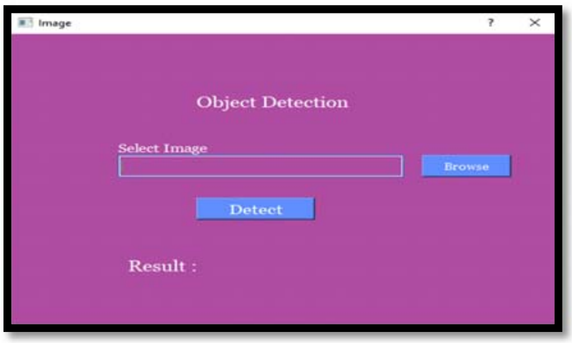


Fig 13: Uploading Image

The results of object detection, as shown in **fig 14**, displays the input image selected/provided by the user along with the name of the object detected as a result and the accuracy of the object detected. We have implemented matplotlib to plot the bounding box required for object localization.

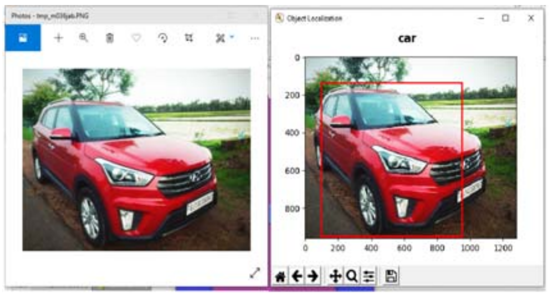


Fig 14(a): Output of Image1

Result: car and Accuracy: 100%

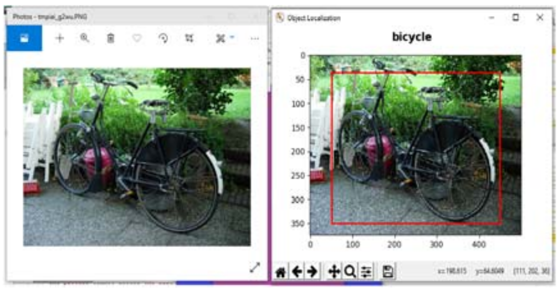


Fig 14(b): Output of Image2

Result: bicycle and Accuracy: 99%

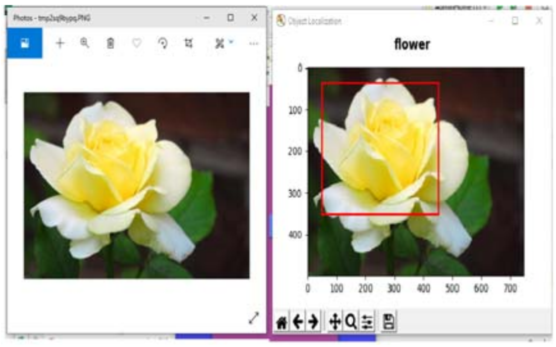


Fig 14(c): Output of Image3

Result: Flower and Accuracy: 97%

4. Conclusion

This paper proposes a new way to recognize and detects objects in an image. This proposed approach can be useful for the system, which seeks Object Detection as a prime feature. The proposed system CNN gives accurate results in detecting objects in an image along with its location. CNN is an efficient way to work with more massive datasets. Our model achieves an accuracy of 96% in detecting objects. In the future, work can be extended by detecting multiple objects and detecting objects that usually considered as background with high accuracy.

References

[1] Rikiya Yamashita1,2 & Mizuho Nishio1,3 & Richard Kinh Gian Do2 & Kaori Togashi1 "Convolutional neural networks: an overview and application in radiology" Insights into Imaging (2018) 9:611–629, https://doi.org/10.1007/s13244-018-0639-9P.
[2] Yasaka K, Akai H, Abe O, Kiryu S (2018) Deep learning with convolutional neural network for differentiation of liver masses at dynamic contrast-enhanced CT: a preliminary study. Radiology 286:887–896.
[3] Liu F, Jang H, Kijowski R, Bradshaw T, McMillan AB (2018) Deep learning MR imaging-based attenuation correction for PET/MR imaging. Radiology 286:676–684.
[4] Simonyan K, Zisserman A (2015) Very deep convolutional networks for large-scale image recognition. arXiv.
[5] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, Y. LeCun OverFeat: integrated recognition, localization, and detection using convolutional networks. arXiv:1312.6229, 2014.