GALGOTIAS UNIVERSITY

(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

# IMAGE RECOGNITION AND OBJECT DETECTION

**A Report for the Evaluation 3 of Project 2**


***Submitted by***

**APOORV**

**BISHNOI**

**(1613105021)**


***in partial fulfilment for the award of the degree***

***of***

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE AND ENGINEERING WITH SPECIALIZATION OF CLOUD COMPUTING AND VIRTUALIZATION**

**SCHOOL OF COMPUTING SCIENCE AND ENGINEERING**

**Under the Supervision of**

**Dr. NARESH KUMAR, M.Tech., Ph.D.,**
**Professor**

**APRIL / MAY- 2020**

GALGOTIAS UNIVERSITY

# SCHOOL OF COMPUTING AND SCIENCE AND ENGINEERING

## BONAFIDE CERTIFICATE

Certified that this project report **"IMAGE RECOGNITION AND OBJECT DETECTION"** is the bonafide work of "**APOORV BISHNOI (1613105021)"** who carried out the project work under my supervision.

**SIGNATURE OF HEAD**

Dr. MUNISH SHABARWAL,
PhD (Management), PhD (CS)
**Professor & Dean,**
**School of Computing Science & Engineering**

**SIGNATURE OF SUPERVISOR**

Dr. NARESH KUMAR, M.Tech., Ph.D.
**Professor**
**School of Computing Science & Engineering**

TABLE OF CONTENTS



# Abstract

Computer Vision is the branch of the science of computers and software systems which can recognize as well as understand images and scenes. Computer Vision is consists of

various aspects such as image recognition, object detection, image generation, image super-resolution and many more. Object detection is widely used for face detection, vehicle detection, pedestrian counting, web images, security systems and self-driving cars. In this project, we are using highly accurate object detection-algorithms and methods such as R-CNN, Fast-RCNN, Faster-RCNN, RetinaNet and fast yet highly accurate ones like SSD and YOLO. Using these methods and algorithms, based on deep learning which is also based on machine learning require lots of mathematical and deep learning frameworks understanding by using dependencies such as TensorFlow, OpenCV, ImageAI etc, we can detect each and every object in image by the area object in an highlighted rectangular boxes and identify each and every object and assign its tag to the object.

# List of Figures



# Introduction

A few years ago, the creation of the software and hardware image processing systems was mainly limited to the development of the user interface, which most of the programmers of each firm were engaged in. The situation has been significantly changed with the advent of the Windows operating system when the majority of the developers switched to solving the problems of image processing itself. However, this has not yet led to the cardinal progress in solving typical tasks of recognizing faces, car numbers, road signs, analyzing remote and medical images, etc. Each of these "eternal" problems is solved by trial and error by the efforts of numerous groups of the engineers and scientists. As modern technical

solutions are turn out to be excessively expensive, the task of automating the creation of the software tools for solving intellectual problems is formulated and intensively solved abroad. In the field of image processing, the required tool kit should be supporting the analysis and recognition of images of previously unknown content and ensure the effective development of applications by ordinary programmers. Just as the Windows toolkit supports the creation of interfaces for solving various applied problems.

Object recognition is to describe a collection of related computer vision tasks that involve activities like identifying objects in digital photographs. Image classification involves activities such as predicting the class of one object in an image. Object localization is refers to identifying the location of one or more objects in an image and drawing an abounding box around their extent. Object detection does the work of combines these two tasks and localizes and classifies one or more objects in an image.

When a user or practitioner refers to the term “object recognition“, they often mean “object detection“. It may be challenging for beginners to distinguish between different related computer vision tasks. So, we can distinguish between these three computer vision tasks with this example:

Image Classification: This is done by Predict the type or class of an object in an image.

Input: An image which consists of a single object, such as a photograph.

Output: A class label (e.g. one or more integers that are mapped to class labels).

Object Localization: This is done through, Locate the presence of objects in an image and indicate their location with a bounding box.

Input: An image which consists of one or more objects, such as a photograph.

Output: One or more bounding boxes (e.g. defined by a point, width, and height).

Object Detection: This is done through, Locate the presence of objects with a bounding box and types or classes of the located objects in an image.

Input: An image which consists of one or more objects, such as a photograph.

Output: One or more bounding boxes (e.g. defined by a point, width, and height), and a class label for each bounding box.

One of the further extension to this breakdown of computer vision tasks is object segmentation, also called “object instance segmentation” or “semantic segmentation,” where instances of recognized objects are indicated by highlighting the specific pixels of the object instead of a coarse bounding box. From this breakdown, we can understand that object recognition refers to a suite of challenging computer vision tasks. For example, image classification is simply straight forward, but the differences between object localization and object detection can be confusing, especially when all three tasks may be just as equally referred to as object recognition.

Humans can detect and identify objects present in an image. The human visual system is fast and accurate and can also perform complex tasks like identifying multiple objects and detect obstacles with little conscious thought. The availability of large sets of data, faster GPUs, and better algorithms, we can now easily train computers to detect and classify multiple objects within an image with high accuracy. We need to understand terms such as object detection, object localization, loss function for object detection and localization, and finally explore an object detection algorithm known as “You only look once” (YOLO).

Image classification also involves assigning a class label to an image, whereas object localization involves drawing a bounding box around one or more objects in an image. Object detection is always more challenging and combines these two tasks and draws a bounding box around each object of interest in the image and assigns them a class label. Together, all these problems are referred to as object recognition.

Object recognition refers to a collection of related tasks for identifying objects in digital photographs. Region-based Convolutional Neural Networks, or R-CNNs, is a family of techniques for addressing object localization and recognition tasks, designed for model performance. You Only Look Once, or YOLO is known as the second family of techniques for object recognition designed for speed and real-time use

# Background

The aim of object detection is to detect all instances of objects from a known class, such as people, cars or faces in an image. Generally, only a small number of instances of the object are present in the image, but there is a very large number of possible locations and scales at which they can occur and that need to somehow be explored. Each detection of the image is reported with some form of pose information. This is as simple as the location of the object, a location and scale, or the extent of the object defined in terms of a bounding box. In some other situations, the pose information is more detailed and contains the parameters of a linear or non-linear transformation. For example for face detection in a face detector may compute the locations of the eyes, nose and mouth, in addition to the bounding box of the face. An example of a bicycle detection in an image that specifies the locations of certain parts is shown in Figure 1. The pose can also be defined by a three-dimensional transformation specifying the location of the object relative to the camera. Object detection systems always construct a model for an object class from a set of training examples. In the case of a fixed rigid object in an image, only one example may be needed, but more generally multiple training examples are necessary to capture certain aspects of class variability.

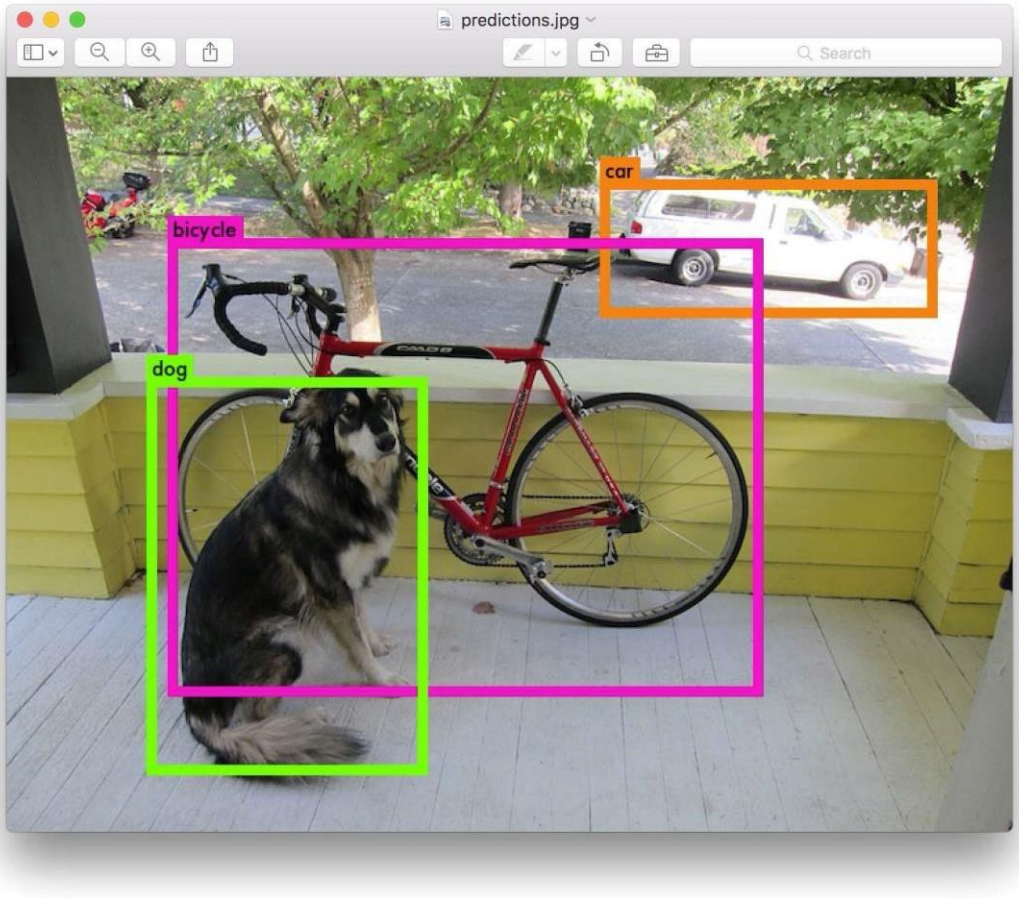


Fig 1:YOLO Object Detection

Convolutional implementation of the sliding windows Before we discuss the implementation of the sliding window using convents, let us analyze how we can convert the fully connected layers of the network into convolutional layers. Fig. 2 shows a simple convolutional network with two fully connected layers each of shape .

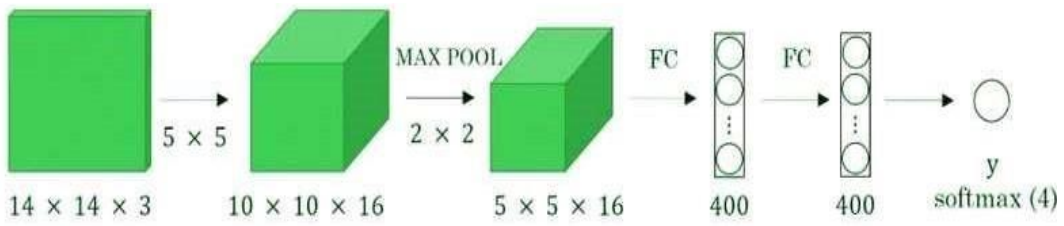


Figure 2: Simple convolution network

A fully connected layer can be converted to a convolutional layer with the help of a 1D convolutional layer. The width and height of this layer is equal to one and the number of filters are equal to the shape of the fully connected layer. An example of this is shown in Fig 3.

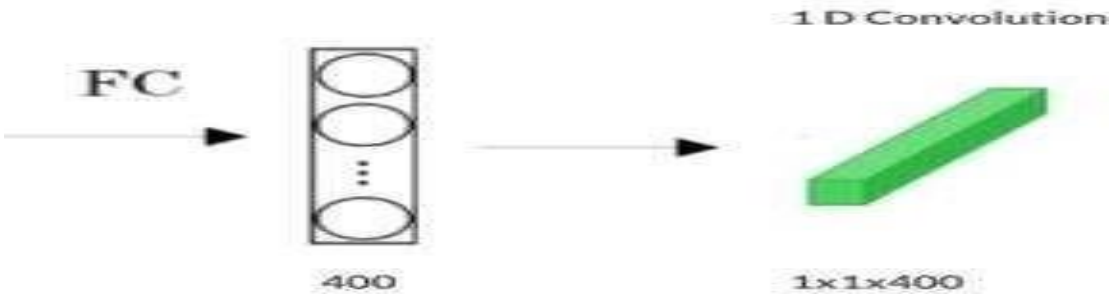


Figure 3: Conversion of fully connected layer to convolutional layer

We can apply the concept of conversion of a fully connected layer into a convolutional layer to the model by replacing the fully connected layer with a 1-D convolutional layer. The number of filters of the 1D convolutional layer is equal to the shape of the fully

connected layer. This representation is shown in Fig 4. Also, the output softmax layer is also a convolutional layer of shape (1, 1, 4), where 4 is the number of classes to predict.

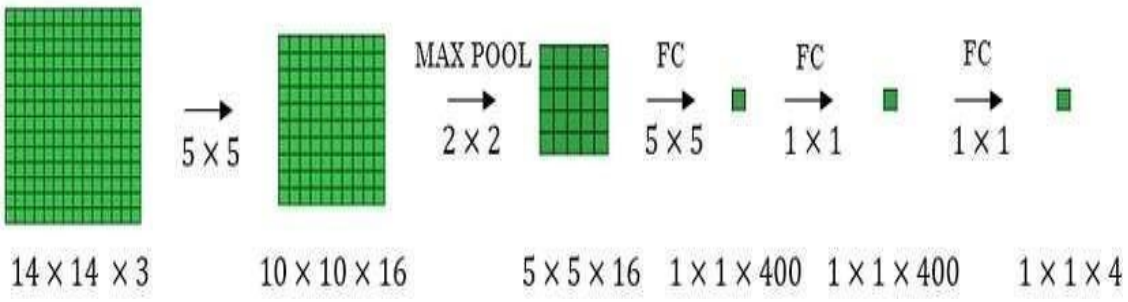


Figure 4: Replacing fully connected layer with 1D convolutional layer

Now, let's extend the above approach to implement a convolutional version of the sliding window. First, let us consider the ConvNet that we have trained to be in the following representation (no fully connected layers).

Let’s assume the size of the input image to be 16 × 16 × 3. If we are using the sliding window approach, then we would have passed this image to the above ConvNet four times, where each time the sliding window crops the part of the input image matrix of size 14 × 14 × 3 and pass it through the ConvNet. But instead of this, we feed the full image (with shape 16 × 16 × 3) directly into the trained ConvNet (see Fig. 6). This results will give an output matrix of shape 2 × 2 × 4. Each cell in the output matrix represents the result of the possible crop and the classified value of the cropped image. For example, the left cell of the output matrix(the green one) in Fig. 6 represents the result of the first sliding window. The other cells in the matrix represent the results of the remaining sliding window operations.

The stride of the sliding window is decided by the number of filters used in the Max Pool layer. In the example above, the Max Pool layer has two filters, and for the result, the sliding window moves with a stride of two resulting in four possible outputs to the given input. The main advantage of using this technique is that the sliding window runs and computes all values simultaneously. Consequently, this technique is really fast. The weakness of this technique is that the position of the bounding boxes is not very accurate. A better algorithm that tackles the issue of predicting accurate bounding boxes while using the convolutional sliding window technique is the YOLO algorithm. YOLO stands for you only look once which was developed in 2015 by Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. It is popular because it achieves high accuracy while running in real-time. This algorithm requires only one forward propagation pass through the network to make the predictions. This algorithm divides the image into

grids and then runs the image classification and localization algorithm (discussed under object localization) on each of the grid cells. For example, we can give an input image of size 256 × 256. We place a 3 × 3 grid on the image.

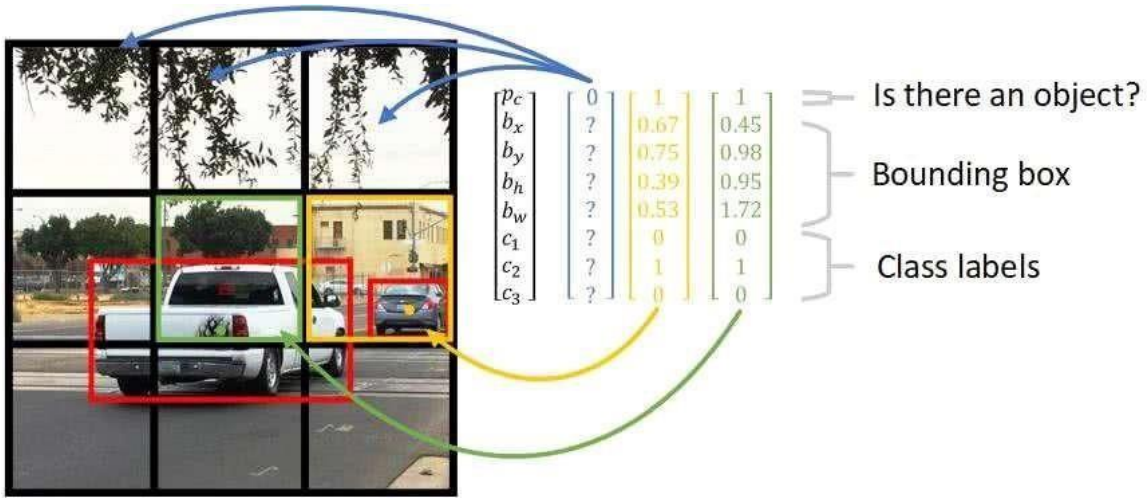


Figure 5: Sliding Window

Next, we shall apply the image classification and localization algorithm on each grid cell.In the image each grid cell, the target variable is defined as Yi,j=[pcbxbybhbwc1c2c3c4]T(6) Do everything once with the convolution sliding window. Since the shape of the target variable for each grid cell in the image is 1 × 9 and there are 9 (3 × 3) grid cells, the final output of the model will be:

final output = 3x3x9

The advantages of the YOLO algorithm is that it is very fast and predicts much more accurate bounding boxes. Also, in practice to get the more accurate predictions, we use a much finer grid, say 19 × 19, in which case the target output is of the shape 19 × 19 × 9.

# LITERATURE SURVEY

In various fields, there is a necessity to detect the target object and also track them effectively while handling occlusions and other included complexities. Many researchers (Almeida and Guting 2004, Hsiao-Ping Tsai 2011, Nicolas Papadakis and Aure lie Bugeau 2010 ) attempted for various approaches in object tracking. The nature of the techniques largely depends on the application domain. Some of the research works which made the evolution to proposed work in the field of object tracking are depicted as follows.

OBJECT DETECTION is an important task, yet challenging vision task. It is a critical part of many applications such as image search, image auto-annotation and scene understanding, object tracking. Moving object tracking of video image sequences was one of the most important subjects in computer vision. It had already been applied in many computer vision fields, such as smart video surveillance (Arun Hampapur 2005), artificial

intelligence, military guidance, safety detection and robot navigation, medical and biological application.

In recent years, a number of successful single-object tracking system appeared, but in the presence of several objects, object detection becomes difficult and when objects are fully or partially occluded, they are obtruded from the human vision which further increases the problem of detection. Decreasing illumination and acquisition angle. The proposed MLP based object tracking system is made robust by an optimum selection of unique features and also by implementing the Adaboost strong classification method.

Background Subtraction method by Horprasert et al (1999), was able to cope with local illumination changes, such as shadows and highlights, even globe illumination changes. In this method, the background model was statistically modelled on each pixel. Computational colour mode, include the brightness distortion and the chromaticity distortion which was used to distinguish shading background from the ordinary background or moving foreground objects.

The background and foreground subtraction method used the following approach. A pixel was modelled by a 4-tuple [Ei, si, ai, bi], where Ei- a vector with expected colour value, si - a vector with the standard deviation of colour value, ai the variation of the brightness distortion and bi was the variation of the chromaticity distortion of the ith pixel. In the next step, the difference between the background image and the current image was evaluated. Each pixel was finally classified into four categories: original

background, shaded background or shadow, highlighted background and moving foreground object. Liyuan Li et al (2003), contributed a method for detecting foreground objects in non-stationary complex environments containing moving background objects. A Bayes decision rule was used for classification of background and foreground changes based on inter-frame colour co-occurrence statistics. An approach to store and fast retrieve colour cooccurrence statistics was also established. In his method, foreground objects were detected in two steps. First, both the foreground and the background changes are extracted using background subtraction and temporal differencing. The frequent background changes were then recognized using the Bayes decision rule based on the learned colour co-occurrence statistics. Both short-term and long term strategies to learn the frequent background changes were used. An algorithm focused on obtaining the stationary foreground regions as said by Álvaro Bayona et al (2010), which was useful for applications like the detection of abandoned/stolen objects and parked vehicles. This algorithm mainly used two steps Firstly, a sub-sampling scheme based on background subtraction techniques was implemented to obtain stationary foreground regions. This detects foreground changes at different time instants in the same pixel locations. This was done by using a Gaussian distribution function. Secondly, som modifications were introduced on this base algorithm such as thresh holding the previously computed subtraction. The main purpose of this algorithm was reducing the amount of stationary foreground detected.

Template Matching is the technique of finding small parts of an image which match a template image. It slides the template from the top left to the bottom right of the image and

compares for the best match with the template. The template dimension should be equal to the reference image or smaller than the reference image. It recognizes the segment with the highest correlation as the target. Given an image S and an image T, where the dimension of S was both larger than T, output whether S contains a subset image I where I and T are suitably similar in pattern and if such I exists, output the location of I in S as in Hager and Bellhumear (1998). Schweitzer et al (2011), derived an algorithm which used both upper and lowers bound to detect 'k' best matches. Euclidean distance and Walsh transform kernels are used to calculate match measure. The positive things included the usage of priority queue improved quality of decision as to which bound-improved and when good matches exist inherent cost was dominant and it improved performance. But there were constraints like the absence of good matches that lead to queue cost and the arithmetic operation cost was higher. The proposed methods dint use queue thereby avoiding the queue cost rather used template matching. Visual tracking methods can be roughly categorized in two ways namely, the feature-based and region-based method as proposed by Ken Ito and Shigeyuki Sakane (2001). The featurebased approach estimates the 3D pose of a target object to fit the image features the edges, given a 3D geometrical model of an object. This method requires much computational cost. Region-based can be classified into two categories namely, parametric method and view-based method. The parametric method assumes a parametric model of the images in the target image and calculates optimal fitting of the model to pixel data in a region. The view-based method was used to find the best match of a region in a search area given the reference template. This has the advantage that it does not require much computational complexity as in the feature-based approach.

# Proposed model

## SqueezeNet:

SqueezeNet is name of a DNN for computer vision. SqueezNet is developed by researchers at DeepScale, University of California, Berkeley, and Stanford University together. In SqueezeNet design, the authors goal is to create a smaller neural network with few parameters that can more easily fit into memory of computer and can more easily be transmitted over a computer network. SqueezeNet is originally released in 2016. This original version of SqueezeNet was implemented on top of the Caffe deep learning software framework. The open-source research community ported SqueezeNet to a number of other deep learning frameworks. And is released in additions, in 2016, Eddie Bell released a part of SqueezeNet for the Chainer deep learning framework. in 2016, Guo Haria released a part of SqueezeNet for the Apache MXNet framework. 2016, Tammy Yang released a port of SqueezeNet for the Keras framework. In 2017, companies including Baidu, Xilnx, Imagination Technologies, and Synopsys demonstrated SqueezedNet running on low-power processing platforms such as smartphones, FPGAs, and custom processors. SqueezeNet ships as part of the source code of a number of deep learning frameworks such as PyTorch, Apache MXNet, and Apple CoreML.In addition, 3rd party developers have created implementation of SqueezeNet that are compatible with frameworks such as TensorFlow. Below is summary of frameworks that support SqueezeNet.

## InceptionV3:

Inception v3 is widely used as image recognition model that has showed to obtain accuracy of greater than 78.1% on the ImageNet dataset. The model is the culmination of many ideas developed by researchers over years. It is based on “Rethinking the Inception Architecture Computer Vision” by Szegedy The model is made of symmetric and asymmetric building blocks, including convolutions, average pooling, max pooling, concats, dropouts, and fully connected layers. Batchnorm is used more throughout the model and applied to activation inputs. Loss is computed via Softmax. A high-level diagram of the model is shown below:

## DenseNet:

DenseNet stands for Densely Connected Convolutional Networks it is one of the latest neural networks for visual object recognition. It is similar to ResNet but has some fundamental differences. With all improvements DenseNets have one of the lowest error rates on CIFAR/SVHN datasets: Error rates on various datasets And for ImageNet dataset DenseNets require fewer parameters than ResNet with same accuracy: Comparison of the DenseNet and ResNet Top-1 error rates on the ImageNet classification dataset as a function of learned parameters and flops during test-time This post assumes past knowledge of neural networks and convolutions. This mainly focus on two topics: Why does dense net differs from another convolution networks. What are the difficulties during the implementation of DenseNet in TensorFlow. If you know how DenseNets works and interested only in TensorFlow implementation feel free to jump to the second chapter or check the source. If you are not familiar with any of these topics to attain knowledge

Compare DenseNet with other convolution networks available Usually Convolution networks work such a way that We have an initial image, say having a shape of (29, 34, 31). After we apply set of convolution or pooling filters on it, squeezing dimensions of width and height and increasing features dimension. So the output from the Li layer is input to the Li+1 layer.

ResNet architecture is proposed for Residual connection, from previous layers to the present layer. input to Li layer is obtain by addition of outputs from previous layers.

In contrast, DenseNet paper proposes concatenating outputs from the previous layers instead of using the summation. So, let's imagine we have an image with shape(28, 28, 3). First, we spread image to initial 24 channels and receive the image (28, 28, 24). Every next convolution layer will generate k=12 features, and remain width and height the same. The output from $L_i$ layer will be (28, 28, 12). But input to the $L_{i+1}$ will be (28, 28, 24+12), for $L_{i+2}$ (28, 28, 24 + 12 + 12) and so on. After a while, we receive the image with same width and height, but with plenty of features (28, 28, 48). All these N layers are named Block in the paper. There's also batch normalization, nonlinearity and dropout inside the block. To reduce the size, DenseNet uses transition layers. These layers contain convolution with kernel size = 1 followed by 2x2 average pooling with stride = 2. It reduces height and width dimensions but leaves feature dimension the same. As a result, we receive the image with shapes (14, 14, 48) Now we can again pass the image through the block with N convolutions. With this approach, DenseNet improved a flow of information and gradients throughout the network, which makes them easy to train. Each layer has direct access to

the gradients from the loss function and the original input signal, leading to an implicit deep supervision. Also at transition layers, not only width and height will be reduced but features also. So if we have image shape after one block (28, 28, 48) after transition layer, we will get (14, 14, 24).


Figure 6: DenseNet approach

Where theta — some reduction values, in the range (0, 1). Whenusing bottleneck layers with DenseNet, maximum depth will be divided by 2. It means that if you have 16 3x3 convolution layers with depth 20 previously (some layers are transition layers),you will now have 8 1x1 convolution layers and 8 3x3 convolutions.Last, but not least, about data preprocessing. In the paper per channel normalization was used. With this approach, every image channel should be reduced by its mean and divided by its standard deviation. In many implementations was another normalization used — just divide every image pixel by 255, so we have pixels values in the range [0, 1]. Note about numpy implementation of per channel normalization. By default images provided with data type unit. Before any manipulations, It is advised to convert the images to any float representation. Because otherwise, a code will fail without any warnings or errors.

# SYSTEM REQUIREMENT

Install Python on your computer system

1. Install ImageAI and its dependencies like TensorFlow, Numpy, OpenCV, etc.

2. Download the Object Detection model file (Retinanet)

5.1 Steps to be followed :-

1) Download and install Python version 3 from official Python Language website

https://python.org

2) Install the following dependencies via pip command:

i. Tensorflow:

Tensorflow is an open-source software library for dataflow and differentiable programming across a range of tasks. It is an symbolic math library, and is also used for machine learning application such as neural networks,etc.. It is used for both research and production by Google. Tensorflow is developed by the Google Brain team for internal Google use. It is released under the Apache License 2.0 on November 9,2015. Tensorflow is Google Brain's second-generation system.1st Version of tensorflow was released on February 11, 2017.While the reference implementation runs on single devices, Tensorflow can run on multiple CPU's and GPU (with optional CUDA and SYCL extensions for general-purpose computing on graphics processing units). TensorFlow is available on various platforms such as64-bit Linux, macOS, Windows, and mobile computing platforms

including Android and iOS. The architecture of tensorflow allows the easy deployment of computation across a variety of platforms (CPU's, GPU's, TPU's), and from desktops - clusters of servers to mobile and edge devices. Tensorflow computations are expressed as stateful dataflow graphs. The name Tensorflow derives from operations that such neural networks perform on multidimensional data arrays, which are referred to as tensors.

pip install tensorflow -command

ii. Numpy:

NumPy is library of Python programming language, adding support for large, multidimensional array and matrix, along with large collection of high-level mathematical function to operate over these arrays. The ancestor of NumPy, Numeric, was originally created by Jim Hugunin with contributions from several developers. In 2005 Travis Olphant created NumPy by incorporating features of computing Numarray into Numeric, with extension modifications. NumPy is open-source software and has many contributors.

pip install numpy -command

iii. SciPy:

SciPy contain modules for many optimizations, linear algebra, integration, interpolation, special function, FFT, signal and image processing, ODE solvers and other tasks common in engineering. SciPy abstracts majorly on NumPy array object ,and is the part of the NumPy stack which include tools like Matplotlib, pandas and SymPy, etc., and an expanding set of scientific computing libraries. This NumPy stack has similar uses to other applications such as MATLAB,Octave, and Scilab. The NumPy stack is also sometimes

referred as the SciPy stack. The SciPy library is currently distributed under BSDlicense, and its development is sponsored and supported by an open communities of developers. It is also supported by NumFOCUS, community foundation for supporting reproducible and accessible science.

pip install scipy -command

iv. OpenCV:

OpenCV is an library of programming functions mainly aimed on real time computer vision. originally developed by Intel, it is later supported by Willow Garage then Itseez. The library is a cross-platform and free to use under the open-source BSD license.

pip install opencv-python -command

v. Pillow:

Python Imaging Library is a free Python programming language library that provides support to open, edit and save several different formats of image files. Windows, Mac OS X and Linux are available for this.

pip install pillow -command

vi. Matplotlib:

Matplotlib is a Python programming language plotting library and its NumPy numerical math extension. It provides an object-oriented API to use general-purpose GUI toolkits such as Tkinter, wxPython, Qt, or GTK+ to embed plots into applications.

pip install matplotlib - command

vii. H5py:

The software h5py includes a high-level and low-level interface for Python's HDF5 library. The low interface expected to be complete wrapping of the HDF5 API, while the high-level component uses established Python and NumPy concepts to support access to HDF5 files, datasets and groups. A strong emphasis on automatic conversion between Python (Numpy) datatypes and data structures and their HDF5 equivalents vastly simplifies the process of reading and writing data from Python.

pip install h5py

viii. Keras

Keras is an open-source neural-network library written in Python. It is capable of running on top of TensorFlow, Microsoft Cognitive Toolkit, Theano, or PlaidML. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible.

pip install keras

ix. ImageAI

ImageAI provides API to recognize 1000 different objects in a picture using pre-trained models that were trained on the ImageNet-1000 dataset. The model implementations provided are SqueezeNet, ResNet, InceptionV3 and DenseNet.

pip3 install imageai --upgrade

3) Download the RetinaNet model file that will be used for object detection using following link

https://github.com/OlafenwaMoses/ImageAI/releases/download/1.0/resnet50_coco_best_v2.0.1.h5

Copy the RetinaNet model file and the image you want to detect to the folder that contains the python file.

# References


- Agarwal, S., Awan, A., and Roth, D. (2004). Learning to detect objects in images via a sparse, part-based representation. IEEE Trans. Pattern Anal. Mach. Intell. 26,1475–1490. doi:10.1109/TPAMI.2004.108

- Alexe, B., Deselaers, T., and Ferrari, V. (2010). "What is an object?," in Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on (San Francisco,CA: IEEE), 73–80. doi:10.1109/CVPR.2010.5540226

- Aloimonos, J., Weiss, I., and Bandyopadhyay, A. (1988). Active vision. Int. J.Comput. Vis. 1, 333–356. doi:10.1007/BF00133571models," in Computer Vision-ECCV 2012 (Florence: Springer),836–849. 6. Azzopardi, G., and Petkov, N. (2013). Trainable cosfire filters for keypoint detection and pattern

- Benbouzid, D., Busa-Fekete, R., and Kegl, B. (2012). "Fast classification using sparse decision DAGs," in Proceedings of the 29th International

Conference on Machine Learning (ICML-12), ICML ‘12, eds J. Langford and J. Pineau (New York, NY:Omnipress), 951–958.

- Bengio, Y. (2012). “Deep learning of representations for unsupervised and transfer learning,” in ICML Unsupervised and Transfer Learning, Volume 27 of JMLRProceedings, eds I. Guyon, G. Dror, V. Lemaire, G. W. Taylor, and D. L. Silver(Bellevue: JMLR.Org), 17–36.